O'REILLY®

Analytics Engineering with Microsoft Fabric and Power BI

The Definitive Guide for Building Enterprise-Grade Analytics Solutions



Early Release

RAW & UNEDITED

Shabnam Watson
& Nikola Ilic

Analytics Engineering with Microsoft Fabric and Power BI

The Definitive Guide for Building Enterprise-Grade Analytics Solutions

With Early Release ebooks, you get books in their earliest form—the authors' raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

Shabnam Watson and Nikola Ilic



OceanofPDF.com

Analytics Engineering with Microsoft Fabric and Power BI

by Shabnam Watson and Nikola Ilic

Copyright © 2026 ABI Cube Corp and Nikola Ilic. All rights reserved.

Published by O’Reilly Media, Inc., 141 Stony Circle, Suite 195, Santa Rosa, CA 95401.

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

Acquisitions Editor: Michelle Smith

Development Editor: Shira Evans

Production Editor: Ashley Stussy

Cover Designer: Karen Montgomery

Interior Designer: David Futato

Interior Illustrator: Kate Dullea

September 2026: First Edition

Revision History for the Early Release

- 2025-08-07: First Release

See *https://oreilly.com/catalog/errata.csp?isbn=9798341645875* for release details.

The O’Reilly logo is a registered trademark of O’Reilly Media, Inc. *Analytics Engineering with Microsoft Fabric and Power BI*, 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.

979-8-341-64582-0

[LSI]

OceanofPDF.com

Brief Table of Contents (*Not Yet Final*)

Part 1: Introduction to Analytics Engineering

Chapter 1: Understanding the Role of an Analytics Engineer (available)

Chapter 2: Core Analytics Engineering: Data Architectures (unavailable)

Chapter 3: Core Analytics Engineering: Data Modeling (unavailable)

Part 2: Working with Microsoft Fabric

Chapter 4: Introduction to Microsoft Fabric (available)

Chapter 5: Data Ingestion and Data Access in Microsoft Fabric (unavailable)

Chapter 6: Working with the Lakehouse in Microsoft Fabric (unavailable)

Chapter 7: Working with the Warehouse in Microsoft Fabric (unavailable)

Chapter 8: Working with the Real-Time Intelligence in Fabric (unavailable)

Chapter 9: When to Choose What in Microsoft Fabric? (unavailable)

Part 3: Working with Microsoft Power BI

Chapter 10: Understanding Power BI Core Components (unavailable)

Chapter 11: Design and Build Semantic Models (unavailable)

Chapter 12: Enhance Semantic Models with DAX (unavailable)

Chapter 13: Optimize Enterprise-scale Semantic Models (unavailable)

Chapter 14: Going Above and Beyond with External Tools (unavailable)

Part 4: Maintain an Analytics Solution

Chapter 15: Implement security and governance (unavailable)

Chapter 16: Manage the Analytics Development Lifecycle (unavailable)

Part 5: Putting It All Together

Chapter 17: Semantic Link (unavailable)

Chapter 18: Create Your Dream Fabric Architecture (unavailable)

Chapter 19: AI Analytics Engineer (unavailable)

OceanofPDF.com

Chapter 1. Understanding the Role of an Analytics Engineer

A NOTE FOR EARLY RELEASE READERS

With Early Release ebooks, you get books in their earliest form—the authors' raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the first chapter of the final book.

If you'd like to be actively involved in reviewing and commenting on this draft, please reach out to the editor at *sevans@gmail.com*.

Every business is a data business now - they just don't know it yet…

Imagine this: It's Monday morning, and you're finding yourself in yet another meeting where the finance team is arguing with the sales team about whose numbers are "right." The CFO insists that lead conversion rates are up by 23% this quarter, while the VP of Sales swears they're only seeing an 11% increase. Meanwhile, the CEO is getting frustrated because nobody can agree on basic facts about the business.

Sound familiar? If you've worked in any data-driven organization in the last five years, you've probably witnessed this exact scenario multiple times.

But, let's be honest - this isn't really (only) about the numbers being wrong. It's about something much more fundamental: the gap between those who understand the data and those who need to use it to make decisions. And that gap, dear reader, is exactly where analytics engineering comes in.

The Making of Analytics Engineering

The story about analytics engineering doesn't start as a great plan in someone's head in a Silicon Valley boardroom. It was more like... well, like most good ideas - born out of pure necessity and a healthy dose of frustration.

