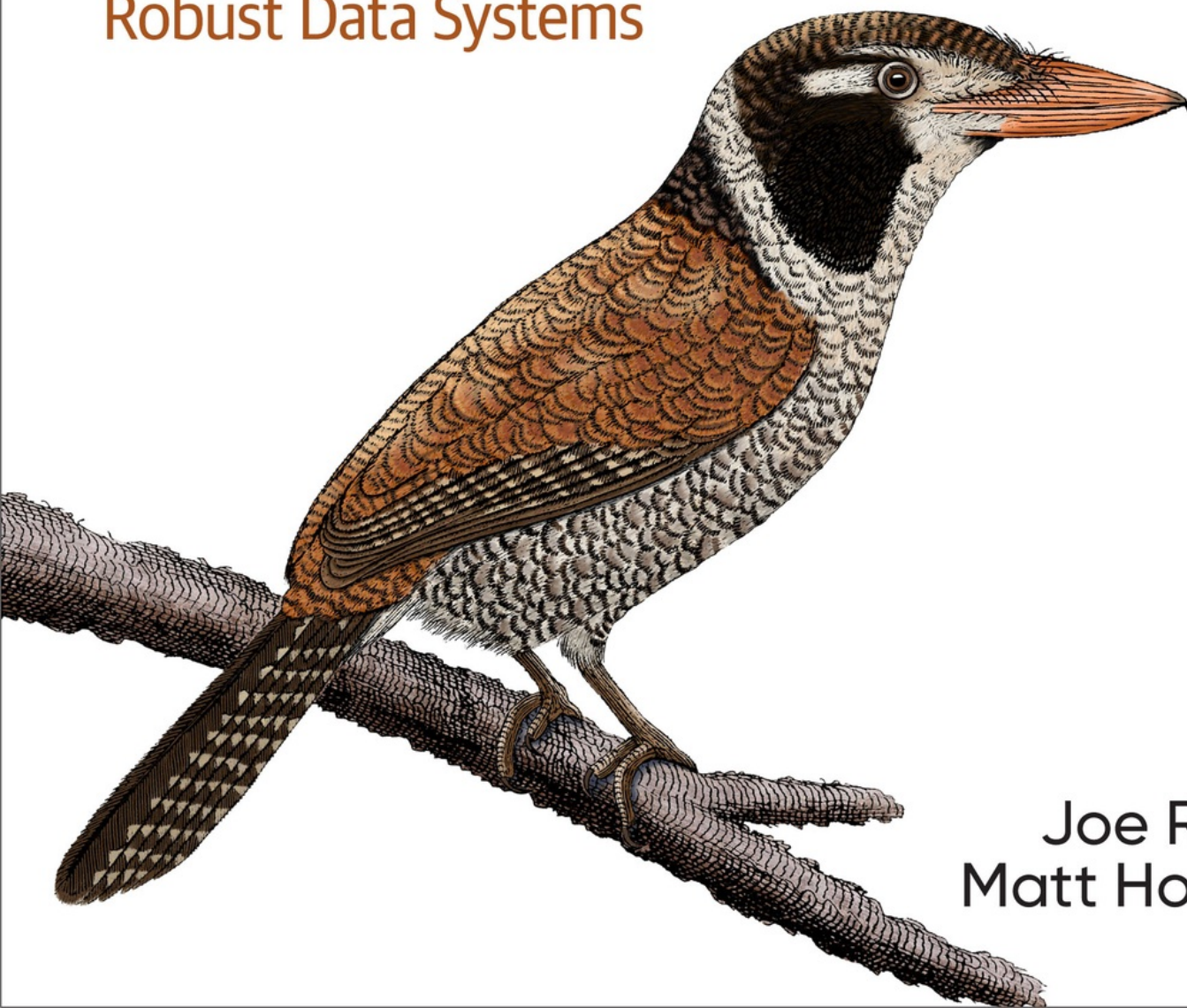


O'REILLY®

Fundamentals of Data Engineering

Plan and Build
Robust Data Systems



Joe Reis &
Matt Housley

Fundamentals of Data Engineering

Plan and Build Robust Data Systems

Joe Reis and Matt Housley

Fundamentals of Data Engineering

by Joe Reis and Matt Housley

Copyright © 2022 Joseph Reis and Matthew Housley. All rights reserved.

Printed in the United States of America.

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

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

Acquisitions Editor: Jessica Haberman

Development Editor: Michele Cronin

Production Editor: Gregory Hyman

Copyeditor: Sharon Wilkey

Proofreader: Amnet Systems, LLC

Indexer: Judith McConville

Interior Designer: David Futato

Cover Designer: Karen Montgomery

Illustrator: Kate Dullea

June 2022: First Edition

Revision History for the First Edition

- 2022-06-22: First Release

See <http://oreilly.com/catalog/errata.csp?isbn=9781098108304> for release details.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Fundamentals of Data Engineering*, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

The views expressed in this work are those of the authors, and do not represent the publisher's views. While the publisher and the authors have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the authors disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

978-1-098-10830-4

[LSI]

Preface

How did this book come about? The origin is deeply rooted in our journey from data science into data engineering. We often jokingly refer to ourselves as *recovering data scientists*. We both had the experience of being assigned to data science projects, then struggling to execute these projects due to a lack of proper foundations. Our journey into data engineering began when we undertook data engineering tasks to build foundations and infrastructure.

With the rise of data science, companies splashed out lavishly on data science talent, hoping to reap rich rewards. Very often, data scientists struggled with basic problems that their background and training did not address—data collection, data cleansing, data access, data transformation, and data infrastructure. These are problems that data engineering aims to solve.

