LEARNING MADE EASY

DataOps.live Special Edition

Data Products

for dummies
A Wiley Brand

Build and manage Data Products

Learn from real-world practitioners

Achieve 10x developer productivity

Brought to you by

DataOps.live

Sanjeev Mohan
Guy Adams
Justin Mullen

About DataOps.live

DataOps.live — the Data Products company — delivers productivity breakthroughs for data teams by enabling agile DevOps automation (#TrueDataOps) and a powerful Developer Experience (DX) to modern data platforms. The DataOps.live SaaS platform brings automation, orchestration, continuous testing, and unified observability to deliver the Data Products you want at the speed the business needs. DataOps.live is a global company funded by Anthos Capital, Notion Capital and Snowflake Ventures, with enterprise clients including Roche Diagnostics and OneWeb.

Want to get started building Data Products? Try DataOps.live for free at www.dataops.live.

A special thanks to our contributors Omar Khawaja, Miguel Morgado, Paul Rankin, Kent Graziano and Wayne Eckerson.

Data Products

Data Products

DataOps.live Special Edition

by Sanjeev Mohan, Guy Adams, and Justin Mullen

for dummies®
A Wiley Brand

Data Products For Dummies®, DataOps.live Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN 978-1-394-21555-3 (pbk); ISBN 978-1-394-21556-0 (ebk)

Publisher's Acknowledgments

Development Editor: Rachael Chilvers

Project Editor: Pradesh Kumar

Acquisitions Editor: Traci Martin

Editorial Manager: Rev Mengle

Business Development Representative: Matt Cox

Table of Contents

Introduction

Data Products are the future for how organizations will organize, develop, and deploy data assets and data applications, using everything that works well about how data has operated up to now, but with the benefits of decades of innovation from the software world. However, getting started can be daunting, and that's where this invaluable book can help.

About This Book

Welcome to *Data Products For Dummies*, a DataOps.live Special Edition.

We cover several topics, including the following:

- A quick introduction to Data Products and how they're different to data as a product (Chapter 1)
- The current state of analytics and why it has to change and modernize (Chapter 2)
- What a Data Product is (and isn't), along with some examples (Chapter 3)
- The many benefits of Data Products for consumers, including real-life examples to inspire you (Chapter 4)
- The ten attributes of a Data Product, and the FAIR (findable, accessible, interoperable, and reusable) data principles (Chapter 5)
- How to build and deploy Data Products, the roles involved, and what you can learn from the Data Product maturity model (Chapter 6)
- The place of Data Products in the data and analytics ecosystem (Chapter 7)
- Ways to start building Data Products today (Chapter 8), in the famous *For Dummies* Part of Tens final chapter

Icons Used in This Book

Throughout this book, different icons are used to highlight important information. Here's what they mean:

EXAMPLE

We illustrate the benefits of Data Products with examples.

REMEMBER

The Remember icon points out things you need to remember when searching your memory bank.

TECHNICAL STUFF

Sometimes we give you a few tidbits of research or facts beyond the basics. So if you like to know the technical details, watch out for this icon.

TIP

The Tip icon highlights information that can make doing things easier or faster.

WARNING

The Warning icon alerts you to things that can be harmful to you or your company.

Beyond the Book

This book will help you discover more about Data Products. If you want resources beyond what this short book offers, visit:

- TrueDataOps podcasts: `https://www.dataops.live/truedataops-podcast`
- *DataOps For Dummies*: `https://www.dataops.live/dataops-for-dummies-download`
- DataOps.live website: `www.dataops.live`
- *Data Mesh* by Zhamak Dehghani: `https://www.oreilly.com/library/view/data-mesh/9781492092384/`
- TrueDataOps.org: `www.truedataops.org`
- It Depends podcast: `https://www.youtube.com/c/SanjeevMohan`

IN THIS CHAPTER

- Understanding the definition of a Data Product
- Defining the usefulness of Data Products
- Laying out the purpose of this book

Chapter 1
Introducing Data Products

Before we can harness the magic of Data Products, we need to first understand what a Data Product is and, equally importantly, what it is *not*. For example, do you know the difference between a Data Product and data as a product? You will soon!

Our hope is that by the end of this book, you'll have moved from theory to practice, applying the guidance that comes from our collective experience to your real-world needs. It's time to unlock the value of Data Products.

Let's get started!

Comparing Data as a Product versus Data Products

Understanding the difference between "Data Products" and "data as a product" is fundamental to the concepts, approaches, and pragmatic guidance provided by this book.

However, "Data Products" have been poorly defined to date, which hasn't been helpful. An early definition of a Data Product was produced by DJ Patil, former United States chief data scientist, in his book *Data Jujitsu: The Art of Turning Data into Product*, 2012. He defined a Data Product as "a product that facilitates an end goal through the use of data." This suggests that *any* digital product can be considered a Data Product if it uses data to facilitate a goal, which proves to be an unworkably wide definition. We'll refine this definition after understanding "data as a product."

The "data as a product" concept is one of the four principles of a data mesh paradigm introduced by Zhamak Dehghani in 2019, and it sets out the way we should think about data as a product. The key characteristics of data as a product include being:

- Discoverable
- Understandable
- Trustworthy
- Natively accessible
- Valuable on its own

So, Dehghani provided a tighter and much clearer definition than we previously had, and this is a definition we subscribe to and advance in this book. "Data as a product" provides us a way of thinking about how we should define, describe, build, govern, and life cycle manage data in a product-based agile approach.

"Data Products" are therefore the tangible implementation of this way of thinking. They're the things you create and with which the business stakeholders interact. Data Products can take many forms, including:

- Simple business reports or dashboards that provide insights into sales performance or customer behavior.
- Complex machine learning models that can predict future trends, a customer's next action, or make recommendations based on past data and make this available to stakeholders.
- Endpoints of secure raw, curated, or aggregated and modelled datasets that are securely shared internally and externally with different cohorts of stakeholders.

REMEMBER

Data Products therefore become the base units of management for an organization's data. While a Data Product requires many things to define it, at its core a Data Product should:

- Convey trust and the product features meant to solve business problems
- Have measurable value
- Have an owner who's responsible for delivering value throughout the product's life cycle from design to retirement

For the purposes of this book, we'll refer to any and all of these as a Data Product.

Taking a Pragmatic View of Data Products

As with any new concept, some material about Data Products has already been written, and much more is bound to be written. However, to date most of this has been abstract and theoretical. There are some great works in this area and they hold a lot of value, but there's a huge gap between these writings and the people on the ground actually trying to make this stuff work.

As with all new things, pioneers who believe in these abstract concepts will step up to say, "I think we can make this work." The authors of this book *are* these pioneers and have also been privileged to work with pioneers in real companies who have made this work without guidance, experience, or blueprints. They've made mistakes of course, and some of those mistakes were painful! But, without exception, these organizations have been wildly successful. They're now the cornerstones of many industry conferences, used as case studies, and their metrics for value and productivity are truly off the charts.

This book exists as an antidote to the abstract and theoretical works. It takes the lessons and learnings from these initial pioneers and condenses these into a much more practical and pragmatic guide to building Data Products.

Thinking of Data Products as the Base Unit of Management

In Chapter 5 we'll discuss parallels between Data Products and software products. However, one parallel is more important than all the others. For a company or group developing a software product, that product is the *base unit of management*. The software product is the thing that has a roadmap; it's the thing that you version, that you release, that has an owner; it's the thing against which value is measured; it's the thing that has a life cycle. Of course, these software products have many things inside them — lines of code, libraries, configurations, UIs, unit tests, documentation — and it's these things that software engineers spend most of their time developing but, at all times, they're working on a specific "software product."


TIP

A line of code or unit test that isn't part of a software product is orphaned and will cause problems everywhere. Making Data Products the base unit of management for an organization's data is *the most fundamental first step* any organization must take.

Don't worry if on day one you have few actual Data Products, perhaps only two to five, as the software world didn't get there overnight either. But making the decision that this approach is your future is the most important step you'll take in the next five years. Once you've made this decision you'll most likely realize and recognize the following:

- Everything else is details around how — this book will help you through many of those details in a pragmatic way with real world advice, but most importantly you'll already be on the right track!
- Stakeholders will rapidly gravitate to this way of thinking, and you'll move from having only two to five Data Products to realizing that you have many more, often 10-100 current and desired Data Products that you can start to life cycle manage through conception, build, change, maintenance, and eventual retirement.

This book exists to help you benefit from the experience of pioneers and practitioners leading the way in describing and defining "data as a product" and practically building and managing "Data Products." It kicks off where the abstract work finishes and hopefully provides some pragmatic guidance, grounded in today's realities, to help the next wave of practitioners and adopters to get the same enormous value without quite so many battle scars!

IN THIS CHAPTER

- Taking a look at problems with the modern data stack
- Identifying impact to business productivity
- Acknowledging the curse of cloud computing
- Addressing the need for discipline

Chapter 2
The Imperative to Change

Today's data stack is a natural progression from the Hadoop stack of the 2010s. In the Hadoop stack, the emphasis was on collecting data quickly and then unleashing its raw power to data consumers. Arguably, it ushered in the era of big data. Suddenly, large amounts of data could be processed in parallel using the map-reduce concepts.

However, it soon became clear that raw processing power wasn't enough. Data consumers needed context and governance, which was missing in this stack.

Fast forward to the 2020s, and a modern data stack arose to solve data management issues through purpose-built solutions with very deep functionality. The stack was so popular that soon it was bursting with new product offerings. In the 2023 MAD landscape (`https://mattturck.com/mad2023/`), the number of products listed rose from 139 in 2012 to 1,460. This has led to the integration overhead, rise in costs, and lack of transparency in data flows.

At this point, business is still asking the same question it's been asking for years: How can I get faster outcomes from the data? Of course, businesses expect the data to be always available and trustworthy.

Is that too much to ask?

Introducing the Current State of Analytics

The way we've built analytics in the cloud isn't sustainable: Something needs to give. We've written this book to address the situation. However, before we delve into the mechanics of Data Products, let's examine the current state of affairs in data and analytics.

Identifying business challenges

Let's start by examining what the ultimate consumers of data want. This isn't typical for technical teams. But the reality is that business professionals don't care about IT techno-babble, like data mesh, data fabric, and Data Products. They want:

- **Faster time to value:** Decisions need to be made with very low latency. Often (or even usually), data engineering teams are bottlenecks. If business is able to access the data using a business-friendly semantic layer, it puts them in control. This is "self-service."
- **Complete data:** With the proliferation of software as a service (SaaS) tools, business teams have the autonomy to deploy any tool that meets their needs. This has resulted in fragmentation of data, which hinders analysis. The data sources may be external or internal, in multi-structured form, and may be ingested in batch or in real-time. This has also created more shadow IT, with individual teams choosing technologies with no overarching vision or architecture.

- **Governance:** This is a loaded word and some companies have banned this term. What businesses want is high-quality data they can trust and that's reliably available — and it must comply with all relevant regulations and standards.

Savvy data-driven companies have initiated steps to resolve these challenges. The solution has to address all aspects relating to people and processes as well as technology. However, this book is concerned with solutions that address the technical challenges.

Defining technical challenges

As the modern data stack becomes more complex, technical challenges abound. Every new product in the stack introduces its own data silo (and perhaps a far newer concept, the metadata silo) and a higher integration overhead.