Back in early 2019, a group of passionate data folks were hanging out in an online community. They were sharing stories about their day-to-day jobs, and a pattern started to emerge. Everyone was dealing with the same problem: traditional data teams were cracking under the pressure of modern data demands.

Data engineers were drowning. They were supposed to be building a robust and resilient data infrastructure, but more often than not, they were also pulled into every data transformation request. "Hey, can you just quickly add this metric to our dashboard?" "Can you modify this report to show regional breakdowns?" "Why doesn't our customer count match between these two systems?"

Meanwhile, data analysts were stuck. They knew the business inside and out, they understood what questions needed answering, but they couldn't access the data they needed without going through the already overwhelmed engineering team. It was like being a chef who wasn't allowed to enter the kitchen.

And that's when someone in that online community said the magic words: "What if we had someone whose job was specifically to bridge the gap **between technology and business**? Someone who cou**ld** speak both languages and focus purely on making data usable?"

That "someone" was the analytics engineer.

The idea quickly took off. The team at dbt Labs (https://www.getdbt.com/) (the company behind the data build tool that's become synonymous with analytics engineering) started evangelizing the role. Companies started to hire for the role. Universities began offering courses on it. And here we are

today, with analytics engineering being one of the fastest-growing roles in tech.

What Exactly does an Analytics Engineer do?

We're glad you asked! But seriously, if you asked five different analytics engineers to describe their job, you'd probably get six different answers. Even the role name can be a bit of a head-scratcher at first. And professionals working *in* the data field sometimes get a bit confused about where one role ends and another begins. It's like trying to explain the difference between a latte, a cappuccino, and an espresso to someone who just wants a good coffee – they're all coffee, but the nuances also matter.

The analytics engineer role is still evolving, and it looks different at every company. But here's our take on it, based on long hours of working with many outstanding analytics engineers and seeing the value they create:

Analytics engineers are the translators between the world of raw data and the world of business decisions.

Think of analytics engineers as skilled craftspeople who step into the messy world of data storage. They don't just clean that space, but also build well-organized, clearly labeled shelves **(** data models), ensure all the items are clean and reliable (data quality and transformation), and create clear pathways so other people can easily find exactly what they need. They are the bridge between the super-technical "how the data gets collected and stored" part and the "what does this data actually mean for the business?" part.

Now, you might be wondering: "Wait, isn't that what a Data Engineer does? Or a Data Analyst?" And that's a great question! While analytics engineers work closely with both, the Analytics Engineer role has its own distinct flavor. They are not always building massive, foundational data pipelines from scratch as many data engineers do (though a sound analytics engineer will understand how they work). And they are not usually the ones deep-

diving into specific datasets to create the final dashboard or build the business report as a data analyst often does (though a skilled analytics engineer will enable them to do it significantly better).

An Analytics Engineer is situated in the specialized space in between, focusing on making data trustworthy, accessible, and optimized specifically for *analytics*. They ensure that when five different people ask “What was our revenue last month?”, they all get the *same, correct* answer, without wanting to pull their hair out trying to find it.

However, what makes analytics engineers special is that they don’t just move data around. They apply the same strict principles that software engineers leverage to build applications: version control, testing, documentation, and continuous integration. Imagine if every data transformation were as reliable and well-documented as the well-established software you use on a daily basis. That’s what analytics engineers bring to the table.

Let us give you a real-life example. Imagine you work for an e-commerce company, and your CEO wants to know the answer to a simple question: “What’s our customer lifetime value by acquisition channel?”

In the old world, this would have been a nightmare. An analyst would have had to:

1. Figure out where customer data lives (probably five different systems)
2. Figure out where transaction data lives (probably two additional systems)
3. Figure out where the acquisition channel data lives (good luck with that!)
4. Write a bunch of complex SQL to join it all together
5. Hope they didn’t make any mistakes in the logic
6. Pray that the data was actually accurate and up-to-date

7. Spend days validating the results

With an analytics engineer on the team, Figure 1-1 displays what happens instead:

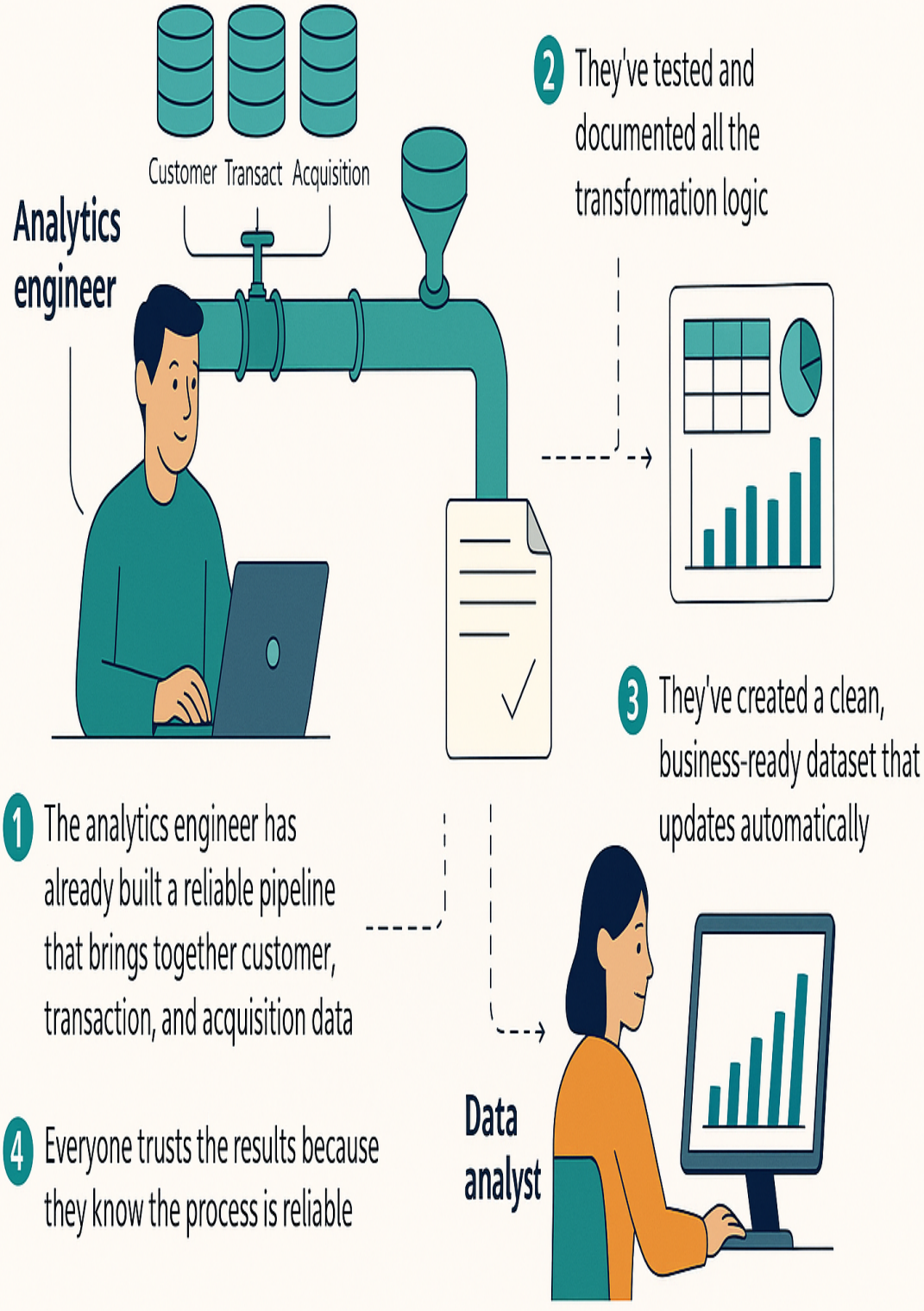
Customer Transact Acquisition
Analytics engineer
2 They've tested and documented all the transformation logic
3 They've created a clean, business-ready dataset that updates automatically
1 The analytics engineer has already built a reliable pipeline that brings together customer, transaction, and acquisition data
Data analyst
4 Everyone trusts the results because they know the process is reliable

Figure 1-1. Benefits provided by the Analytics Engineer on the data team

1. The analytics engineer has already built a reliable pipeline that brings together customer, transaction, and acquisition data
2. They've tested and documented all the transformation logic
3. They've created a clean, business-ready dataset that updates automatically. The analyst can simply query the prepared dataset and get their answer in minutes
4. Everyone trusts the results because they know the process is reliable

The data analyst gets to focus on the fun part - finding insights and telling stories. The data engineer gets to focus on infrastructure instead of ad-hoc data requests. And the business gets reliable answers to their questions. It's not a win-win scenario, but rather a win-win-win.