What This Book Isn't

Before we cover what this book is about and what you'll get out of it, let's quickly cover what this book *isn't*. This book isn't about data engineering using a particular tool, technology, or platform. While many excellent books approach data engineering technologies from this perspective, these books have a short shelf life. Instead, we try to focus on the fundamental concepts behind data engineering.

What This Book Is About

This book aims to fill a gap in current data engineering content and materials. While there's no shortage of technical resources that address specific data engineering tools and technologies, people struggle to understand how to assemble these components into a coherent whole that applies in the real world. This book connects the dots of the end-to-end data lifecycle. It shows you how to stitch together various technologies to serve the needs of downstream data consumers such as analysts, data scientists, and machine learning engineers. This book works as a complement to O'Reilly books that cover the details of particular technologies, platforms and programming languages.

The big idea of this book is the *data engineering lifecycle*: data generation, storage, ingestion, transformation, and serving. Since the dawn of data, we've seen the rise and fall of innumerable specific technologies and vendor products, but the data engineering life cycle stages have remained essentially unchanged. With this framework, the reader will come away with a sound understanding for applying technologies to real-world business problems.

Our goal here is to map out principles that reach across two axes. First, we wish to distill data engineering into principles that can encompass *any relevant technology*. Second, we wish to present principles that will stand the test of *time*. We hope that these ideas reflect lessons learned across the data technology upheaval of the last twenty years and that our mental framework will remain useful for a decade or more into the future.

One thing to note: we unapologetically take a cloud-first approach. We view the cloud as a fundamentally transformative development that will endure for decades; most on-premises data systems and workloads will eventually move to cloud hosting. We assume that infrastructure and systems are *ephemeral* and *scalable*, and that data engineers will lean toward deploying managed services in the cloud. That said, most concepts in this book will translate to non-cloud environments.

Who Should Read This Book

Our primary intended audience for this book consists of technical practitioners, mid- to senior-level software engineers, data scientists, or analysts interested in moving into data engineering; or data engineers working in the guts of specific technologies, but wanting to develop a more comprehensive perspective. Our secondary target audience consists of data stakeholders who work adjacent to technical practitioners—e.g., a data team lead with a technical background overseeing a team of data engineers, or a director of data warehousing wanting to migrate from on-premises technology to a cloud-based solution.

Ideally, you're curious and want to learn—why else would you be reading this book? You stay current with data technologies and trends by reading books and articles on data warehousing/data lakes, batch and streaming systems, orchestration, modeling, management, analysis, developments in cloud technologies, etc. This book will help you weave what you've read into a complete picture of data engineering across technologies and paradigms.

Prerequisites

We assume a good deal of familiarity with the types of data systems found in a corporate setting. In addition, we assume that readers have some familiarity with SQL and Python (or some other programming language), and experience with cloud services.

Numerous resources are available for aspiring data engineers to practice Python and SQL. Free online resources abound (blog posts, tutorial sites, YouTube videos), and many new Python books are published every year.

The cloud provides unprecedented opportunities to get hands-on experience with data tools. We suggest that aspiring data engineers set up accounts with cloud services such as AWS, Azure, Google Cloud Platform, Snowflake, Databricks, etc. Note that many of these platforms have *free tier*

options, but readers should keep a close eye on costs, and work with small quantities of data and single node clusters as they study.

Developing familiarity with corporate data systems outside of a corporate environment remains difficult and this creates certain barriers for aspiring data engineers who have yet to land their first data job. This book can help. We suggest that data novices read for high level ideas, and then look at materials in the *additional resources* section at the end of each chapter. On a second read through, note any unfamiliar terms and technologies. You can utilize Google, Wikipedia, blog posts, YouTube videos, and vendor sites to become familiar with new terms and fill gaps in your understanding.

What You'll Learn and How It Will Improve Your Abilities

This book aims to help you build a solid foundation for solving real world data engineering problems.

By the end of this book you will understand:

- How data engineering impacts your current role (data scientist, software engineer, or data team lead).
- How to cut through the marketing hype and choose the right technologies, data architecture, and processes.
- How to use the data engineering lifecycle to design and build a robust architecture.
- Best practices for each stage of the data lifecycle.

And you will be able to:

- Incorporate data engineering principles in your current role (data scientist, analyst, software engineer, data team lead, etc.)
- Stitch together a variety of cloud technologies to serve the needs of downstream data consumers.

- Assess data engineering problems with an end-to-end framework of best practices
- Incorporate data governance and security across the data engineering lifecycle.

The Book Outline

This book is composed of four parts:

- Part I, “Foundation and Building Blocks”
- Part II, “The Data Engineering Lifecycle in Depth”
- Part III, “Security, Privacy, and the Future of Data Engineering”
- Appendices **A** and **B**: cloud networking, serialization and compression

In **Part I**, we begin by defining data engineering in **Chapter 1**, then map out the data engineering lifecycle in **Chapter 2**. In **Chapter 3**, we discuss *good architecture*. In **Chapter 4**, we introduce a framework for choosing the right technology—while we frequently see technology and architecture conflated, these are in fact very different topics.

Part II builds on **Chapter 2** to cover the data engineering lifecycle in depth; each lifecycle stage—data generation, storage, ingestion, transformation and serving—is covered in its own chapter. **Part II** is arguably the heart of the book, and the other chapters exist to support the core ideas covered here.

Part III covers additional topics. In **Chapter 10**, we discuss *security and privacy*. While security has always been an important part of the data engineering profession, it has only become more critical with the rise of for profit hacking and state sponsored cyber attacks. And what can we say of privacy? The era of corporate privacy nihilism is over—no company wants to see its name appear in the headline of an article on sloppy privacy practices. Reckless handling of personal data can also have significant legal ramifications with the advent of GDPR, CCPA and other regulations. In short, security and privacy must be top priorities in any data engineering work.