WARNING

This fragmentation doesn't do anything to accelerate business outcomes, but instead introduces challenges:

- **Poor data quality:** When data changes hands at so many interfaces in a pipeline, there's no single point of accountability. Data quality problems fall through the cracks and take a great deal of time and resources to debug.
- **Unreliable data:** Failures in a complex system can go unnoticed if proper monitoring steps aren't incorporated. These brittle pipelines put organizations at risk of reporting incorrect information.
- **Manual processes:** To overcome the first two challenges, organizations used to depend on manual scripts and processes. This isn't scalable as the volume, variety, and usage of data increases rapidly. Manual processes slow down the workloads needed to generate data-driven decisions and introduce more unnecessary risks.
- **Insufficient security and privacy:** Complexity is the enemy of security. Lack of transparency and accountability within the pipeline cause security loopholes. Without centralized unified data governance, security, and privacy policy enforcement, organizations can't guarantee adherence to regulatory compliances.

- **Lack of reusability and discovery:** In an era of cost containment, organizations shouldn't have to reinvent the wheel, but instead leverage existing artifacts, such as reports, machine learning (ML) models, and transformations logic. However, these need to be packaged adequately for consumption and made discoverable.

WARNING

If these challenges aren't addressed, they'll block innovation and data-driven decisions. The overall effect is poor developer experience and lower productivity. When this happens, it directly reduces business productivity.

Addressing business impact

In traditional on-premises environments, developers used many band-aids to match numbers and manually fix problems. However, as data volumes increased, so have the numbers of eager data consumers who want to run ever-more use cases. This leads to further headaches:

- **Too late and too little:** By the time you get answers to your problems, you may have missed the opportunity. Take an example of a marketing campaign that wants to fine-tune its messaging in real time based on an analysis of clickstream data, surveys, and historical buying patterns. All this data resides in different locations and yet must be presented to the data consumer as a single coherent package.
- **Wasted resources:** Errors happen. In fact, the larger your system, the more places where something can go wrong. If your data consumers are the first people to find out errors, it further erodes trust in data teams. Now the data team must firefight and try to locate broken pipelines. They're no longer focused on business strategic projects.
- **High cost:** Manual processes, debugging errors, and a lack of transparency in pipelines all lead to higher costs.
- **Lack of trust:** Business teams are unable to trust the data quality. In addition, an inability to ensure that data privacy meets regulatory compliance further reduces data's utility.

TIP

It's imperative to rethink the way we deliver business outcomes.

Taking a Holistic Approach to Modernization

Delivering outstanding business productivity is every organization's dream. It turns out that data teams have a central role to play in this. High developer productivity enables businesses to innovate faster and deliver high-quality data outcomes rapidly.

Software engineering made huge strides in the early 2000s, with the introduction of disciplines like product management, agile development, and DevOps. Data teams are now beginning to adopt the same principles. With the advent of cloud computing, these trends have accelerated. If current advancements continue, the data applications space will look very different in five years.

REMEMBER

The advantages of higher business and data team productivity not only helps organizations to become more efficient but also to increase customer satisfaction. You can manage risks better, improve compliance, and make faster decisions.

Avoiding the hidden price of cloud computing

The hidden cost of cloud computing is that the risks, overheads, and costs move from the build phase to the operational phase.

Cloud computing has eased the process of coding to a point where anyone with basic technical skills can quickly onboard to a cloud development environment and start building. This is great for agility. Isn't this what we've been asking for? Be careful what you ask for.

The "Day 0" culture of the cloud celebrates builders, but this comes at the cost of maintainability. In a world where everyone likes to build new stuff, no one likes to maintain. What happens when new use cases emerge and systems need to be refactored. Did you build for future-use cases?

WARNING

In addition, developers lacking software engineering training can easily introduce inefficiencies. For example, in the interest of speed, they may make multiple copies of data sources, leaving behind data silos and technical debt. Indeed, managing technical debt has become an entire subdiscipline within software development. The data world may need to follow suit.

Finding a disciplined approach

What's the solution? On the one hand, you want to welcome self-service agility from the business user. On the other, you want a streamlined and repeatable development process. The answer lies in ensuring that you have proper discipline across various dimensions:

- **Fiscal control:** Rampant and erratic development leads to cost overruns. Temporary objects are created to handle jobs and are never purged. This bloat also affects performance and opens the company to unnecessary security and privacy risks.

  

  TIP

  The solution to encourage fiscal discipline is an area called FinOps (`https://www.dataops.live/spendview`).

- **Avoiding reinvention:** Data teams working in silos tend to reinvent the wheel. Every time they need to perform a task, they rebuild data models, reports, and transformation scripts. Not only is this wasted effort, it also slows down business decision making.

  

  TIP

  Issues pertaining to inefficient data management practices are solved by using DataOps. This space consists of best practices for environment and configuration management, continuous testing, integration and deployment, monitoring and auditing, and automating as many tasks as possible, such as orchestration of jobs. For more information, please read *DataOps For Dummies* (`https://www.dataops.live/dataops-for-dummies-download`).

- **Project management:** The prevailing approach to developing data outcomes is project-based. Once a project ends, the team is assigned to other initiatives. This creates a problem. Nobody is responsible for the continuous maintenance of

the data outcomes. Often, if there's a problem with data quality, the first person to detect this is the consumer, such as a CFO using a dashboard. It's a huge problem that the team responsible for producing data outcomes is the last to know. No wonder data teams have struggled to gain the trust of business teams. So where do you find a team to fix this problem? No one is up to speed on it, and it may have used its own set of technologies that aren't used elsewhere. Project-based data is the enemy of scale and efficiency.

The antidote is to bring product management practices into the data teams — just like software development incorporated agile development practices two decades ago.

The remainder of this book focuses on the product management aspects for data outcomes; in other words, the concept of Data Products.

Data Products are a transformational way to alleviate the limitations of current development practices. Done properly, they underpin the trust, efficiency, accuracy, and consistency of data outcomes — the hallmarks of every successful data-driven organization.

IN THIS CHAPTER

- Determining business and technical definitions of Data Products
- Looking at components of Data Products
- Exploring examples of Data Products

Chapter 3
What is a Data Product?

Let's work on getting a clear and unambiguous understanding of what a Data Product is, and what it is not.

ChatGPT has taken the world by storm. It provides an excellent user experience, remembers the intent of previous questions, then forms a chain of answers. In this way, it looks and behaves like a Data Product. But is it?

How about a data mart? It's built to answer specific business domain questions, so surely it's a Data Product?

The answer to both of these questions is no. This chapter explores what makes up a true Data Product.

Defining Data Products

Definitions need to be treated with caution. As many people can disagree as agree. We have to be flexible, ensuring that textbook definitions don't constrain creativity and innovation. However, it's valuable to baseline our understanding because without common definitions, we run the risk of slapping the label "Data Product" on any data output.

Purists will say that a Data Product is nothing new. For others, a Data Product is less about what they're building and more about how they build and deploy it. Indeed, applying product management best practices is a crucial defining factor of Data Products.

In addition to product management aspects, a Data Product offers superior, consistent, and reliable data access. If done properly, this enables consumers to get better answers to their questions to improve business decisions and outcomes.

REMEMBER

A successful Data Product:

- Should demonstrate two key attributes: a great user experience and trustworthiness.
- Has an owner who's accountable for its quality and reliability.
- Is a self-contained interface to get answers to all kinds of questions.
- In most cases, is consumed via a self-service interface. Most Data Products are read-only. (Head to Chapter 5 for more about the attributes of Data Products.)

Figure 3-1 summarizes the components of a mature Data Product. It's of course possible to start with a far more simplistic version of a Data Product as you begin your journey.

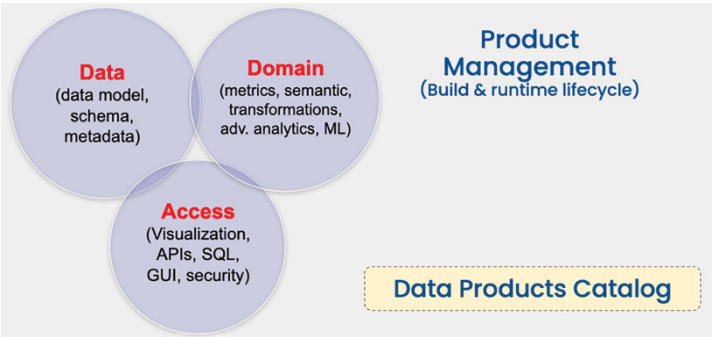

FIGURE 3-1: The components of a mature Data Product.

REMEMBER

Unlike older monolithic data outcomes, a Data Product decouples and isolates business needs into self-contained units that contain data and business logic and can be accessed by various means. Another departure from the norm is that a single business team

is responsible for managing and deploying the Data Product. In some ways, it is akin to the microservices paradigm in software engineering (see Chapter 5).

Taking a look under the hood

A Data Product is a combination of:

- **Data sets:** They may be in a table, view, a machine learning (ML) model, or a stream. The data may be raw data or curated data integrated from multiple sources. The Data Product must publish its data model.
- **Domain model:** This adds the necessary business logic and a semantic layer — a "describer." The semantic layer abstracts the technical layout of the storage layer and instead exposes business-friendly terms to users. For example, a column, 'cust_001' may be translated into 'customer name.' This layer also stores various calculations and metrics.
- **Multi-faced data access:** This provides various approaches that suit different target consumers. For example, some analysts may prefer SQL access, while another team or downstream app may prefer APIs, while others prefer visualization options. To maintain data privacy and security, access control policies are enforced.

TIP

A Data Products Catalog is also critical. This makes Data Products discoverable and with all the necessary attributes documented. It acts as a marketplace for Data Products. The catalog documents expected inputs and outputs and acts as a "contract" between the Data Products development team and the data consumers. Once the Data Products are consumed, the catalog can be used to visualize its usage telemetry and to see its version history.

The Data Products catalog may not be a standalone product or a new addition to your burgeoning modern data stack. Often, it's an extension of the existing metadata catalog (head here for more details: `https://www.forbes.com/sites/forbesbusinesscouncil/2023/02/07/making-data-products-mainstream-introducing-the-data-products-catalog/`).

Exploring the business characteristics of Data Products

Business users don't care how IT people label and categorize technology. They're focused on solving challenges or providing new opportunities for the organization. For IT staff to effectively explain a Data Product to the business, it must be free of technical jargon.

REMEMBER

Today's users go to a dashboard for analytical answers, an ML model for prescriptive insights, and search databases for diagnostic queries. By contrast, a Data Product offers *unified self-contained access* to get answers to different types of questions: diagnostic, predictive, prescriptive, analytical, and so on.

EXAMPLE

To separate a real-life Data Product from business talk, we can get some help from the physical world of products. Imagine a box of your favorite cereal. The box has the goods (say, cornflakes), a description of its ingredients' nutrition details, the expiration date, and its price. You can definitely find this product in the cereal aisle of a grocery store and purchase it.

Imagine you can somehow procure the cereal without the box. Is it still a product? After all, the cereal hasn't changed, but based on the above description, it's no longer a product.