The Big ETL to ELT (R)evolution

Now that we have explained what analytics engineers are supposed to do, let's understand one of the biggest shifts in the data ecosystem, which made analytics engineering not just possible, but necessary - the transition from ETL to ELT. Don't worry if you don't know what these acronyms stand for, as we will break them down here.

ETL

ETL stands for Extract - Transform - Load. This is the old way of doing things. The common workflow starts by extracting the data from various sources, transforming it into the shape and format you need, and then loading it into a central storage repository (usually an enterprise data warehouse). The key thing we want you to keep in mind here is - data transformation happens *before* data enters the warehouse.

ELT

ELT stands for Extract - Load - Transform. This is the new way of handling data. We are extracting data from various sources, dumping it in the warehouse or any other storage location (who said lakehouse?) in its raw form, and then transforming it afterward. “Big deal. You just switched the order of two letters”, we hear you saying out loud.. But, believe us when we tell you that this seemingly small modification revolutionized the way data teams work.

ETL vs. ELT

In the old days, storage was expensive and compute resources were fairly limited. Therefore, you had to be really mindful about what data you kept and how you transformed it. Data engineers (by the way, they weren’t called like that back then, but rather ETL developers or Full-stack Business Intelligence developers) would have spent hours and hours tinkering and implementing complex transformation logic, and once the data was loaded into the warehouse, it was pretty much set in stone. Want to add a new column, or a new metric? You’d better pray that the data engineer has time to modify the entire data transformation pipeline.

But, then cloud computing entered our lives. Over night, the storage became cheap and the compute resources scalable to unprecedented scales. Big players, like Microsoft, Google, and Amazon started offering storage platforms that could store enormous amounts of data for a price that was a fraction of the cost of the traditional solutions. In addition, processing data at scale when needed was no longer a problem.

This shift changed everything. Why would you spend weeks writing complex transformation logic? Why not simply dump all the raw data into your warehouse or lakehouse and figure out transformations later? Why limit yourself to only the portion of the data you *think* you’ll need when storage is so cheap?

The ELT approach introduced at least three obvious benefits.

Speed

Getting the data into a warehouse or lakehouse became significantly faster. Remember, no more waiting for complex transformations to complete before you can start analyzing the data.

Flexibility

With all the data available in its raw form, data professionals could explore new questions and create new metrics without interfering with upstream pipelines.

Simplicity

Transformations could now be written in multiple programming languages,including SQL (Structured Query Language), which is the language of choice for the majority of data professionals.

However, there is a small catch. Somebody still has to *actually perform* those transformations. The data doesn't magically transform itself. And, that's exactly where analytics engineers jumped in.

Figure 1-2 displays how different data roles evolved with the shift from ETL to ELT.