In the course of working in data engineering, doing research for this book and interviewing numerous experts, we thought a good deal about where the field is going in the near and long term. **Chapter 11** outlines our highly

speculative ideas on the future of data engineering. By its nature, the future is a slippery thing. Time will tell if some of our ideas are correct. We would love to hear from our readers on how their visions of the future agree with or differ from our own.

In the appendix, we cover a handful of technical topics that are extremely relevant to the day to day practice of data engineering, but didn't fit into the main body of the text. Specifically, cloud networking is a critical topic as data engineering shifts into the cloud, and engineers need to understand serialization and compression both to work directly with data files, and to assess performance considerations in data systems.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

Constant width

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

Constant width bold

Shows commands or other text that should be typed literally by the user.

Constant width italic

Shows text that should be replaced with user-supplied values or by values determined by context.

TIP

This element signifies a tip or suggestion.

NOTE

This element signifies a general note.

WARNING

This element indicates a warning or caution.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

800-998-9938 (in the United States or Canada)

707-829-0515 (international or local)

707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at

<https://oreil.ly/fundamentals-of-data>.

Email bookquestions@oreilly.com to comment or ask technical questions about this book.

For news and information about our books and courses, visit <https://oreilly.com>.

Find us on LinkedIn: <https://linkedin.com/company/oreilly-media>

Follow us on Twitter: <https://twitter.com/oreillymedia>

Watch us on YouTube: <https://www.youtube.com/oreillymedia>

Acknowledgments

When we started writing this book, we were warned by many people that we faced a hard task. A book like this has a lot of moving parts, and due to its comprehensive view of the field of data engineering, it required a ton of research, interviews, discussions, and deep thinking. We won't claim to have captured every nuance of data engineering, but we hope that the results resonate with you. Numerous individuals contributed to our efforts, and we're grateful for the support we received from many experts.

First, thanks to our amazing crew of technical reviewers. They slogged through many readings, and gave invaluable (and often ruthlessly blunt) feedback. This book would be a fraction of itself without their efforts. In no particular order, we give endless thanks to Bill Inmon, Andy Petrella, Matt Sharp, Tod Hanseman, Chris Tabb, Danny Lebzyon, Martin Kleppman, Scott Lorimor, Nick Schrock, Lisa Steckman, and Alex Woolford.

Second, we've had a unique opportunity to talk with the leading experts in the field of data on our live shows, podcasts, meetups, and endless private calls. Their ideas helped shape our book. There are too many people to name individually, but we'd like to give shoutouts to Bill Inmon, Jordan Tigani, Zhamak Dehghani, Shruti Bhat, Eric Tschetter, Benn Stancil, Kevin Hu, Michael Rogove, Ryan Wright, Egor Gryaznov, Chad Sanderson, Julie Price, Matt Turck, Monica Rogati, Mars Lan, Pardhu Gunnam, Brian Suk, Barr Moses, Lior Gavish, Bruno Aziza, Gian Merlino, DeVaris Brown,

Todd Beauchene, Tudor Girba, Scott Taylor, Ori Rafael, Lee Edwards, Bryan Offutt, Ollie Hughes, Gilbert Eijkelenboom, Chris Bergh, Fabiana Clemente, Andreas Kretz, Ori Reshef, Nick Singh, Mark Balkenende, Kenten Danas, Brian Olsen, Lior Gavish, Rhaghu Murthy, Greg Coquillo, David Aponte, Demetrios Brinkmann, Sarah Catanzaro, Michel Tricot, Levi Davis, Ted Walker, Carlos Kemeny, Josh Benamram, Chanin Nantasenamat, George Firican, Jordan Goldmeir, Minhaaj Rehman, Luigi Patruno, Vin Vashista, Danny Ma, Jesse Anderson, Alessya Visnjic, Vishal Singh, Dave Langer, Roy Hasson, Todd Odess, Che Sharma, Scott Breitenother, Ben Taylor, Thom Ives, John Thompson, Brent Dykes, Josh Tobin, Mark Kosiba, Tyler Pugliese, Douwe Maan, Martin Traverso, Curtis Kowalski, Bob Davis, Koo Ping Shung, Ed Chenard, Matt Sciorma, Tyler Folkman, Jeff Baird, Tejas Manohar, Paul Singman, Kevin Stumpf, Willem Pineaar, and Michael Del Balso from Tecton, Emma Dahl, Harpreet Sahota, Ken Jee, Scott Taylor, Kate Strachnyi, Kristen Kehrer, Taylor Miller, Abe Gong, Ben Castleton, Ben Rogojan, David Mertz, Emmanuel Raj, Andrew Jones, Avery Smith, Brock Cooper, Jeff Larson, Jon King, Holden Ackerman, Miriah Peterson, Felipe Hoffa, David Gonzalez, Richard Wellman, Susan Walsh, Ravit Jain, Lauren Balik, Mikiko Bazeley, Mark Freeman, Mike Wimmer, Alexey Shchedrin, Mary Clair Thompson, Julie Burroughs, Jason Pedley, Freddy Drennan, Jake Carter, Jason Pedley, Kelly and Matt Phillipps, Brian Campbell, Faris Chebib, Dylan Gregerson, Ken Myers, and many others.

If you're not mentioned specifically, don't take it personally. You know who you are. Let us know and we'll get you on the next edition.

We'd also like to thank the Ternary Data team, our students, and the countless people around the world who've supported us. It's a great reminder the world is a very small place.