To start with, it can't give you a good experience from the outset. If you have questions about the freshness of the content, you have no way of knowing when it might go bad, nor can you go to the brand producer to request a refund. You don't know who the manufacturer is. It doesn't promote consumer "trust" and "experience" in its packaging.

Let's apply this to digital objects. Just as physical products have a brand, digital products must have an identity. This comprises a label, tag, user consent, purpose, and a statement of trust and reliability. With the Data Product, beyond the core function, it may include the output table schema, data dictionary, data distribution, semantic layer, metric, agreements and contracts, and other telemetry including intermediate snapshots to help service the product in runtime.

Today, the documentation, business logic, metric, and so on exist but aren't a part of the table. They're an afterthought and are "out

of band," like on a SharePoint site or in various different business intelligence (BI) tools. The documentation soon goes out of sync as the schemas evolve. Also, if an individual uses a different data access tool, the logic may not be available. This is common in the traditional approach that leads to duplication of effort and increases the chance of error.

REMEMBER

Simply publishing a data set doesn't make it a Data Product. It must have the other components, too, including:

- A product management process
- The domain wrapper comprising a semantic layer
- Business logic and metrics
- Access

The Data Product should also serve broad domain use cases. For instance, the marketing team may collect customer data from multiple sources, such as Salesforce, SAP, Marketo, Hubspot, website logs, and surveys, to produce a "customer master." This foundational data asset can then be combined with the rest of the components discussed earlier and be packaged as the marketing team's Data Product. Other domains, like sales and finance, can trust its data and use it to derive their own outcomes or even build their own Data Products. Data Products make data agreements more transparent and actionable between data producers and consumers.

Identifying technical characteristics

A Data Product abstracts the physical storage location of the content, which may be built using data sources on-premises or in multiple cloud providers. It also hides the complexity of data pipelines from data consumers. A pipeline may involve data movement, data virtualization, in-memory, caching, a lakehouse, or a fabric.

EXAMPLE

Let's go back to the cereal aisle! This abstraction is similar to a consumer not having to think about how their cereal was manufactured, packaged, and transported. In the past, we expected the business to understand technology to be as effective as possible. In a data-driven world, business users can confidently expect the same consistent outcome every time, picking up that box of cornflakes without needing to know any of the details.

Finally, the technical definition is incomplete without documenting the non-functional attributes the business needs, such as a repeatable experience, reliability, concurrency, response time, and uptime.

Will the Real Data Product Please Stand Up?

There's little agreement in the industry on what a tangible Data Product looks like. This section examines typical data outcomes.

Identifying table, schema, and view — oh my!

A table by itself is not a Data Product because it may have references to keys in other tables. In other words, it may not be self-contained. Some people disagree, as a table can easily be self-contained by flattening or denormalizing it. But does that constitute a Data Product?

It can be, if combined with a semantic layer — the user-friendly business representation of the data that makes use of common business terms. Why is this important?

REMEMBER

A Data Product aims to give its consumer a superior self-service user experience without them needing to know the physical details. In addition, it's abstracting the user from changes in the source schema. When the schema changes, the Data Product owner creates a new version of the Data Product and makes it available in the Data Product catalog.

In other words, product management aspects and attributes are critical in order for a Data Product to be called a Data Product.

If each table and its metrics become a Data Product, you'd soon have a chaotic unmanageable mess. And what's the point of a Data Product if each table is one?

Distinguishing data warehouses and data marts from Data Products

Data Products should follow the shift-left principle and be created by domain teams for an unbounded set of use cases. A Data Product more closely aligns with business domain entities, events, and their interactions and behaviors. The Data Product owner is accountable for delivering the Data Product's agreed quality, while the responsibility for defining data quality is done by the data consumer based on their requirements.

TECHNICAL STUFF

Data marts were built to answer specific business domain questions, so they surely must be a Data Product, right? Wrong. Data marts, data warehouses, data lakes, and lake houses are *data management platforms* as opposed to being a Data *Product.* Traditionally, a data mart is an IT deliverable that arrives after a long data warehouse build, at which time business needs may have already changed. If the product management approach was applied to a data mart, it could be used to develop Data Products or become a Data Product. A data mart product should be agile and support various modes of visualization, advanced analytics, and query engines.

Introducing examples of Data Products

A report, otherwise known as a dashboard, is one example of a Data Product. It has access to data and metrics. The access can be via APIs or a language like SQL. As expected, it must have a designated product owner and be built using product management principles. A datashare is a logical extension to this report provided to specific cohorts of internal or external users.

An ML model, perhaps around customer churn or sentiment analysis, follows the same criteria as above for reports and dashboards. This is another example of a Data Product.

Finally, a packaged application that meets well-defined business needs is an example of a Data Product. This application may be a React app or built using Streamlit, an open source user interface (UI) framework owned by Snowflake. It qualifies as a Data Product so long as it meets all the ownership and product management requirements covered in this chapter.

We could go on; this list of examples isn't complete as there are other types of Data Products, but it's time to look at the benefits.

IN THIS CHAPTER

- Learning about the benefits of Data Products
- Exploring real-life use cases

Chapter 4
Reaping the Benefits of Data Products

Wondering what Data Products have done for organizations today? Well, wonder no more! This chapter explains how powerful this approach has become for developing modern data applications.

Data is a victim of its own success. It's such a critical ingredient for the success of any organization that new use cases and consumers keep sprouting up from the dusty corners of every business.

WARNING

While this may be good news, data teams now face ever-tighter deadlines to deliver the goods. The traditional way of creating data applications isn't scalable and data teams are becoming a bottleneck. The older approach leads to data outcomes that lack accountability and data pipelines that aren't reliable.

This is where Data Products make their grand appearance. They help businesses to achieve better time to value from their data assets. Data Products are like a layer of abstraction over the raw data with a single purpose — to meet business needs with high quality and increase business productivity.

Exploring High Level Benefits of Data Products

Data Products are transforming many businesses, providing real-life benefits that make organizations far more efficient and give them competitive advantage.

Benefits are delivered across four main areas, as Figure 4-1 shows: user experience, trust, cost, and performance.

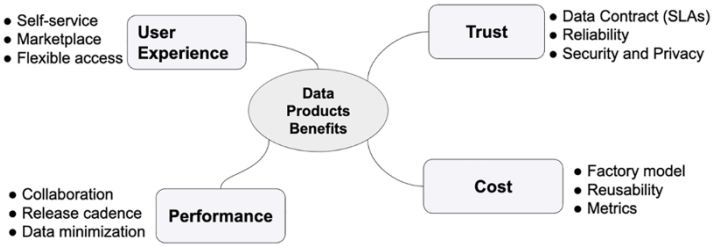

FIGURE 4-1: The benefits of Data Products.

Creating a superior user experience

Data Products liberate data by flipping the priorities: business value and use-cases first, followed by infrastructure and data management. Traditionally, data platforms and data management have taken precedence over use cases. The flipping of intent/prioritization alone is enough to explain why Data Products are so effective in delivering business advantage.

The overarching benefit of Data Products is that they unlock the insights stored inside corporate data assets to a wide range of data consumers. Data Products can help achieve the dream of data democratization. They do this through a superior user experience.

REMEMBER

Most data consumers aren't experts in data models and so are at a disadvantage in doing analysis that requires joining data from multiple sources. As a result, a semantic layer that applies common business terms is a critical part of Data Products as it provides a user-friendly interface.

Data Products alleviate the need to go to the raw data as they're built and deployed to provide authorized consumers with business value. Data access benefits include:

- **Self-service:** Consumers, including data engineers, data scientists, and citizen analysts, can access the Data Products. Consumers may also be outside the organization. Data Products provide a consistent and well-defined interface to all current and even future users.
- **Marketplace:** Data Products are searchable in a catalog or marketplace where the product's attributes are clearly described. (See Chapter 5 to learn more about Data Product attributes.) The marketplace also shows consumption patterns to help monitor their usage and value.
- **Flexible access:** Data Products are designed to allow multiple access approaches, such as low/no code or programmatic. Different departments and teams should be able to use the tool of their choice to access data. This includes analytical engines, IDEs, APIs like SQL or GraphQL, SDKs, reports, and dashboards. Soon, the access will be through large language models (LLMs), which will further shift the focus from data infrastructure and management toward business value.

REMEMBER

Data Products help businesses to become more productive as they remove friction between producers and consumers. Their biggest differentiator is to delight the consumer (and enable them to achieve more) with a superior user experience.

Earning trust

Data infrastructure today has built a reasonable supply chain from raw data sources to business intelligence (BI). But it hasn't ensured security, privacy, quality, and reliability of the products that end users need. What's the use of having always-on access to data if its content can't be trusted? Poor data quality is a problem that's dogged organizations for decades. Often, the end user, such as a CFO, is the first to discover quality issues. In the traditional approach, there's no single owner accountable for guaranteeing quality.

Data Products are a fresh approach that address trust in ways that include:

- **Data contract (service level agreement or SLA):** A data contract defines the SLAs and is in-line with data. In traditional data apps, the contract may sit on a file system like

SharePoint and is out-of-band. The descriptions quickly go out of sync. A Data Product is more sophisticated as it caters not only to existing consumers but also future ones who may build derived Data Products.


TIP

Consider the ubiquitous OpenAI definitions in the REST API world: these are effectively the data contracts, and they're produced automatically by the same code that builds the APIs themselves. They can't be wrong and can't be out of date.

- **Reliability:** The Data Product owner is accountable for ensuring the reliability and quality of data across all dimensions (completeness, freshness, availability, duplication, and so on). SLAs are defined in the data contract mentioned above.
- **Security and privacy:** Data access policies apply to Data Products, and direct access to underlying raw data is restricted. Data Products extend the data governance discipline to protect both the producers and consumers in ensuring that data privacy regulatory compliance is being both maintained and audited.

Data contracts should be defined in a standard format that's machine readable — the data contract (sometimes called the Data Product manifest) from one Data Product will be used to automate many of the inputs of a consuming Data Product. Of course, human readable versions of this should be available too.

Reducing cost

This space, also known as FinOps (Financial Operations), is even more critical in the cloud due to its consumption-based pricing model. Hence, the old school of on-premises development is no longer sustainable.


REMEMBER

Data Products follow product management best practices that have been learned from software engineering. The design of Data Products to enable multiple access patterns and analytical approaches like Pandas, Spark, Duckdb, or Ray, depending on the use case, helps users to deliver optimal performance-cost-value for a given use case. Data Products offer several benefits that help in cost reduction, including:

- **Factory model:** By eliminating *ad hoc* approaches to building data outcomes, Data Products bring about standardization. They involve a data platform that includes automation, orchestration, and observability.
- **Reusability:** It's common to see thousands of dashboards and models that different teams have built for the same purpose. By design, Data Products are discoverable and reusable. They reduce the costs associated with unnecessary development.
- **Optimization:** Consumers increasingly require larger and larger amounts of data to do their work. A traditional dashboard may have only needed a megabyte of data to be useful, but retraining a machine learning (ML) model can require terabytes of data. Moving this much data around is problematic and expensive. A good Data Product will encourage moving the processing to the data using techniques like Snowflake's Snowpark.
- **Consumption metrics:** Data Products can be easily measured and inefficiencies easily removed. For example, if usage falls below the threshold or a new version is developed, the older one is retired. This reduces infrastructure and maintenance costs.