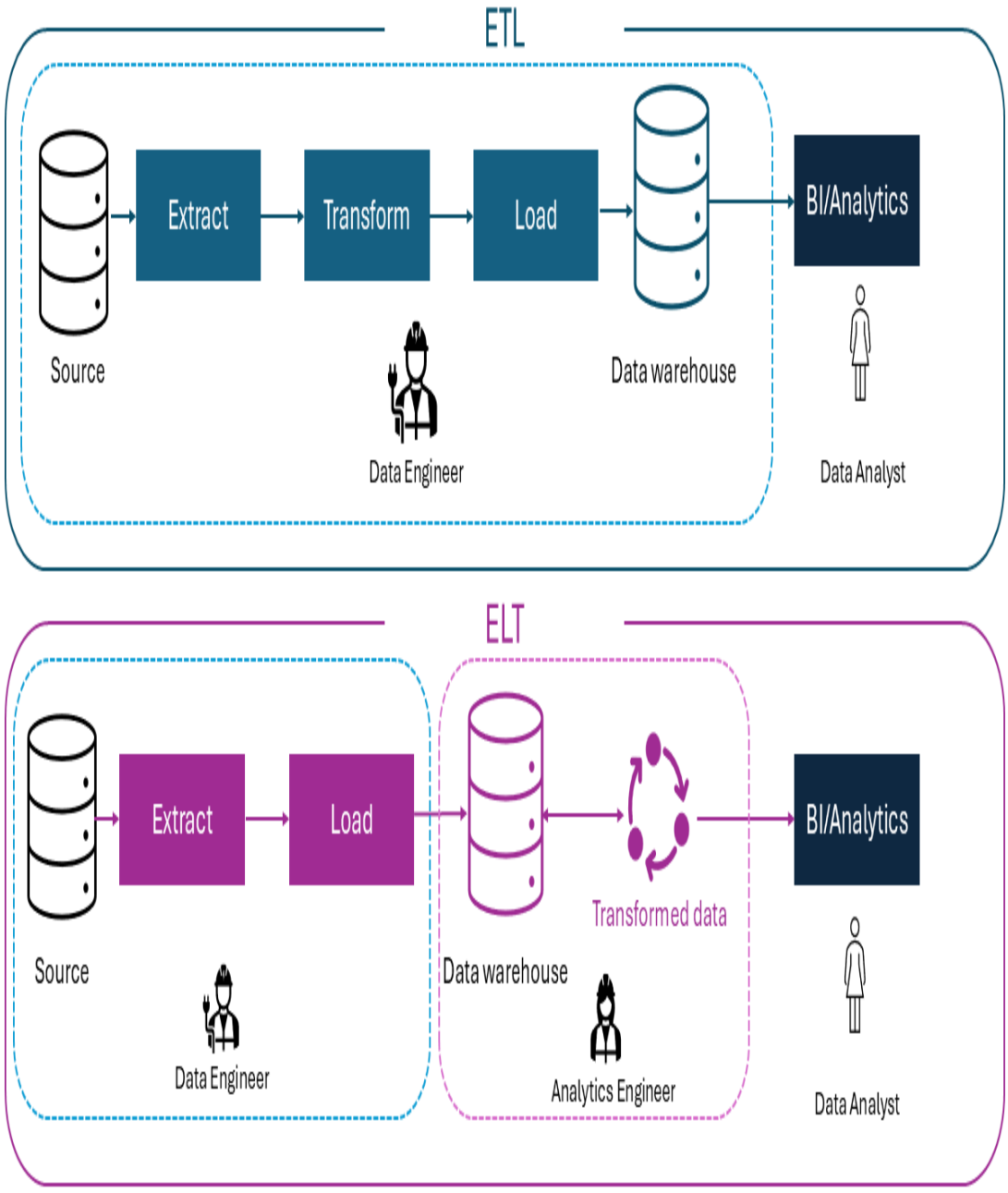

Figure 1-2. Data roles evolution with the broader adoption of the ELT workflow

To put in the scope of the ETL vs. ELT paradigms: in the ETL world, data engineers owned the entire pipeline from beginning to end. In the ELT world, data engineers can focus on the EL part, while analytics engineers

take ownership of the “T” part - transforming that raw data into something useful.

This separation of work made everyone’s life better. Data engineers could focus on infrastructure and reliability. Analytics engineers could focus on business logic and data modeling. And data analysts could focus on extracting insights and storytelling.

Analytics Engineer vs. Data Engineer vs. Data Analyst

However, regardless of how this distinction makes absolute sense in theory, in real life, the lines are blurred, and tasks between these roles often overlap, especially in smaller organizations. So, in this section, we’ll try to break down common tasks for each of these three roles and clarify distinctions.

Data Engineer

Let’s kick it off with data engineers. Think of a data engineer as someone who is an infrastructure specialist. Someone who is a city planner and a utility worker in the data world. They build and maintain pipelines through which water (data) flows.

The core responsibilities of a data engineer are as follows:

- Building pipelines that can handle millions of records per day
- Ensuring that these pipelines won’t break if there are any upstream changes in the source system
- Optimizing storage performance and costs
- Building APIs (API stands for Application Programming Interface, as a connection between two software components) and other interfaces that applications may leverage to access data

- Ensuring data security and compliance with numerous regulations

A typical toolset of a data engineer consists of programming languages such as Python, Scala, or Java, and an understanding of distributed systems, databases, and cloud infrastructure. They are the ones who get called at 3 AM if the pipeline breaks.