Working with the O'Reilly crew was amazing! Special thanks to Jess Haberman for having confidence in us during the book proposal process, our amazing and extremely patient development editors Nicole Taché and Michele Cronin for invaluable editing, feedback and support. Thank you also to the superb production crew at O'Reilly (Greg and crew).

Joe would like to thank his family—Cassie, Milo, and Ethan—for letting him write a book. They had to endure a ton, and Joe promises to never write another book again ;)

Matt would like to thank his friends and family for their enduring patience and support. He's still hopeful that Seneca will deign to give a five star review after a good deal of toil and missed family time around the holidays.

Part I. Foundation and Building Blocks

Chapter 1. Data Engineering Described

If you work in data or software, you may have noticed data engineering emerging from the shadows and now sharing the stage with data science. Data engineering is one of the hottest fields in data and technology, and for a good reason. It builds the foundation for data science and analytics in production. This chapter explores what data engineering is, how the field was born and its evolution, the skills of data engineers, and with whom they work.

What Is Data Engineering?

Despite the current popularity of data engineering, there's a lot of confusion about what data engineering means and what data engineers do. Data engineering has existed in some form since companies started doing things with data—such as predictive analysis, descriptive analytics, and reports—and came into sharp focus alongside the rise of data science in the 2010s. For the purpose of this book, it's critical to define what *data engineering* and *data engineer* mean.

First, let's look at the landscape of how data engineering is described and develop some terminology we can use throughout this book. Endless definitions of *data engineering* exist. In early 2022, a Google exact-match search for “what is data engineering?” returns over 91,000 unique results. Before we give our definition, here are a few examples of how some experts in the field define data engineering:

Data engineering is a set of operations aimed at creating interfaces and mechanisms for the flow and access of information. It takes dedicated specialists—data engineers—to maintain data so that it remains available and usable by others. In short, data engineers set up and operate the organization’s data infrastructure, preparing it for further analysis by data analysts and scientists.

—From “Data Engineering and Its Main Concepts” by
AlexSoft¹

The first type of data engineering is SQL-focused. The work and primary storage of the data is in relational databases. All of the data processing is done with SQL or a SQL-based language. Sometimes, this data processing is done with an ETL tool.² The second type of data engineering is Big Data-focused. The work and primary storage of the data is in Big Data technologies like Hadoop, Cassandra, and HBase. All of the data processing is done in Big Data frameworks like MapReduce, Spark, and Flink. While SQL is used, the primary processing is done with programming languages like Java, Scala, and Python.

—Jesse Anderson³

In relation to previously existing roles, the data engineering field could be thought of as a superset of business intelligence and data warehousing that brings more elements from software engineering. This discipline also integrates specialization around the operation of so-called “big data” distributed systems, along with concepts around the extended Hadoop ecosystem, stream processing, and in computation at scale.

—Maxime Beauchemin⁴

Data engineering is all about the movement, manipulation, and management of data.

—Lewis Gavin⁵

Wow! It’s entirely understandable if you’ve been confused about data engineering. That’s only a handful of definitions, and they contain an enormous range of opinions about the meaning of *data engineering*.

Data Engineering Defined

When we unpack the common threads of how various people define data engineering, an obvious pattern emerges: a data engineer gets data, stores it, and prepares it for consumption by data scientists, analysts, and others. We define *data engineering* and *data engineer* as follows:

Data engineering is the development, implementation, and maintenance of systems and processes that take in raw data and produce high-quality, consistent information that supports downstream use cases, such as analysis and machine learning. Data engineering is the intersection of security, data management, DataOps, data architecture, orchestration, and software engineering. A data engineer manages the data engineering lifecycle, beginning with getting data from source systems and ending with serving data for use cases, such as analysis or machine learning.

The Data Engineering Lifecycle

It is all too easy to fixate on technology and miss the bigger picture myopically. This book centers around a big idea called the *data engineering lifecycle* (Figure 1-1), which we believe gives data engineers the holistic context to view their role.