Cost containment has become an even hotter topic as organizations navigate higher costs and face increased economic pressures.

Improving performance

Businesses want to quickly respond to their requirements. A marketing department running a campaign can't afford to wait for the overburdened data engineering team to meet their needs.

Data Products liberate data. Their self-service data access helps businesses with being more responsive. Performance benefits include:

- **Collaboration:** Business/IT alignment has been a buzzword for decades but has never came to fruition. Data Products allow various producers and consumers to collaborate.

- **Release cadence:** The older approach was cumbersome, and launching new services was time consuming. Data Products (when supported by proper DataOps approaches) streamline build and deploy, allowing organizations to reduce release cycles from months to weeks or even days.
- **Data minimization:** In a project-oriented mode, unused data assets are almost never retired. They continue to occupy storage, reduce performance, increase compute costs, and increase risks. Data Products follow a more disciplined and value-based approach that eliminates these bad behaviors.

REMEMBER

Data Products make organizations far more efficient.

EXPLORING REAL-LIFE EXAMPLES

Organizations around the world are already benefiting from the Data Products approach. Here are two examples.

Roche Diagnostics

The world's largest biotech company, Roche employs 90,000 people in 100 countries. Diagnostics is a core division, focused on delivering life-changing value.

Decentralized domain-driven Data Products produced by more than 40 teams at Roche are enabling this global pharmaceutical giant to benefit from next-gen capabilities including self-service and analytics. At the time of writing, Roche has over 350 Data Products in production. One example of a Data Product is from the manufacturing domain to enable predictive maintenance of instruments that manufacture drugs on the factory floors of several global plants. This Data Product has already led to large cost savings.

This Data Product is used by the service department to create their own derived Data Product that joins the instrument telemetry data with HR and training data. This product's data consumers use the outcomes to improve productivity and service levels.

This has led to increased business agility, with teams working autonomously but within an overall framework. Internal and external demands are met faster and more effectively.

This approach has moved the business towards becoming a true data-driven enterprise. Data Product teams can adapt faster and deliver more (and higher quality) products for business end users while maintaining all the governance and security required. The company can onboard new Data Products extremely fast.

Roche benefits include:

- Rapid creation and release of Data Products — Minimum Viable Product (MVP) time reduced from six months to six to eight weeks.
- Monthly releases increased from one every three months to over 1,000.
- Robust data governance for compliance, with centralized policy definitions and enforcement — every single deploy has been through rigorous testing and peer review.
- Time savings and improved efficiency through automation (of all infrastructure) and orchestration (of over 15 different systems).
- Business discovery through automated metadata management with highly accurate and therefore usable data catalog contents.

The Head of Data Platforms at Roche Diagnostics says: "It's about speed. The quicker you can get information to the people making the decisions, the better we can fulfil our promise of giving patients what they need next."

OneWeb

OneWeb is a data-driven global satellite communications company, delivering high-speed low-latency services through a growing constellation of satellites in low Earth orbit.

Data Product teams at this pioneering satellite communications provider are driving new business value from huge data volumes across incredibly complex operations. Data relates to how the business operates, its partners, customers, and Internet of Things (IoT) telemetry data on the status and health of millions of devices 24/7.

(continued)

(continued)

OneWeb teams benefit from a single environment that governs every item of data, automates every data pipeline, and enables a high productivity culture of self-service.

With data discoverability made easy, the volumes involved are huge: The environment ingests 55 billion rows every day, and manages 1,000 trillion rows of data. With data quality critical, each data source is subject to more than 200 automated tests leading to:

- Accelerated creation and release of high-quality Data Products.
- Cross-functional analytics with improved collaboration and productivity.
- Centralized governance for all OneWeb decentralized data, and Data Products.
- Improved user experience as users are empowered to access data from multiple sources easily, complete their work, add value, and share back enriched data sets.
- Analyzing one billion records takes just one day.

The Product Owner, Digital Products at OneWeb says: "Our Self Service Data Hub empowers all users to consistently break new ground and increase business value. We've grown fast to include hundreds of developers and operators. It's easy, seamless, and transparent."

IN THIS CHAPTER

- Introducing FAIR data principles
- Identifying the top ten attributes of a Data Product
- Taking a look at a Data Product's development and ecosystem
- Considering a Data Product's definition, consumption, and usage
- Examining microservices analogies

Chapter 5
Identifying the Attributes of Data Products

This chapter takes things up a notch by looking at what makes a good Data Product, providing practical examples and identifying some of the best (and not so good) use cases. We're calling this: "Data Products done right."

At a basic level, successful Data Products are founded on the FAIR principles of **F**indable, **A**ccessible, **I**nteroperable, and **R**eusable. While these principles are perfectly legitimate, are they enough? Put another way, would it be possible to build a Data Product that is FAIR but still doesn't achieve all the business objectives? The answer is yes, which means that we need to extend these principles to be a little more pragmatic, as shown in Figure 5-1.

When considering the key characteristics of a good Data Product, you need to consider two dimensions:

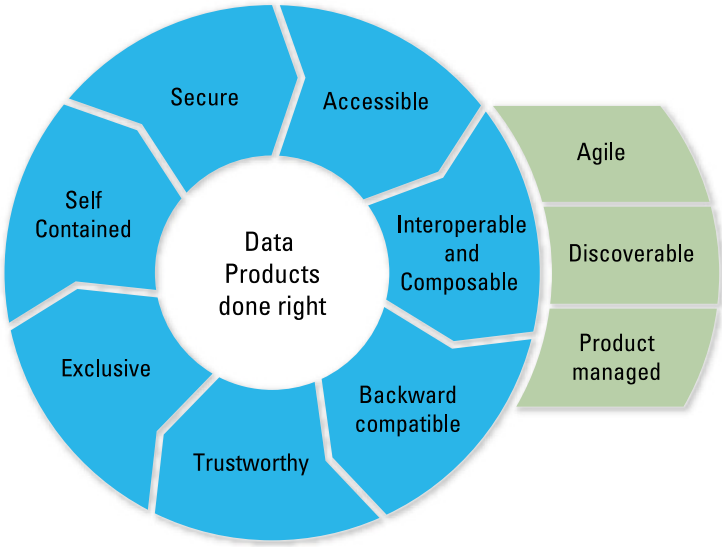

FIGURE 5-1: Data Products done right.

- First, consider the attributes of the *Data Product* itself, as it relates to its definition, consumers, and its usage (the wheel in Figure 5-1).
- Second, consider attributes relating to the development *process* and the platform/infrastructure used to develop those products and where they continue to live.

In this chapter we explore the attributes of Data Products, using microservices as a helpful real-world analogy.

Utilizing FAIR Data Principles

The *FAIR Guiding Principles for scientific data management and stewardship* were published in 2016. They provide practical guidance to improve the Findability, Accessibility, Interoperability, and Reuse of digital assets to help you arrive at the Data Products you want.

These principles extend to metadata, data, and their supporting infrastructure and are implemented through something called the FAIRification Process. This process has seven steps:

1. Retrieving non-FAIR data.
2. Analyzing the retrieved data.
3. Defining the semantic model for the dataset (how you describe and structure the data).
4. Making the data linkable.
5. Assigning license(s).
6. Defining metadata — proper metadata supports all elements of FAIR.
7. Deploying the data resource that's now findable, accessible, interoperable, and reusable.

You can find more details of principles and process at `https://www.go-fair.org/fair-principles/`.

These FAIR principles underpin the top ten attributes of a good Data Product.


TIP

Please don't think that you have to define or achieve all of these attributes on day one — just like with implementation of agile principles themselves, not all of these will be equally relevant for each organization and you don't need to worry about all of them initially. We discuss this further in Chapter 6.

Examining the attributes of development and the ecosystem

In terms of the process used to create Data Products, aligned with FAIR principles, the three key attributes are:

- Discoverable
- Product managed
- Agile

Discoverable

For a Data Product to be useful, it must be findable by its target users. This is relatively easy if you have a small team, a handful of Data Products, and a limited pool of users. But it becomes more of a challenge on an enterprise scale and must be addressed, for example, by using a searchable data catalog (such as Collibra, data.world, or Alation).

EXAMPLE

In some ways, this is where the data world is ahead of the software world. In general, the discoverability of software products (and, in particular, their data interfaces such as REST APIs) isn't systematically solved. Microservice catalogs do exist, but their use isn't as prevalent as data catalogs. This is probably because so much of the semantic knowledge and usage information about software interfaces is available using standards such as swagger/OpenAPI specification — making software interfaces, for the most part, self-documenting.

Product managed

TIP

To be successful, a Data Product must be managed and life cycled like any other digital product. A life cycle covers a lot of different areas, but fundamentally it means considering every element of data as part of a Data Product. This takes us back to our concept of *Data Products as a first-class citizen*. This includes how you plan and prioritize work, how you think about releases and versioning, even how you interact with shareholders. It also includes the critical, but often undiscussed, topic of technical debt. In the software world, identifying, tracking, measuring, and addressing technical debt is a vital element of any product. Technical debt isn't something that can always be avoided, and can't always be easily fixed, but it *can* be well managed.

Finally, while it shouldn't be a contradiction (but all too often is), Data Product life cycle decisions should be driven by data about that Data Product. A Data Product owner should be able to qualitatively and quantitively measure:

- How a Data Product is being used (for example, how many queries are being run, and by whom)
- How much value a Data Product is providing to its consumers
- How regularly and quickly value is being delivered to the business
- How much a Data Product is costing the business (both in terms of technology and human resource costs)

Only when armed with this information can a Data Product owner make explainable decisions that give the business the highest return on investment (ROI) and evolve that Data Product to deliver increasing value for the stakeholders that rely on it to make real-world decisions.

EXAMPLE

This is almost completely analogous to any software product. You will have a prioritized backlog to work through and execute and will be supported by roles such as the product owner. For more information, check out *Product Management For Dummies* (https://www.wiley.com/en-us/Product+Management+For+Dummies-p-9781119264033).

Agile

The Data Product must be structured to enable an agile and well-governed development process for all, with collaboration between developers made safe and easy.

TIP

Everything should be stored in a version-controlled source of truth, such as Git, that allows for large numbers of concurrent users. This approach allows features or enhancements to be developed and deployed independently, without conflicting with each other. In short, everything inside the Data Product needs to be set up to enable development to take place at speed and at scale, but safely and independently in self-contained sandboxes.

EXAMPLE

Pioneered by the software world, agility trumps almost everything. This is partly because if you're sufficiently agile, it almost doesn't matter how badly you get the other stuff wrong because you can fix it or improve it in minutes. (See the later "Exclusive" section.) It's also something critical to get right at the very start. In most software projects, the continuous integration/continuous deployment (CI/CD) process is usually created before, or with, the very first lines of code.

REMEMBER

The concept of building some software and adding agility through CI/CD later is broadly meaningless. In the software world, you sort the agility, the automation, the DevOps, and the CI/CD *first*, because every single line of configuration and code from that moment onwards is faster, cheaper, better, and of higher quality.

Looking at the attributes of definition, consumption, and usage

How do you define a good Data Product in terms of meeting consumer/user needs and adhering to the FAIR principles? The top seven attributes are that it must be:

- Accessible
- Exclusive

- Secure
- Self-contained
- Trustworthy
- Backwards compatible
- Interoperable and composable


REMEMBER

You're creating Data Products for the benefit of your stakeholders and consumers, not for the benefit of data engineers, and not slavishly following data architecture norms. The goal is to work together to create the Data Products that are valuable to consumers and stakeholders and that support better and more timely decision making in relation to your business goals and objectives.