If you want to learn more about data engineering in general, we suggest the great book: "Fundamentals of Data Engineering" by Joe Reis and Matt Housley:

Data Analyst

Data analysts are the business translators and storytellers. They are like investigative journalists who happen to work with spreadsheets and charts instead of doing interviews.

Here is the list of typical data analyst's tasks:

- Understanding what questions the business needs answered
- Exploring data to find interesting patterns and insights
- Creating dashboards and reports that help people make decisions
- Presenting findings to business stakeholders, in a way that they can understand and act on
- Developing domain expertise in the business area they are supporting

They usually work with various BI tools, such as Power BI, Tableau, Looker, or similar. They also need to be great communicators and have a thorough understanding of the business domain they are supporting.

Analytics Engineer

Here we are! Analytics engineers are bridge builders and quality controllers. They design how the data flows through the organization.

The core tasks of an analytics engineer would be:

- Taking raw data and transforming it into cleaned and conformed datasets
- Applying well-known software engineering principles in data realm (testing, documentation, version control)
- Building data models that reflect business processes and definitions
- Ensuring data quality and catching errors before they reach end users
- Creating and supporting self-service capabilities, so analysts can answer their own questions

Analytics engineers are usually proficient with programming languages like SQL or Python, and need to understand both technical concepts and business requirements. In the following sections, we will expand on these points and examine what a typical day in the life of an analytics engineer looks like.

To wrap up this section - if we think about our company as a newspaper,data engineers would be responsible for the printing press and distribution network; analytics engineers would be the editors who check facts and structure the content; data analysts would be journalists who write actual stories.

To put it all together, Table 1-1 summarizes the skills and responsibilities of the aforementioned data professionals.

Table 1-1. The skills and responsibilities matrix for each of the roles, along with the likely proficiency level in that particular category.

Skill/Responsibility	Data Engineer	Analytics Engineer	Data Analyst
Infrastructure	High	Medium	Low
Data modeling	Medium	High	Medium
Business analysis	Low	Medium	High
Coding/Programming	High	Medium	Low
Data visualization	Low	Medium	High
System design	High	Low	Low
Stakeholder communication	Low	Medium	High

A Day in the Life of an Analytics Engineer

If you're imagining someone staring at a glowing screen, with hundreds of lines of code on the screen, like in a hacker movie... well, there's some of that (minus the dramatic Hollywood lighting, of course). But seriously, one of the most rewarding things about being an analytics engineer is that no two days are exactly the same. One day you might be deep in thought designing a new data model, the next you could be troubleshooting a weird data issue, and the day after that, you might be brainstorming with the marketing team. However, even with that variety, there are some core aspects that are almost always part of the job. Let's break them down in plain English:

Data Modeling (The Blueprint Designer)

This is perhaps the most important task of an analytics engineer. Imagine you're building with LEGOs. You could just dump all the bricks in a giant pile and hope for the best, right? Alternatively, you could design a clear plan, sort your bricks, and decide how they should be connected to build something awesome. Data modeling is like that for data. We design the structure - how different pieces of information (think of customers, orders, products) relate to each other - so that it's easy to understand, quick to query, and makes sense for the people who need to analyze it. We're creating the official, go-to "map" of the company's analytical data.

Data Transformation (The Data Chef)

Raw data, straight from the source, is often… well, a bit messy. It may have missing pieces, unusual formatting, or simply not be in the required shape for analysis. So, a big part of an analytics engineer's job is taking that raw data and transforming it. They clean it up, combine it with other data, calculate new things (e.g. "customer lifetime value"), and basically get it into proper condition. This often happens withín the ELT process that we previously discussed. Think of an analytics engineer as a chef taking raw ingredients and turning them into a delicious, ready-to-serve meals.

Building and Maintaining Data Pipelines (The 'Last Mile' Plumber)

While Data Engineers often build the highly complex, company-wide data highways, analytics engineers often focus on the "last mile" pipelines. These are the pathways that take our nice, transformed data from the central data repository and deliver it smoothly to the business intelligence tools, dashboards, and reports that people use every day. Analytics engineers make sure these pipelines are reliable and efficient.

Ensuring Data Quality and Reliability (The Data Detective)

This is of paramount importance. If the business can't trust the data, then all our fancy models and transformations are useless. Therefore, an

analytics engineer spends a lot of time putting checks and balances in place to ensure the data is accurate, complete, and up-to-date. They investigate when numbers look off, search for the source of errors, and fix them. It's like being a detective, always on the lookout for clues if something isn't quite right.