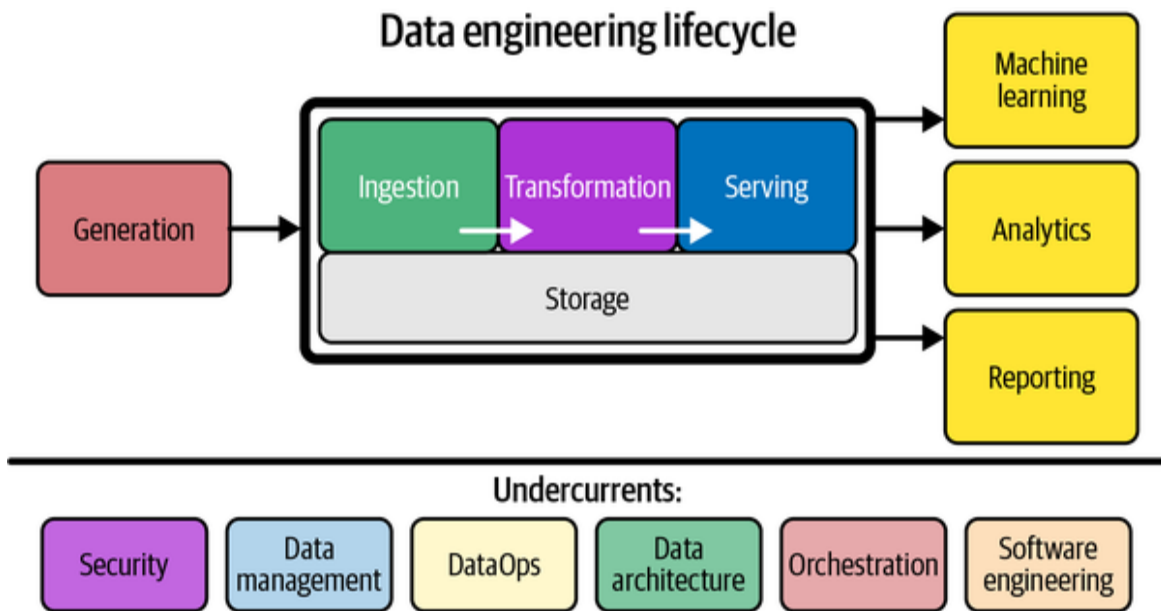


Figure 1-1. The data engineering lifecycle

The data engineering lifecycle shifts the conversation away from technology and toward the data itself and the end goals that it must serve. The stages of the data engineering lifecycle are as follows:

- Generation
- Storage
- Ingestion
- Transformation
- Serving

The data engineering lifecycle also has a notion of *undercurrents*—critical ideas across the entire lifecycle. These include security, data management, DataOps, data architecture, orchestration, and software engineering. We cover the data engineering lifecycle and its undercurrents more extensively in **Chapter 2**. Still, we introduce it here because it is essential to our definition of data engineering and the discussion that follows in this chapter.

Now that you have a working definition of data engineering and an introduction to its lifecycle, let's take a step back and look at a bit of history.

Evolution of the Data Engineer

History doesn't repeat itself, but it rhymes.

—A famous adage often attributed to Mark Twain

Understanding data engineering today and tomorrow requires a context of how the field evolved. This section is not a history lesson, but looking at the past is invaluable in understanding where we are today and where things are going. A common theme constantly reappears: what's old is new again.

The early days: 1980 to 2000, from data warehousing to the web

The birth of the data engineer arguably has its roots in data warehousing, dating as far back as the 1970s, with the *business data warehouse* taking shape in the 1980s and Bill Inmon officially coining the term *data warehouse* in 1990. After engineers at IBM developed the relational database and Structured Query Language (SQL), Oracle popularized the technology. As nascent data systems grew, businesses needed dedicated tools and data pipelines for reporting and business intelligence (BI). To help people correctly model their business logic in the data warehouse, Ralph Kimball and Inmon developed their respective eponymous data-modeling techniques and approaches, which are still widely used today.

Data warehousing ushered in the first age of scalable analytics, with new massively parallel processing (MPP) databases that use multiple processors to crunch large amounts of data coming on the market and supporting unprecedented volumes of data. Roles such as BI engineer, ETL developer, and data warehouse engineer addressed the various needs of the data warehouse. Data warehouse and BI engineering were a precursor to today's data engineering and still play a central role in the discipline.

The internet went mainstream around the mid-1990s, creating a whole new generation of web-first companies such as AOL, Yahoo, and Amazon. The dot-com boom spawned a ton of activity in web applications and the backend systems to support them—servers, databases, and storage. Much of the infrastructure was expensive, monolithic, and heavily licensed. The vendors selling these backend systems likely didn't foresee the sheer scale of the data that web applications would produce.

The early 2000s: The birth of contemporary data engineering

Fast-forward to the early 2000s, when the dot-com boom of the late '90s went bust, leaving behind a tiny cluster of survivors. Some of these companies, such as Yahoo, Google, and Amazon, would grow into powerhouse tech companies. Initially, these companies continued to rely on the traditional monolithic, relational databases and data warehouses of the 1990s, pushing these systems to the limit. As these systems buckled, updated approaches were needed to handle data growth. The new

generation of the systems must be cost-effective, scalable, available, and reliable.

Coinciding with the explosion of data, commodity hardware—such as servers, RAM, disks, and flash drives—also became cheap and ubiquitous. Several innovations allowed distributed computation and storage on massive computing clusters at a vast scale. These innovations started decentralizing and breaking apart traditionally monolithic services. The “big data” era had begun.

The *Oxford English Dictionary* defines **big data** as “extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions.” Another famous and succinct description of big data is the three *V*’s of data: velocity, variety, and volume.

In 2003, Google published a paper on the Google File System, and shortly after that, in 2004, a paper on MapReduce, an ultra-scalable data-processing paradigm. In truth, big data has earlier antecedents in MPP data warehouses and data management for experimental physics projects, but Google’s publications constituted a “big bang” for data technologies and the cultural roots of data engineering as we know it today. You’ll learn more about MPP systems and MapReduce in Chapters 3 and 8, respectively.

The Google papers inspired engineers at Yahoo to develop and later open source Apache Hadoop in 2006.⁶ It’s hard to overstate the impact of Hadoop. Software engineers interested in large-scale data problems were drawn to the possibilities of this new open source technology ecosystem. As companies of all sizes and types saw their data grow into many terabytes and even petabytes, the era of the big data engineer was born.

Around the same time, Amazon had to keep up with its own exploding data needs and created elastic computing environments (Amazon Elastic Compute Cloud, or EC2), infinitely scalable storage systems (Amazon Simple Storage Service, or S3), highly scalable NoSQL databases (Amazon DynamoDB), and many other core data building blocks.⁷ Amazon elected to offer these services for internal and external consumption through

Amazon Web Services (AWS), becoming the first popular public cloud. AWS created an ultra-flexible pay-as-you-go resource marketplace by virtualizing and reselling vast pools of commodity hardware. Instead of purchasing hardware for a data center, developers could simply rent compute and storage from AWS.