Accessible

You need to publish and target your Data Products in ways that are easily accessible and useful to your consumers. If you're building a Data Product specifically for analysts to build dashboards on top of, a bunch of XML files on a disk or folder somewhere is unlikely to be a useful way to materialize your Data Product.


TIP

A data cloud such as Snowflake (`https://www.snowflake.com/en/`) or an alternative is a much more accessible platform for a much wider variety of use cases, especially with features such as data sharing.

This is equally true at a data modeling level — if your expected consumer is going to be a data analyst using a business intelligence (BI) tool, then data presented publicly using a highly normalized and complex data model is unlikely to be useful. For example, it has low accessibility for the data analyst and should be presented in a more flattened, well described, consumable view.


EXAMPLE

In the early days of software products, methods of accessing software and specific software functions were highly disparate. Over a long period, numerous different native applications have been almost exclusively migrated to web or mobile applications. Similarly, proprietary machine-to-machine interfaces were replaced with Simple Object Access Protocol (SOAP) and then REST or GraphGL interfaces. It's a recurring pattern that simpler and common denominator interfaces invariably win out, both because

of the ubiquitous skill set needed and the enormous technology ecosystem support. While there are some good theoretical arguments for custom APIs on top of Data Products, in practice SQL is both plenty good enough and totally ubiquitous. And with the key to accessibility being adoption by the existing workforce with existing skills, this is a critical point.

Exclusive

The published interface of the Data Product *must be* the only way to access the data (no back doors can exist). You reserve the right to change anything behind that interface should you wish, as long as the published interface itself is controlled, tested, and backwards compatible.

EXAMPLE

Think about a software microservice that includes a relational database, some code, and a REST API, as shown in Figure 5-2.

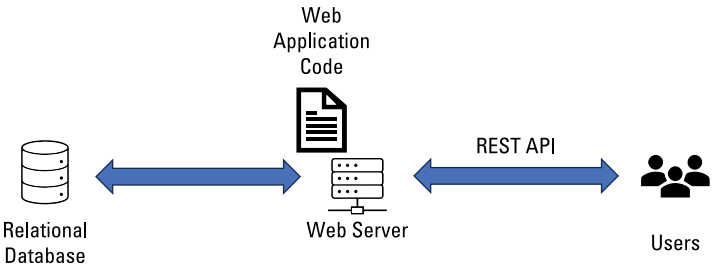

FIGURE 5-2: Simplified microservice architecture.

All any user would see is the REST API. The details of the code, database, and so on should be invisible. If a user wanted some data that was in the database but not available via the API (and the product manager agreed it as a priority), what would happen? Would the team give the user access to the underlying database? Never. That would be a terrible breach of basic product principles. Specifically, there's a supported and controlled interface for the product users, and this is the *only* and *exclusive* interface. So, the product team would add access to the missing data to the REST API. The user has to wait a few hours or days, but this is now handled in a supported, clean, and governed way and is available to future users. Also, in practical terms, you don't want to create another new supported interface, thereby creating a huge additional future support burden and technical debt.

TIP

Exclusive links closely to agile. If you have low agility in a product, you may be forced by business pressures to do it the wrong way. If you can ensure the right way is at worst nearly as fast, you can easily resist the pressure for short-term hacks that ultimately become lead weight around engineers' necks.

Returning to a critical point about exclusive, in the data world, this issue may seem less pressing. Your public and controlled interface may simply be a set of tables and views. Your internal structures may also be other tables and views. "Just give me access to that raw table," seems far more innocuous than, "Please let me bypass the API and access your database."

But it's not. The functional and non-functional problems it will create are the same. Don't let technological similarities fool you. Internal must mean internal, no ifs, buts, or maybes. You cannot bypass the exclusive way you enable access (see Figure 5-3). And if you want to enable more access, you must extend your exclusive interface or create a new one.

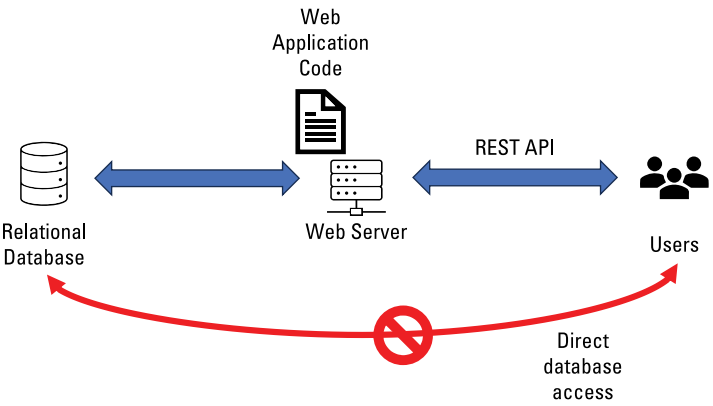

FIGURE 5-3: Never allow bypassing the published API!

Secure

Data Products must meet the access, confidentiality, and other data-specific restrictions, standards, and controls of your business. This links to the product managed attribute. Details of what secure actually means in practice will vary significantly between organizations; however, one common trap that teams not used to agile Data Product development fall into is to overthink and

overdesign the security model. They spend a lot of time thinking about what might be needed and building a complex role-based access control (RBAC) model with hundreds and thousands of minor variations to cover edge cases. This should be avoided as you can use DataOps and automation to generate these RBAC models systematically based on simple sets of rules.

REMEMBER

It's worth noting that access to the metadata about a Data Product (that is, the entry in the Data Product catalog) should generally be made available far more widely. Apart from a few special cases, it's recommended that visibility of Data Products themselves are kept as open as possible within the organization. It's rarely useful to hide, secure, and lock down the existence of the Data Product itself.

EXAMPLE

Few software parallels are needed here. Secure simply means that only suitable, authenticated, and authorized users get access to the output of the Data Product through the exclusive interfaces provided.

Self-contained

This refers to the scope of a Data Product, its focus, and inherent value. As a general rule, a Data Product should be as small as it can be, but no smaller, while being designed from the perspective of a user and what they need. A key test is that a Data Product shouldn't require other Data Products to provide the expected value for key uses cases of that Data Product. Clearly this doesn't mean that every question is answerable with just one Data Product, but if you find that half your users are having to join your Data Product with another one to answer relatively simple questions, then this is a red flag that the scope of your Data Product is too small.

A simple example of this problem is a "Sales" Data Product that exposes a "Product ID" column. However, in order to find out anything about this product, such as its name, type, or category, it requires joining on to the "Product" Data Product. If most of your users are having to do this joining themselves every time, that's a sign of an issue. In this example, it may still be appropriate to have isolated or standalone Data Products for Sales and Product, but a third, composite Data Product should exist that provides a joined, flattened view of these optimized for analytical use cases.

EXAMPLE

In many ways, "self-contained" is the hardest attribute to quantify, and one that even the software world struggles with. Let's look at two extremes: a single software product for an entire business or function — the monolithic software applications of old — versus a microservice to simply serve up an avatar picture for a user.

Ultra-fine granularity means that even a simple software-as-a-service (SaaS) application could end up being hundreds of microservices. While the overhead (with correct automation, CI/CD, and so on) should be low, it's not zero, and so probably too many. In practice, if you're displaying the user's avatar, you're probably also displaying their username, last log-in date, and perhaps recent activity. If these common use cases require you to visit four or five separate microservices, they aren't really valuable on their own.

So, you work left: You start with your best guesses about the use cases and how this data is likely to be used, and you try to craft the scope of your software and Data Products to be reasonably narrow, but ensure that each is valuable on its own. You won't get this right every time, because you're trying to make decisions with imperfect data. Don't beat yourself up: That's life, and most likely your scope will end up being good enough. If it turns out fundamentally wrong and causing huge problems, then it's entirely acceptable to end of life one Data Product and release two or three to replace it (or merge two or three into one). Software teams do this regularly.

TIP

Normalize this and make it acceptable to consumers by giving them an overlapping transition period. The worst mistake would be to become paralyzed due to lack of perfect information and make no decision.

Trustworthy

As important as data quality, the Data Product must provide commitments to consumers around completeness (including associated of levels of automated testing), accuracy, and timeliness.

EXAMPLE

It's an interesting characteristic that, in general, users are inherently trusting of the data they get from software products. This is likely because data interfaces for software products are often directly on top of the operational/transactional source of truth. The goal for the data world should be to establish the same level of assumed trust from users of Data Products.

Backwards compatible

A Data Product must be backwards compatible, with that compatibility proven by automated testing. Ideally this testing should be so accessible to a developer that they're notified of a break in backwards compatibility even before committing a change, giving them an opportunity to solve their need in a different, no breaking way.

TECHNICAL STUFF

This does raise an interesting question about what constitutes a breaking change of a Data Product. There are clearly some obvious ones, such as dropping a table or renaming a column. And there are some which are definitely fine, such as adding a column. While some people would say that this breaks backwards compatibility since the output for someone doing the equivalent of "select *" will change, this isn't the case. Whether explicit or not, a Data Product producer should be able to assume that a consumer takes basic tests to protect themselves, such as including columns in a select.

A data contract is a two-way street. The producer commits to certain things, but the consumer also commits to following good practice and basic precautions. There are, however, some gray areas. For example, does removing a "not null" test on a column constitute a breaking change? This is debatable, but we'd argue that if this test was published as part of the Data Product manifest (the "official" representation of a Data Product), then a consumer could reasonably build their downstream application (in whatever form that was) assuming that this field/column was not null. If nulls start appearing, their application may well break and therefore this could reasonably be regarded as a breaking change. Whatever your stance on this sort of thing, it's clear that offering/agreeing backwards compatibility only really adds any value if it also details a bit about what it actually means.

Only assume a commitment to backwards compatibility once the product reaches the required level of maturity and is released in general availability (GA). A common pitfall for data teams is releasing too early and making a commitment against something that's not fit for purpose and needs structural changes quickly.

TIP

Don't release your Data Product too early while you're still evaluating and finessing it. While it's still in a pre-release state, you can reserve the right to change it until it reaches a point of maturity. That's the point to commit to backwards compatibility.

The reality is that no matter how much care is taken, there will always be a time when a structural change is needed, either for strong internal technical reasons or good business reasons. At this point you have no choice but to make breaking changes. But how do you do that?

The answer was solved in the software world a long time ago and is now universally accepted: You version your Data Product interfaces (for example v1, v2) and allow them to co-exist. When you need to make a breaking change, you do it as part of a new v2, and once this is stable and released, you give your consumers a period of time to migrate from v1 to v2 before you deprecate v1. Of course, this requires work from consumers and shouldn't be done lightly by the producer, but it gives the consumers a chance to make the necessary changes in a planned and orderly fashion.

EXAMPLE

The microservice analogy for this hardly needs stating — software interfaces, especially REST APIs, have been almost universally versioned since their inception, specifically to solve this problem.

Interoperable and composable

Do you have datasets that can physically be used together? Interoperability is key: There must be an element of agreement and standardization to make this happen, whether that comes from a top-down mandated approach or bottom-up committee-led collaboration. People often mistake self-service or domain-driven for "teams can do whatever they want."