Documentation and Defining Metrics (The Dictionary Writer)

This one probably sounds a bit boring, but it's still incredibly important. If "revenue" means one thing to the sales team and another thing to the finance team, you are in trouble. Analytics engineers help implement key business metrics clearly, and they document everything along the way - what data means, where it comes from, and how it's calculated. This creates a shared understanding across the company. This way, analytics engineers basically create the official company dictionary for data.

Working with Stakeholders (The Translator & Collaborator)

Going back to the start of this section - analytics engineers don't just sit in a dark room with the data. They talk to a whole range of different people – business analysts, product managers, marketing teams, sometimes even executives. An analytics engineer needs to understand what questions they're trying to answer, what problems they're facing, and what kind of data would help them. Then they translate those needs into technical solutions.

To conclude, a regular day in the life of an analytics engineer usually represents a mix of technical work, problem-solving, and communication. It's about being both a data architect and a data artisan, ensuring the information flowing through a company is not just present, but also has a purpose.

What's in the Analytics Engineer Toolbox?

Now that we know *who* an analytics engineer *is* (and who is not), let's try to explain what's in their toolbox - simply put, *how* they perform the tasks they are supposed to do. Here is the list of skills and tools you should have under your belt to become a successful analytics engineer.

SQL

> If there is a single skill that is the undisputed king of the "Analytics Engineering Land", it's definitely SQL. SQL stands for Structured Query Language and represents the *lingua franca* for the majority of data professionals. You may not know SQL and still thrive in various data-related roles, but this doesn't apply to analytics engineers - for them, SQL know-how is a must.

Data modeling

> Another non-negotiable skill. A sound analytics engineer must be aware of different data modeling concepts and patterns and thoroughly understand, not only their advantages and disadvantages, but also when to use them. We cover data modeling as part of core analytics engineering in-depth in Chapter 3.

Data architectures

> What is the difference between a warehouse and a lakehouse? How about relational database management systems and data lakes? This is the "bread and butter" of the analytics engineering workloads - the way the data is stored predominantly affects many subsequent actions. Hence, as an analytics engineer, you need to grasp the fundamentals of various architectural paradigms. We cover data architectures as part of core analytics engineering in-depth in Chapter 2.

Version control

> In larger organizations, collaboration and teamwork are something that comes as a natural requirement. Therefore, understanding core concepts

of version control systems, such as Git, is of paramount importance. We cover the topic of managing the analytics development lifecycle in Chapter 16.

Python

While SQL is still a primary language of analytics engineering, having some Python skills is an added bonus. Python is a versatile programming language that can be used in a variety of scenarios, which don't include data-related scenarios only. However, from a data manipulation point of view, Python may sometimes even outmuscle SQL. Think of scenarios when you need to interact with APIs, or perform complex data quality checks. Therefore, even though Python is not a top requirement for analytics engineers, it will still add a massive value to your "weight" as an analytics engineer.

Soft skills

In the myriad of technical requirements and hard-core programming skills, we often tend to forget something that really matters for analytics engineers. Soft skills are frequently an underrated superpower. The thing is - we can't "install" them, but they are still a must for someone who thrives in the analytics engineer role. Think of communication, problem-solving, team-orientation, curiosity, and, why not, saying out loud, empathy as well. Putting yourself in the shoes of the people who will use your data is sometimes more important than building the best data model in the world.

> **NOTE**
>
> You may have noticed that we've barely mentioned any particular tool or vendor in this section. Don't worry, we didn't forget it. It's been done on purpose, as building and nurturing your core analytics engineering skills is more important than knowing a particular tool inside and out. We don't want to say that learning a specific tool shouldn't be your goal - on the contrary! However, our goal in this part of the book is to look at the analytics engineer through a tool-agnostic lens and equip you with the fundamentals. This will make you more resilient and enable an easier transition between various tools and environments in the future.
>
> In the later parts of the book, you'll learn how to apply these core principles in the Microsoft Data Platform solutions.

What about dbt (Data Build Tool)?

Writing about analytics engineering without explicitly mentioning dbt (Data Build Tool) may sound…well, strange. The reason is simple - dbt was the first to coin the title Analytics Engineer in 2019, and since then has been constantly evangelizing the role.