As AWS became a highly profitable growth engine for Amazon, other public clouds would soon follow, such as Google Cloud, Microsoft Azure, and DigitalOcean. The public cloud is arguably one of the most significant innovations of the 21st century and spawned a revolution in the way software and data applications are developed and deployed.

The early big data tools and public cloud laid the foundation for today's data ecosystem. The modern data landscape—and data engineering as we know it now—would not exist without these innovations.

The 2000s and 2010s: Big data engineering

Open source big data tools in the Hadoop ecosystem rapidly matured and spread from Silicon Valley to tech-savvy companies worldwide. For the first time, any business had access to the same bleeding-edge data tools used by the top tech companies. Another revolution occurred with the transition from batch computing to event streaming, ushering in a new era of big “real-time” data. You'll learn about batch and event streaming throughout this book.

Engineers could choose the latest and greatest—Hadoop, Apache Pig, Apache Hive, Dremel, Apache HBase, Apache Storm, Apache Cassandra, Apache Spark, Presto, and numerous other new technologies that came on the scene. Traditional enterprise-oriented and GUI-based data tools suddenly felt outmoded, and code-first engineering was in vogue with the ascendance of MapReduce. We (the authors) were around during this time, and it felt like old dogmas died a sudden death upon the altar of big data.

The explosion of data tools in the late 2000s and 2010s ushered in the *big data engineer*. To effectively use these tools and techniques—namely, the Hadoop ecosystem including Hadoop, YARN, Hadoop Distributed File

System (HDFS), and MapReduce—big data engineers had to be proficient in software development and low-level infrastructure hacking, but with a shifted emphasis. Big data engineers typically maintained massive clusters of commodity hardware to deliver data at scale. While they might occasionally submit pull requests to Hadoop core code, they shifted their focus from core technology development to data delivery.

Big data quickly became a victim of its own success. As a buzzword, *big data* gained popularity during the early 2000s through the mid-2010s. Big data captured the imagination of companies trying to make sense of the ever-growing volumes of data and the endless barrage of shameless marketing from companies selling big data tools and services. Because of the immense hype, it was common to see companies using big data tools for small data problems, sometimes standing up a Hadoop cluster to process just a few gigabytes. It seemed like everyone wanted in on the big data action. Dan Ariely [tweeted](#), “Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it.”

Figure 1-2 shows a snapshot of Google Trends for the search term “big data” to get an idea of the rise and fall of big data.

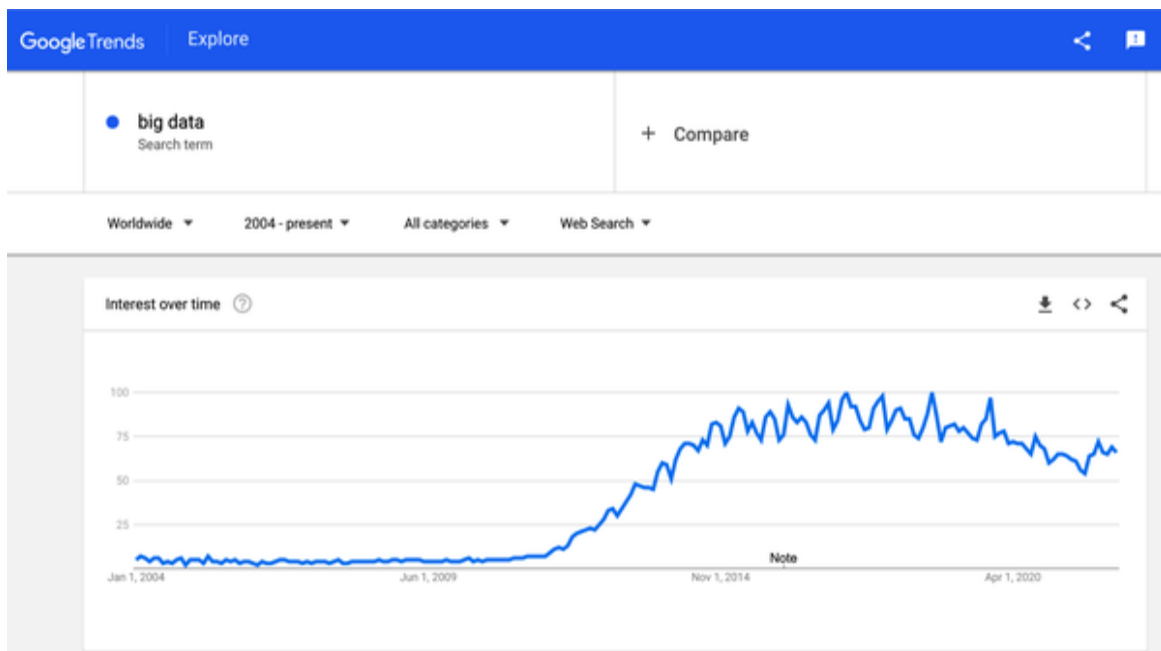


Figure 1-2. Google Trends for “big data” (March 2022)

Despite the term's popularity, big data has lost steam. What happened? One word: simplification. Despite the power and sophistication of open source big data tools, managing them was a lot of work and required constant attention. Often, companies employed entire teams of big data engineers, costing millions of dollars a year, to babysit these platforms. Big data engineers often spent excessive time maintaining complicated tooling and arguably not as much time delivering the business's insights and value.

Open source developers, clouds, and third parties started looking for ways to abstract, simplify, and make big data available without the high administrative overhead and cost of managing their clusters, and installing, configuring, and upgrading their open source code. The term *big data* is essentially a relic to describe a particular time and approach to handling large amounts of data.

Today, data is moving faster than ever and growing ever larger, but big data processing has become so accessible that it no longer merits a separate term; every company aims to solve its data problems, regardless of actual data size. Big data engineers are now simply *data engineers*.