WARNING

In fact, the opposite is true. The more decentralized teams building Data Products are, the greater the need for teams to coordinate with each other to ensure that, at an enterprise level, everything is still compatible and interoperable.

A critical extension to this is the concept of composable Data Products. You can broadly split Data Products into two types, foundational and composite, as shown in Figure 5-4.

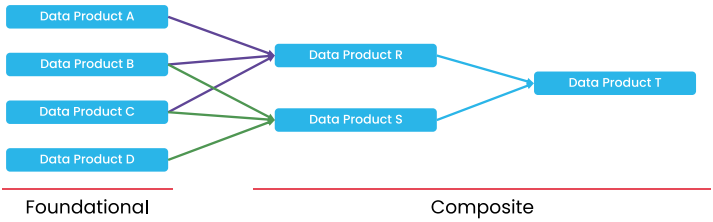

FIGURE 5-4: Composable Data Products: foundational and composite.

A foundational (or raw) Data Product is what most people think of first. This is a Data Product where the input is external data (in files, a relational database, behind an API or any number of other data sources), the data is ingested/loaded, curated, or mildly transformed in some way, and then made available for downstream consumers. Typically, when considering foundational Data Products, the downstream consumer is an analytical consumer producing BI reports/dashboards or perhaps a data scientist. These are valid (and vital) consumers.

EXAMPLE

However, other types of Data Product consumers are also essential. One example is creating a Data Product using information that's already within the platform and either publishing it in some fundamentally different way, such as highly aggregated and summarized. Another example is creating a Data Product using existing information and mixing that with data from other Data Products, such as joining sales, customer, and marketing data to create a 360-degree customer view.

This is the concept of a composable Data Product; a Data Product where it's expected that the output of one upstream Data Product will be used as one of the inputs to a downstream Data Product. This simple step unlocks an entire additional layer of business value and is a widely used approach by Data Products pioneers.

EXAMPLE

In the software world, composability is taken for granted. Microservices expect to be able to use other microservices as inputs and fully expect their outputs to be consumed by yet more microservices. This is mostly straightforward because microservices interfaces are so standardized around REST APIs, and over time a large ecosystem of libraries, tools, and, most importantly, expertise, have grown up around this. However, the area of interoperability is still an area where the software world sometimes struggles. Without proper coordination it's still customary for common concepts to be represented differently, causing challenges for consumers. Perhaps it's comforting to know that even an industry that has been doing this for decades still struggles . . . or perhaps not!

Dealing with multiple interfaces

An eagle-eyed reader may spot a problem at this point. Consider the case of a Data Product that has two target consumers:

- Human/analytical users
- Data Product engineers creating new Data Products

And let's assume that this Data Product is using a specific data modeling technique like Data Vault. If you publish your Data Product interface as in native Data Vault format, downstream Data Product engineers are very happy. They can use this directly with other data in the same form.

However, a Data Vault isn't directly usable at all by human users. The solution is to flatten this out and denormalize it into some sort of mart. Human users are now happy, but the data engineers will now have to reconstruct their Data Vault from this. This solution is hardly the efficiency you're aiming for!

So which option do you choose? Both!

It's a misnomer (albeit a very understandable one) that a Data Product must only have one published interface. Yes, the interfaces must be accessible, secure, and exclusive (that is, not bypassed), but nothing says they must be unique. In the scenario above, the accessibility principle dictates that your published interfaces need to be accessible for your consumers. Two highly different types of consumers require two parts to the published interface — in this case one flattened for analytics and one Data Vault for other data engineers.

If this makes you twitch a little bit, you're not alone. We've spent years, even decades, striving for a single source of truth and all of a sudden we're allowing two?!

Don't worry — these aren't two sources of truth! There *must* be a single, normative set of the data which *is* the source of truth. However, it's then okay to have multiple ways or facades through which to present this.

EXAMPLE

It may seem a little harder to find a good microservice analogy to this concept of multiple interfaces to the same data. However, consider the following two cases.

If data is usually available via a REST API, but you need some of it in real time, do you expect the REST API to be able to do this? No — you'd use something specifically for this, such as WebSockets. Does this mean you remove your REST API? Again, no. You have different use cases and these require different interfaces onto still the same fundamental set of data.

Consider that you want to download all of your historical messages and content from a social media platform. You could download each, one at a time, via the REST API, but this would be very inconvenient, with certainly low accessibility. Instead, most social media providers give the option to "download all history as a ZIP file." Again, totally different consumer use case, same data, different interface.

IN THIS CHAPTER

- Exploring Data Product roles and responsibilities
- Introducing the Data Product Maturity Model
- Taking a look at the stages to build and deploy Data Products

Chapter 6
Building and Deploying Data Products

This chapter provides a high-level guide to building your first Data Product and setting up a process to create a factory model to repeatedly build more.

Building Data Products should be fun. After all, you have a concrete business goal and accountability for your deliverables that can be measured by their impact on the business. And Data Products have a life cycle that goes from design to retirement. This contrasts with a traditional project-oriented approach that lacks a sense of ownership and is temporary in nature.

The goal of the Data Product journey is having an outcome that directly benefits the organization in a way that previously was cumbersome and incomplete.

Understanding the Roles Involved

Creating Data Products rapidly is a team effort. The first job is to understand which roles are involved in designing, building, and deploying Data Products. And some of the roles are familiar from typical data and analytics tasks, as shown in Figure 6-1. The diagram shows the four key roles, which are often given different titles.

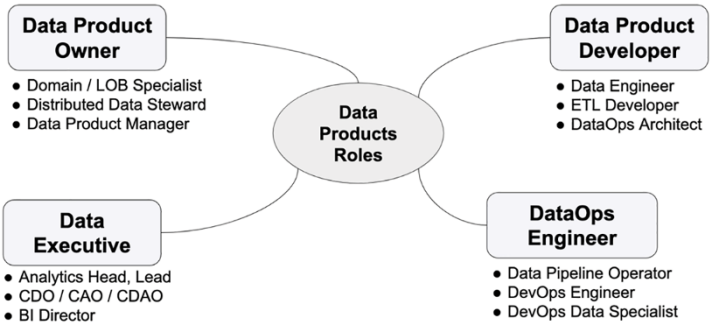

FIGURE 6-1: Roles involved in building and deploying Data Products.

Table 6-1 explains the responsibilities attached to these roles.

TABLE 6-1 Key Roles in the Data Products Lifecycle

Role	Accountability	Responsibilities
Data Product Owner	The "glue" between data producers and consumers Success of Data Products through its lifecycle	Data consumer requirements, expectations SLAs, quality, access Ensure regulatory compliance Refresh cycles, change requests Measure usage
Data Product Developer	Develop and maintain Data Products	Set up sandbox and data pipelines Accelerate with Data Product solution templates Develop business logic and code with reusability Automated testing

Role	Accountability	Responsibilities
DataOps Engineer	Setup environment and infrastructure to build, test, and deploy	Force multiplier for Data Product engineers/teams Automate CI/CD in declarative manner Data Products catalog Monitor and audit
Data Executive	Demonstrate business value and developer productivity	Track regulatory compliance Understand usage Optimize total cost of ownership (TCO)

Looking at the Data Product Maturity Model

It's crucial to understand your current state before embarking on the Data Products journey. This provides a baseline and identifies your level of maturity.

REMEMBER

The maturity model should cover all three legs of the stool:

- **People:** Data Products require a different mindset that applies product management focus to data. They need all stakeholders to agree on this new paradigm.
- **Process:** Data Products follow agile principles that are very different from waterfall approaches of the past.
- **Technology:** The architecture for monolithic software deliverables may no longer be suited to the rapid iteration necessary to build and deploy Data Products.

The maturity model consists of emerging, evolving, and mature phases. However, Data Products are a relatively new concept, so most organizations are in the emerging or evolving space.

Exploring the emerging phase

You're in the emerging phase if you're yet to build a Data Product. Congratulations, as you're in the majority! And this is probably why you're reading this book.


TIP

Your first task is to define domains. In this stage of maturity, your domains will match the lines of business. As you're new to this space, your primary mission is experimentation. Don't be afraid of trying and failing fast. This is the fastest way to move up the maturity curve!

You may not need dedicated roles at this stage. Build a few Data Products (following the steps mentioned below) then monitor their benefits.


WARNING

Newbies will often decentralize their central data team and move it to domains but not change any other aspects of their data outcomes. They'll then declare the outcomes as Data Products. But decentralization alone doesn't make a Data Product. You still need to assign ownership and change practices to your product management approach.

Entering into the evolving phase

Once you have a good grasp of foundational Data Products, it's time to instill all the steps necessary to scale. At the evolving stage, ensure that Data Product owners and other roles are established.


TIP

Before you go to production with your first set of Data Products, revisit security, governance, and DataOps practices. Moving from centralized to domain-driven teams requires a higher level of engagement across teams.

Assess if your current infrastructure is suitable to deploy your Data Products. Many organizations have discovered that they can adapt their current state while others have had to procure new platforms. One example of where the current infrastructure may not be suitable is automation of security policies across domains.


REMEMBER

At this phase, you should have a centralized governance structure with federated decision-making process. Set up integration with other systems in the data and analytics ecosystem, such as data observability, DataOps, Data Products catalog, and data marketplace.

To move up the maturity curve, ensure that data quality has been addressed and data consumers have clear visibility into the veracity and provenance of data elements.

Reaching the mature phase

Congratulations if you have mastered Data Product production and deployment at scale! Few companies have achieved this level — yet — and they have already gained a substantial competitive advantage. They generally have a center of excellence to propagate the benefits of Data Products across all lines of business.


REMEMBER

Mature organizations have data sharing agreements with partners who consume the Data Products. They have strong access control procedures and monitoring to protect sensitive data and meet relevant compliance guidelines.

So, what next? Organizations at this mature stage should be looking to enhance their Data Products by incorporating advanced technologies. For example, explore adding natural language interface through large language models (LLMs).

Preparing to Build and Deploy

In the past, after gathering business requirements, you'd jump straight into the build phase.

Data Products need a more disciplined approach that, at first, can look daunting. But this is necessary for fast iteration using a factory model that enables repeatability and rapid deployment.


REMEMBER

Data Products are owned by domain leaders, and hence, are a business deliverable. The build process starts with business teams defining their outcomes. Next, the infrastructure team builds the mold for repeated Data Product production. Finally, the technical team codes and deploys the Data Products.

Figure 6-2 shows the steps for building mature Data Products.

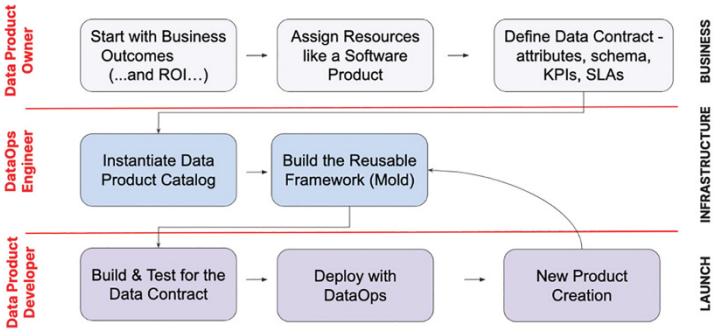

FIGURE 6-2: The steps to build and deploy Data Products.