The 2020s: Engineering for the data lifecycle

At the time of this writing, the data engineering role is evolving rapidly. We expect this evolution to continue at a rapid clip for the foreseeable future. Whereas data engineers historically tended to the low-level details of monolithic frameworks such as Hadoop, Spark, or Informatica, the trend is moving toward decentralized, modularized, managed, and highly abstracted tools.

Indeed, data tools have proliferated at an astonishing rate (see [Figure 1-3](#)). Popular trends in the early 2020s include the *modern data stack*, representing a collection of off-the-shelf open source and third-party products assembled to make analysts' lives easier. At the same time, data sources and data formats are growing both in variety and size. Data engineering is increasingly a discipline of interoperation, and connecting various technologies like LEGO bricks, to serve ultimate business goals.

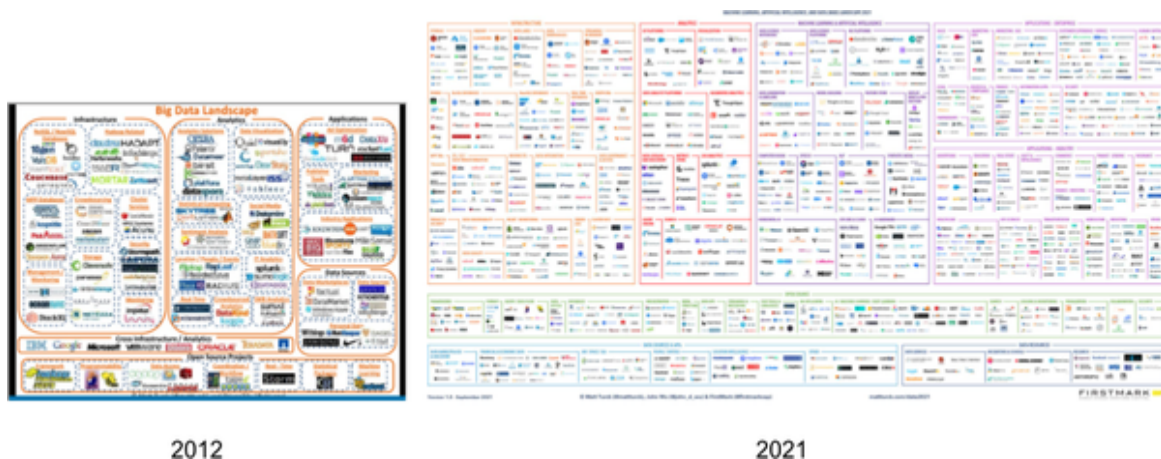


Figure 1-3. Matt Turck's Data Landscape in 2012 versus 2021

The data engineer we discuss in this book can be described more precisely as a *data lifecycle engineer*. With greater abstraction and simplification, a data lifecycle engineer is no longer encumbered by the gory details of yesterday's big data frameworks. While data engineers maintain skills in low-level data programming and use these as required, they increasingly find their role focused on things higher in the value chain: security, data management, DataOps, data architecture, orchestration, and general data lifecycle management.⁸

As tools and workflows simplify, we've seen a noticeable shift in the attitudes of data engineers. Instead of focusing on who has the “biggest data,” open source projects and services are increasingly concerned with managing and governing data, making it easier to use and discover, and improving its quality. Data engineers are now conversant in acronyms such as *CCPA* and *GDPR*;⁹ as they engineer pipelines, they concern themselves with privacy, anonymization, data garbage collection, and compliance with regulations.

What's old is new again. While “enterprising” stuff like data management (including data quality and governance) was common for large enterprises in the pre-big-data era, it wasn't widely adopted in smaller companies. Now that many of the challenging problems of yesterday's data systems are solved, neatly productized, and packaged, technologists and entrepreneurs have shifted focus back to the “enterprising” stuff, but with an emphasis on

decentralization and agility, that contrasts with the traditional enterprise command-and-control approach.

We view the present as a golden age of data lifecycle management. Data engineers managing the data engineering lifecycle have better tools and techniques than ever before. We discuss the data engineering lifecycle and its undercurrents in greater detail in the next chapter.

Data Engineering and Data Science

Where does data engineering fit in with data science? There's some debate, with some arguing data engineering is a subdiscipline of data science. We believe data engineering is *separate* from data science and analytics. They complement each other, but they are distinctly different. Data engineering sits upstream from data science (Figure 1-4), meaning data engineers provide the inputs used by data scientists (downstream from data engineering), who convert these inputs into something useful.

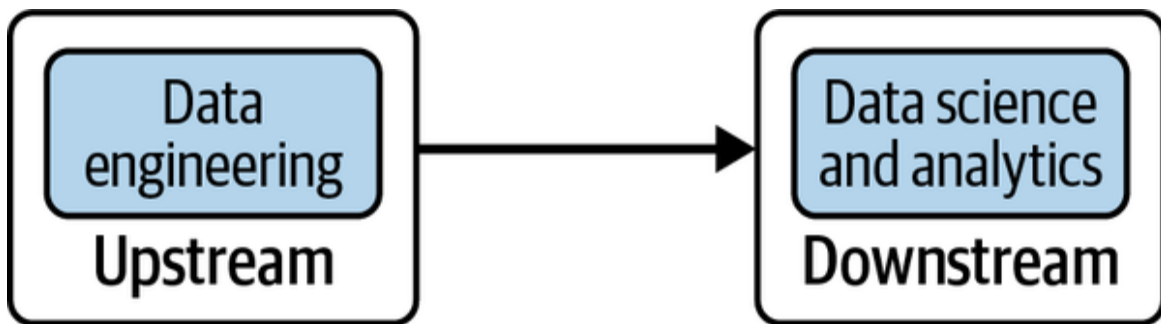


Figure 1-4. Data engineering sits upstream from data science

Consider the Data Science Hierarchy of Needs (Figure 1-5). In 2017, Monica Rogati published this hierarchy in [an article](#) that showed where AI and machine learning (ML) sat in proximity to more “mundane” areas such as data movement/storage, collection, and infrastructure.

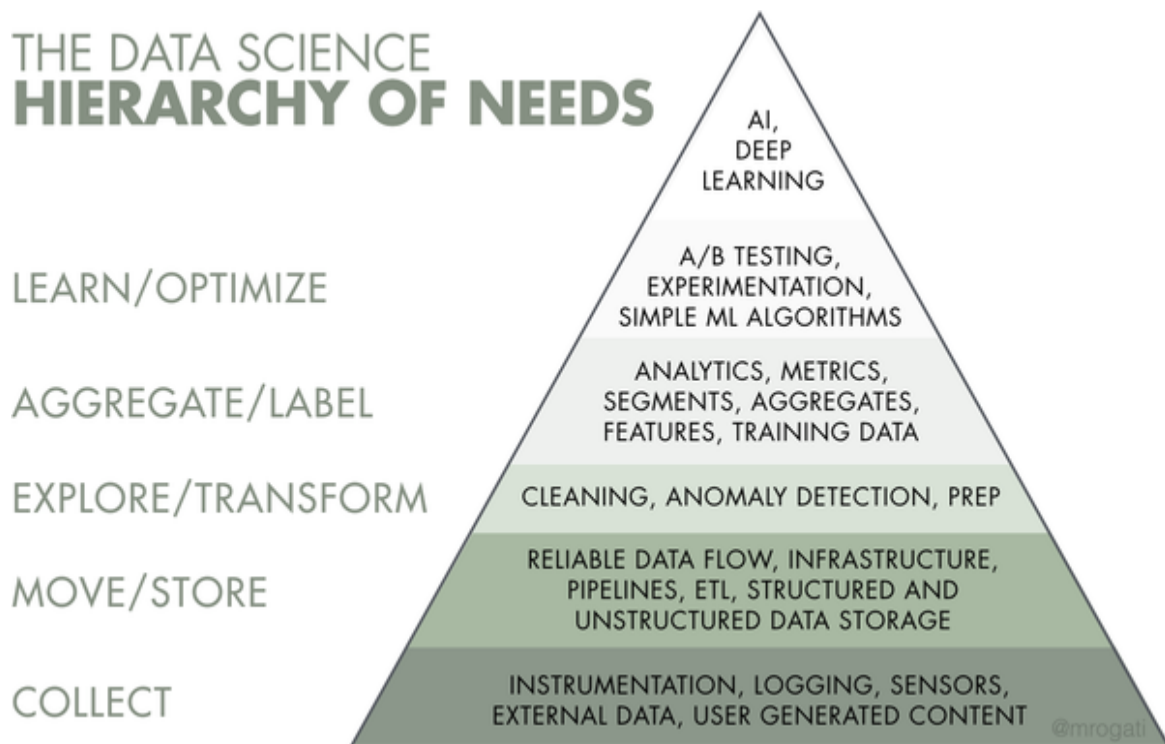


Figure 1-5. The Data Science Hierarchy of Needs

Although many data scientists are eager to build and tune ML models, the reality is an estimated 70% to 80% of their time is spent toiling in the bottom three parts of the hierarchy—gathering data, cleaning data, processing data—and only a tiny slice of their time on analysis and ML. Rogati argues that companies need to build a solid data foundation (the bottom three levels of the hierarchy) before tackling areas such as AI and ML.

Data scientists aren't typically trained to engineer production-grade data systems, and they end up doing this work haphazardly because they lack the support and resources of a data engineer. In an ideal world, data scientists should spend more than 90% of their time focused on the top layers of the pyramid: analytics, experimentation, and ML. When data engineers focus on these bottom parts of the hierarchy, they build a solid foundation for data scientists to succeed.

With data science driving advanced analytics and ML, data engineering straddles the divide between getting data and getting value from data (see [Figure 1-6](#)). We believe data engineering is of equal importance and

visibility to data science, with data engineers playing a vital role in making data science successful in production.



Figure 1-6. A data engineer gets data and provides value from the data

Data Engineering Skills and Activities

The skill set of a data engineer encompasses the “undercurrents” of data engineering: security, data management, DataOps, data architecture, and software engineering. This skill set requires an understanding of how to evaluate data tools and how they fit together across the data engineering lifecycle. It’s also critical to know how data is produced in source systems and how analysts and data scientists will consume and create value after processing and curating data. Finally, a data engineer juggles a lot of complex moving parts and must constantly optimize along the axes of cost, agility, scalability, simplicity, reuse, and interoperability (Figure 1-7). We cover these topics in more detail in upcoming chapters.



Figure 1-7. The balancing act of data engineering

As we discussed, in the recent past, a data engineer was expected to know and understand how to use a small handful of powerful and monolithic technologies (Hadoop, Spark, Teradata, Hive, and many others) to create a data solution. Utilizing these technologies often requires a sophisticated understanding of software engineering, networking, distributed computing, storage, or other low-level details. Their work would be devoted to cluster administration and maintenance, managing overhead, and writing pipeline and transformation jobs, among other tasks.

Nowadays, the data-tooling landscape is dramatically less complicated to manage and deploy. Modern data tools considerably abstract and simplify