Getting started in the business stage

Data Products are value-centric and solve well-defined business domain problems. Domain leads identify the right products to build and to build them right. A domain lead could be head of the line of business. He or she will assign a Data Product owner (DPO).

Before Data Products are built, the DPO is responsible for:

- Estimating how much value will be gained and getting funded. The DPO needs to communicate the value to all stakeholders and gain their buy-in.
- Establishing the team for the entire lifecycle. Remember, unlike a project, the resources required will last for the entire duration of the product. This team will depend upon the needs of the Data Product. For example, one team may have the DPO, a business analyst, and a data engineer, while another may have a data scientist.
- Documentation! Data Products, unlike software products, don't always have a user interface. So, all the attributes of a Data Product need to be documented and made shareable as a data contract. The DPO must clearly define business goals, inputs and outputs, KPIs, SLAs, schema, and the attributes defined in Chapter 5.

TIP

Collaboration is one of the keys to success. The DPO should share the vision across business and technical teams and always try to avoid technical jargon.

Moving into the infrastructure stage

A platform-centric technical approach enables the factory model. This provides two essential ingredients: data governance and DataOps.

The DataOps engineer is responsible for:

- Establishing all the infrastructure used to build and deploy Data Products. This includes a Data Products catalog that has the data contract and metadata relating to usage.
- Setting up the DataOps foundation, which typically includes orchestration, automation, continuous testing, and environment setup capabilities.
- Educating the domain teams to easily use the infrastructure so that they can focus on the business aspects and not get distracted by technical minutia.
- Simplifying and generalizing the technical capabilities so that they can scale as organizations move up the maturity curve and add more complex use cases.

TIP

Standardize on a data platform that reduces data movement and avoids creating data silos, such as Snowflake, Databricks, or Google BigQuery. And choose a DataOps platform that can act as a central nervous system for the entire lifecycle, such as DataOps.live.

Reaching the launch stage

Congratulations! You've finally reached the all-too-crucial development coding stage where the business logic to meet the functional requirement is developed, tested, and deployed.

TIP

In general, there shouldn't be much difference between the traditional approach except for applying DataOps best practices of version control, continuous testing, and automated deployment. Applying these practices helps alleviate the chances of accumulating technical debt.

The Data Product developer works with the extended IT teams and is accountable for:

- Developing the business logic to meet the functional goals. They still use the corporate standard languages which may be SQL, Python, Java, and so on. With the rise of generative AI, many organizations are exploring copilots to assist in writing code.
- Testing the code. This also follows well-known standards, except in Data Products, you test against the commitments made inside data contracts. The data contract acts as the documentation of the requirements and service level agreements (SLAs). Your DataOps setup should help with best practices, such as version control, branch/merge, and continuous testing.
- Deploying using the DataOps best practices of continuous integration and continuous deployment (CI/CD). This helps in improving the release cycle timing of Data Products.

Once you've completed the three stages and deployed the Data Product in production, your work isn't complete. Be ready for understanding the usage of the product and consumer feedback. Data Products is a journey consisting of iterative new versions. Once the consumers start using the Data Product, new requirements will emerge.

IN THIS CHAPTER

- Understanding the need for an ecosystem
- Looking at the components of an ecosystem
- Improving the effectiveness of Data Products

Chapter 7
Exploring the Data Products Ecosystem

In this chapter, we look at the place of Data Products in the vibrant data and analytics ecosystem. The bi-directional interplay between various components of the ecosystem is needed to deliver Data Products quickly, reliably, and at scale.

The importance of the ecosystem arises from various corners. One of them is that most organizations are dealing with a rapid expansion of their data usage. Businesses have realized the competitive advantage of their data assets and are hungry to mine for insights. This higher expectation from data requires a well-structured ecosystem.

The second reason is that organizations are moving to a mix of centralized and decentralized approaches. This increases the risk of issues falling through the cracks; hence the need for a well-integrated ecosystem.

Introducing Ecosystem Components

In this section, we look at the various major building blocks that interact with Data Products, as shown in Figure 7-1.

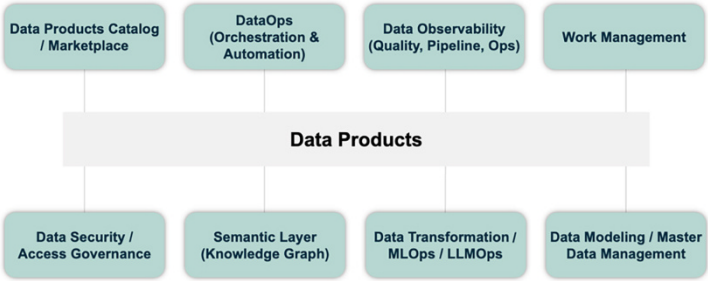

FIGURE 7-1: The building blocks (components) of the logical Data Products ecosystem.

The ecosystem has been labelled "logical" because in the fast-paced technology landscape, some of the capabilities may be embedded in a single product. For example, it's common to see modern iterations of DataOps products embed data observability features.

TIP

A well-integrated ecosystem will help you achieve the critical goals of Data Products: domain-centric agility and autonomy. It will help you deliver Data Products at scale.

When it comes to the ecosystem, your mileage will vary. Depending upon your level of maturity (see Chapter 6) and your requirements, you may need some or most of the components of the ecosystem shown in Figure 7-1.

Let's examine each of the components.

Starting with Data Products catalog / marketplace

A Data Products catalog is a repository of Data Products and its associated metadata. It's also called a Data Product registry. It tracks the entire life cycle of Data Products creation. Essentially, this catalog houses both the build and the operations of Data Products. It's a single pane of glass to discover Data Products and their associated metadata. A data consumer may use it for:

- **Certification:** Identify which Data Product is certified and gold-star-rated, and learn its quality aspects in order to decide which Data Product to use. For example, if there are 15 Data Products named "quarterly sales," data consumers

can determine which one is approved so everyone is working with the same numbers. At the same time, data managers can remove, rename, or annotate the other 14 instances.

- **Usage:** Understand the usage of Data Products and discover "data tribes." This serves the two purposes of telling the user which Data Product is appropriate and which unused Data Products to retire. Automation will mark Data Products stale based on pattern and usage analysis and thus reduce data debt.
- **Efficiency:** Explore operational metadata for Data Products, such as security access rights, data creators, version numbers, purpose, and user consent.
- **Productivity:** Calculate productivity via "data telemetry," such as the frequency of releases, number of data-related goals and objectives met, the level of buy-in, and support for the data strategy within the organization.
- **Innovation:** Enhance data-driven innovations such as identifying new Data Products that could deliver new products or services.

REMEMBER

A Data Products catalog serves as the marketplace of Data Products for internal and/or external users. Some organizations are already monetizing Data Products, so the existence of a data marketplace has become essential. A key distinction between a catalog and a marketplace is that a catalog focuses on providing information and discovery (potentially to a greater depth), whereas a marketplace includes functionality around fulfilment. For example, if you see a Data Product you like, you can take action directly within the marketplace to request access.

In some organizations, the data marketplace may sit outside the realm of a Data Products catalog.

Exploring DataOps

Successful deployment of Data Products is inextricably linked to the adoption of DataOps practices that govern the life cycle of building and operations. DataOps practices are used by developers and maintainers of Data Products. DataOps can help to increase the velocity of delivering Data Products by automating the data pipeline tasks.

Some of the DataOps capabilities include:

- **Automation:** The ability to automate most tasks involved with developing and deploying Data Products including infrastructure automation, environment creation, and branch management.
- **Orchestration:** Orchestrating all of the tools in the end-to-end ecosystem of tasks in a data pipeline, ensuring that tasks are executed in the correct order and that dependencies are met. This increases reliability and reduces risks or errors associated with manual tasks.
- **Continuous testing:** Applying the "shift left" principles of verifying that the data pipeline is working correctly ensures that the data is accurate and that the Data Products are performing as expected.
- **CI/CD:** Continuously integrating and continuously delivering code ensures that new changes to the data pipeline are deployed quickly and reliably.
- **Version management:** Maintaining the source of truth and all the different versions/branches of this including merging, cherry picking, and reverting. This is almost always based on Git.

Including data observability

Observability came into prominence as an evolution of the work that the application performance monitoring (APM) vendors, such as Splunk, Datadog, and New Relic, had started. Their focus is on performance and reliability of the infrastructure. Data observability takes the lessons learned from APM and infrastructure observability and applies them to data.


REMEMBER

Data observability diagnoses the internal state of the entire data value chain by observing its outputs with the goal of proactively rectifying issues.

Data observability serves three use cases:

- **Reliability:** Provides Data Product developers with the ability to trace the path of that failure, to walk upstream and determine what's really broken at processing, computation, or infrastructure and quality level.

- **Data quality monitoring:** It monitors and tracks unexplained changes in the shape of the data from expected norms. This is the most common use of data observability tools as it helps with establishing trust in data.
- **FinOps:** It tracks costs incurred when the Data Products are used with the goal of optimizing the overall spending. It provides a granular view into how costs are being incurred, ideally before they impact budgets and forecasts.

Using the data observability capabilities of monitoring, alerts, and notifications, proactive action can be taken to ensure that data pipelines underneath Data Products are running efficiently and cost effectively.

Taking a look at work management

In order to manage and prioritize the backlog around a Data Product, a work management system is required. This allows the creation and decomposition of tasks, estimation, and prioritization, as well as including the business 6 for work.

EXAMPLE

The work management system should also integrate with the DataOps tool. For example, when you're ready to work on a work item ticket, with a single click you should be able to have a branch created in the DataOps system and have a development environment automatically created. Similarly, all commits and merge requests created in the DataOps system should be immediately visible in the work management system.

Ensuring data security and access governance

Data consumers should have consistent access to all decentralized Data Products in different domains through centralized policy management and a universal authorization layer. This ensures interoperability of data and business context in a self-service manner that maximizes its usability.

TIP

Role based access control (RBAC) with attribute-based access control (ABAC) should be used to reduce the complexity of access control and to apply consistent access policies at scale. User identities are tied to roles, and roles are tied to policies. The policies are further attached to the underlying data elements, their attributes, and even user attributes, such as the data consumer's geographical location or job title.

REMEMBER

Data Products use RBAC and ABAC dynamically. As Data Products may be used by data consumers not yet identified, it isn't always possible to determine security policies. As a result, the usage metrics and metadata-based approach may be the right choice for many. This ecosystem component is also used to ensure relevant data privacy regulatory compliance requirements are met.

Defining semantic layer / knowledge graph

A semantic layer is a layer of abstraction that sits between the data and the business users. A semantic model is a formal representation of the meaning of the data. It can be used to define the relationships between different entities in the data, and it can be used to create a common vocabulary for understanding the data.

Semantic layers make Data Products more user-friendly and easier to understand. They can also help to ensure that Data Products are consistent with each other, which is essential in a decentralized environment.

Collaborating with data transformation and machine learning integration

Data engineering tasks in the Data Products ecosystem need a higher level of collaboration than in the past when all team members belonged to a centralized team. In the new approach, each team owns its data and the model. They transform the data as per the needs of their own Data Products. However, often some of the data may come from a different domain.

Therefore, the data transformation tool needs to be closely integrated with the rest of the Data Products. To engender agility, the data transformation capabilities may be built into other components of the ecosystem.

Similarly, in the Data Products that require artificial intelligence (AI) or machine learning (ML), fully fledged standalone MLOps products may be overkill compared to embedded options.

TIP

These products also serve another purpose — metadata extraction. The metadata is used to build end-to-end lineage, which is exposed in the Data Products catalog. The lineage is used to speed up root cause analysis in case of errors and impact

analysis to determine effects of upstream changes into downstream consumers.

Relying on data modeling / master data management

Data modeling turned into a lost art when the era of big data and schema-on-read arrived. Prior to that, data modeling was exactly how business needs were encapsulated into the design of relational database management systems. A data model preceded any technical development of the desired solution.

REMEMBER

In the Data Products world, data modeling is especially crucial because domains need to talk to each other and need a common language so that their Data Products can be used across the organization.

An enterprise data model may lead to one or more Data Products. In other words, the data model allows different domains to collaborate and interoperate during the design and build phases. It helps business and IT teams align on the goals of Data Products.

However, rediscovering our data modeling doesn't mean a wholesale return to the upfront, monolithic data modeling (analogous to the old software development approaches of producing an Entity Relationship Diagram (ERD) before we write a single line of code). In an agile Data Product world, modeling also becomes an agile process — you iterate your data model as you make changes. The data model becomes automatically documented as part of your Data Product build pipeline.

TIP

There are many data modeling approaches available including star schema, Data Vault, Kimbell form, and many others. There's no strong alignment of any of these to Data Products. Your team can use whatever best suits your skills and data.

Data models should be a part of the Data Products documentation for the data consumers to be aware of.

Master data management (MDM) products are often used to link common terms and definitions across domains so that they speak a common language.

IN THIS CHAPTER

- Getting started
- Knowing what to think about now and later
- Putting in work up front to prevent trouble in the future

Chapter 8
Ten Ways to Start Building Data Products

Data Products are the future for how organizations will organize, develop, and deploy data assets and data applications. They keep everything that works well about how data has operated up to now, but add in the benefits of decades of innovation from the software world. Some organizations are already on this journey, but for many, while the value and benefits are impossible to ignore, getting started can be daunting. In this chapter we provide you with a blueprint for success.

Think Differently

The value of thinking differently surely transcends Data Products but is definitely worth discussing for our purposes!

Building Data Products requires a different sort of thinking than traditional data projects. However, this isn't the only place where success is fundamentally dependent on new ways of thinking. The move from on-prem/legacy data warehouse platforms to

cloud native has always had enormous potential benefits, and yet some organizations failed to realize these benefits. Why? They approached new technology and paradigms with old school thinking.

The same issue occurred when organizations tried to move on-prem servers and compute processes to the cloud. They were promised huge gains in scalability, resiliency, performance, and cost. But some organizations just tried to lift and shift their servers without any of the required refactoring and were surprised when the benefits didn't materialize. Leaps forward that promise great benefits can usually deliver them, but only when accompanied by the new thought processes required to support them.


TIP

This change in thought process can't be restricted to just you: it must be shared by all the key people involved. Nothing kills progress like different people with different thought processes.

Pass Out the Hats!

As discussed in Chapter 6, some key roles are required to make this work. It's important to understand that roles don't mean people; they simply mean work to be done. When you're starting out, it's far more important to understand the functions to be performed and ensure that someone is going to do them than to worry about having dedicated people. It's entirely possible to start with two people, one wearing the Executive and Product Owner hat and one wearing the DataOps Developer and Data Products Developer hat.

Choose Your Battles

There are many principles for the development of software products, particularly microservices. But does every microservice follow every one of them from the first day of development? No! There are some fairly immutable principles like DevOps and Security, but many of the others are flexible. It doesn't mean that they're unimportant, but not every principle is equally important and not every successful software product will tick every box.

TIP

It's better to think about these more like Agile principles — you need to know WHAT they do, you need to know WHY they're important, and you need to know WHEN they deliver value. Once you know all of this, you then decide how little or how much of those principles make sense for you. Consider Data Product principles in much the same way — understand them all and then make a decision on which ones are most important to your organization and start with those.

WARNING

It's very important to differentiate between "what is easiest" and "what delivers the most value." We all fall into the trap of gravitating to the path of least resistance, but challenge yourself to focus on best value not least work. But don't neglect the opportunity for a few quick wins either!

Start With the Data You Have

As you've already seen, especially if you've already started to move workloads to a modern data platform, many aspects of a Data Product do look a lot like what came before. If you have a data project or data asset that's in broadly good shape, then retrofitting Data Product principles can be a good way to move forward.

However, if you haven't started on this process, there are huge advantages to starting out with a subset of the Data Product principles and attributes and growing from there.

Know What You DON'T Need

If you look back to the ten attributes discussed in Chapter 5, there's a lot there. And from the "Choose Your Battles" section above, it's clear that you may not need some elements for a long while, maybe ever. But the even bigger challenge is just how to get started. So, let's focus on what you *don't* need on day one.

If you only have two or three Data Products, discoverability is unlikely to be a big issue. You need to have something, but on day one a simple shared page may be enough.

Interoperability and composability are unlikely to need a big focus with a small number of Data Products.

Backwards compatibility is also a low priority for day one. You have barely built the first version so why would you need to think about the second version? With that said, take a look later in the chapter at the section titled, "Recognize Where a Little Bit of Thought Will Pay Dividends."

Being trustworthy is of course important, but think critically about *how* much of this you need to get started. It may be less than you think! Very much the same is true for security.

WARNING

Many organizations have become hung up on making their Data Products self-contained, spending far too many cycles designing and redesigning, trying to get them perfect. Perfect is often the enemy of good, and almost always the enemy of quick! Apply a sensible amount of thinking and then just get something built. Getting real feedback from your stakeholders on scope is far more valuable than endless design cycles anyway!

Know What You DO Need

There are, however, some pieces that you should never take shortcuts on.

Before you start anything, you need to capture, prioritize, and manage the work. Therefore, a reasonable level of product management is key along with a basic work management system.

In reality, you're going to do more work on your Data Product in the earliest stages of its life. Adopting Agile approaches and a DataOps system that supports them will make this development much more efficient, effective, and better managed. To implement these *after* you've done the bulk of the work makes no sense at all. It would be like putting down the protective sheets when 90 percent of the room has been painted — the right idea, but much too late!

Exclusivity is also key from day one, not so much because of any strong technical reason, but if you don't start off setting the correct expectations and practices with your users, you will never rein them back in! Start out as you mean to go on.

Recognize Where a Little Bit of Thought Will Pay Dividends

At times, just a little bit of thought today will create HUGE value in the future. The challenge is of course knowing where! Here we provide a few points that some of the pioneers wish they'd thought of on day one!

Firstly, which of these lists makes more sense?

- DATA_PRODUCT
- DATA_PRODUCT_V2
- DATA_PRODUCT_V3

or

- DATA_PRODUCT_V1
- DATA_PRODUCT_V2
- DATA_PRODUCT_V3

The second is far more logical, has no special cases, and doesn't require special coding to differentiate V1 from the others. All this is needs is a small amount of thought to include a version notation somewhere in the Data Product deployment into the data platform and hard code this to V1. You can then ignore this for a year, maybe even two, but you'll be very glad you did when you come to need a V2.

REMEMBER

Some things are much more painful to retrofit than others, such as not starting with DataOps and Agile from the beginning, for example. But within the modeling of your Data Product, the structure of your primary identifiers/keys is much harder to tweak than many other things. The main situation where you might need to do this is around interoperability. So, take a little time, identify the things that are most likely to be needed to create interoperability between Data Products, and just write down how they should be handled.

Remember that Nothing Speaks Like Success

In reality, few organizations are going to make a full and wholesale move to a Data Products approach without some piloting and validation. If you turn up to a management and business review and present two to three Data Products that:

- Are automatically built and rebuilt
- Have a named Data Product owner with well-defined responsibilities
- Have agile and efficient development processes
- Give consumers excellent visibility into the Data Products, live
- Have transparent and prioritized roadmaps
- Have a clear model for future development and versioning
- Make clear where they are and what their maturity journey looks like for the future

. . . it will be a rare business indeed that doesn't buy into this journey with you!

Start your journey, show your initial successes, and use this as a launchpad to bigger and better things!

Create a Value Model

The business benefits of a Data Products approach are discussed in a number of places in this book. But if you want to make a case to the business or to management, you need to be a bit more quantitative. Obviously, the exact details will differ significantly between different organizations. However, what you can do is define some areas and a set of questions, including:

- How long does it currently take to create a minimum viable Data Product you can put in the hands of your stakeholders? What if you could do this in four weeks? What value would that create?

- How many releases are you doing for each data asset/product per month/quarter? What if you could do several per week?
- How much would it cost the business if backwards compatibility broke? What if this was automatically checked and prevented?
- What is the overhead for creating development/test environments (or the overhead of trying to share an environment) for each feature? What if this was created automatically and at virtually no cost?
- What is the cost of having old/stale information in your data catalog? What if this was automatically updated every time a Data Product was updated?
- What is the cost of business users not having trust in the data? What would it be worth if business users not only trusted the data but had direct visibility into the Data Products themselves?
- What is the cost to the business of having to coordinate breaking changes across many people and teams simultaneously? What would it be worth if a Data Product could be fundamentally changed and consumers had months to convert to the new approach?
- How long does it take to get an urgent change into production in a safe and well governed way? What would it be worth to the business if this could be done in less than an hour?

Against each of these questions, you should be able to make some assumptions, document them, and from these calculate some actual, quantified business value.

Consider What the Future Looks Like

Let's be clear — the world of Data Products is simultaneously enormously valuable and here to stay, but on the other hand it's still relatively new in its modern incarnation. Consider that before 2023, virtually no one had even heard of large language models

and now they're everywhere! Can we really, confidently, say that we can forecast what the future for Data Products will look like in one or two years? No, of course not! However, what we can say with a high degree of confidence is that it's very unlikely that something new will come up that can't fit well into the model and approach that we outline in this book.

REMEMBER

The future isn't certain, but the value of a flexible and future-proof approach is!

Unleash the power of Data Products

Data is a victim of its own success — it's such a critical ingredient for any organization that data teams face ever-tighter deadlines to deliver the goods. The traditional way of creating data applications isn't scalable and data teams are becoming a bottleneck. The older approach leads to data outcomes that lack accountability and data pipelines that aren't reliable.

Data Products are the future for how organizations will organize, develop, and deploy data assets and data applications. But current information about Data Products can be complex and confusing. *Data Products For Dummies* is your one-stop guide to moving beyond the theory to embark on a pragmatic and practical journey towards building Data Products for your organization. Using relatable examples throughout, this book demystifies Data Products and sets you on your way to unlocking this powerful approach for developing modern applications.

Inside...

- Understand the relationship between Data Products and data as a product
- Discover where you are on the Data Product maturity model
- Improve time to value, reliability, security, and privacy
- Deliver a superior customer experience with Data Products
- Lower costs by reducing wasted resources
- Be inspired by real-life examples

Dataops.live

ISBN: 978-1-394-21555-3

Not For Resale

for dummies
A Wiley Brand

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.