For those of you who may not be aware, dbt is a tool that enables transforming raw data into clean, tested, and documented datasets, all using SQL. The key advantage of dbt is the native support for core software engineering principles - version control, automated testing, modular design, and documentation - on top of data models. Since it doesn't extract or load any data, but rather focuses solely on the transformation (T part of the ELT), it usually sits on top of a data warehouse, as an additional layer between the warehouse and the data serving layer.

We don't want to take away even a bit of credit from dbt for evangelizing the analytics engineer role. However, we believe that, similar to the medallion architecture design pattern that was introduced by Databricks and is now widely adopted across numerous platforms, the analytics engineer role has also outgrown its creators. You can absolutely be a fantastic analytics engineer without knowing dbt at all.

If you want to learn more about the medallion architecture, we strongly encourage you to read the book “Building Medallion Architectures” by Piethein Strengholt:

Summary

Born out of the growing need to manage vast data volumes efficiently, the analytics engineer role emerged to address a gap between traditional data engineers, who focus heavily on infrastructure, and data analysts, who interpret data to tell business stories. However, analytics engineers are not just a simple fill-the-gap role. They are also taking care that the best software engineering practices, such as version control, testing, documentation, and continuous integration, are in place to ensure data consistency, accuracy, and reliability within the organization.

Although we are sure that you’ll find hundreds of definitions about the analytics engineer responsibilities, their typical tasks include: data modeling, transforming messy raw data into structured formats, creating reliable data pipelines, ensuring data quality, clearly documenting business metrics, and communicating with diverse stakeholders. While analytics engineers primarily use SQL and focus on data modeling techniques, familiarity with Python, data architectures, and soft skills significantly enhances their efficiency.

With the industry shift from ETL (Extract->Transform-> Load) to ELT (Extract-> Load-> Transform), analytics engineers have taken on greater responsibility for transforming raw data after it reaches storage, thus enabling data engineers and data analysts to focus more effectively on their respective tasks and responsibilities.

In the next chapter, we’ll take you on a tour through various data architectures.

OceanofPDF.com

Chapter 2. Introduction to Microsoft Fabric

> **A NOTE FOR EARLY RELEASE READERS**
>
> With Early Release ebooks, you get books in their earliest form—the authors' raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.
>
> This will be the fourth chapter of the final book.
>
> If you'd like to be actively involved in reviewing and commenting on this draft, please reach out to the editor at *sevans@gmail.com*.

Microsoft Fabric is a cloud-based platform that brings various data tools into one place. Think of it as a one-stop shop for working with data. If you are wondering why you would need an end-to-end analytics platform, imagine this:

It's Monday morning (and let's be honest, we are not exactly fans of Monday mornings), when your boss drops a last-minute request on your desk:

"I need a report showing sales trends, customer behavior, and product returns by lunchtime."

Before Fabric, this request would have kicked off a whole chain reaction. You would have to message the data engineer to pull and clean the data. That task alone could've taken hours, maybe even days, in the past. Then you would have had to ask the data scientist to run some quick analysis on that data using a separate tool. Finally, you would have had to get the enriched data from the data scientist so you could build the report in Power

BI. With so many handoffs and disconnected tools, it was anyone's guess whether the report would be ready by lunchtime.

Now imagine you're working in Fabric where all key players have access to the same environment called a workspace. The data engineer has already built a pipeline that brings the latest data into Fabric on a schedule. Because all Fabric tools read and write the same open-standard data format, the data scientist can easily access that same data and enrich it with sentiment analysis using built-in notebooks. Finally, you're able to transform the enriched data and build an interactive dashboard in Power BI - all without ever leaving Fabric.

No more application hopping. No more data duplication. No more broken handoffs. Just one connected platform, one smooth workflow, and one very happy boss!

Whether you're an analytics engineer, a data scientist, or a business analyst, you can all work together in Fabric in the same easy-to-use environment. Everything in Fabric is connected and works smoothly together. Fabric supports the entire analytics process - from ingesting or accessing data remotely, all the way to presenting it through reports, dashboards, and alerts.

All the tools can read and write the same open-source data format to the same place, called OneLake, which is like a shared hard drive in the cloud. Additionally, you have access to built-in AI features and governance tools.

Figure 2-1 shows a high level overview of the main components of Fabric.

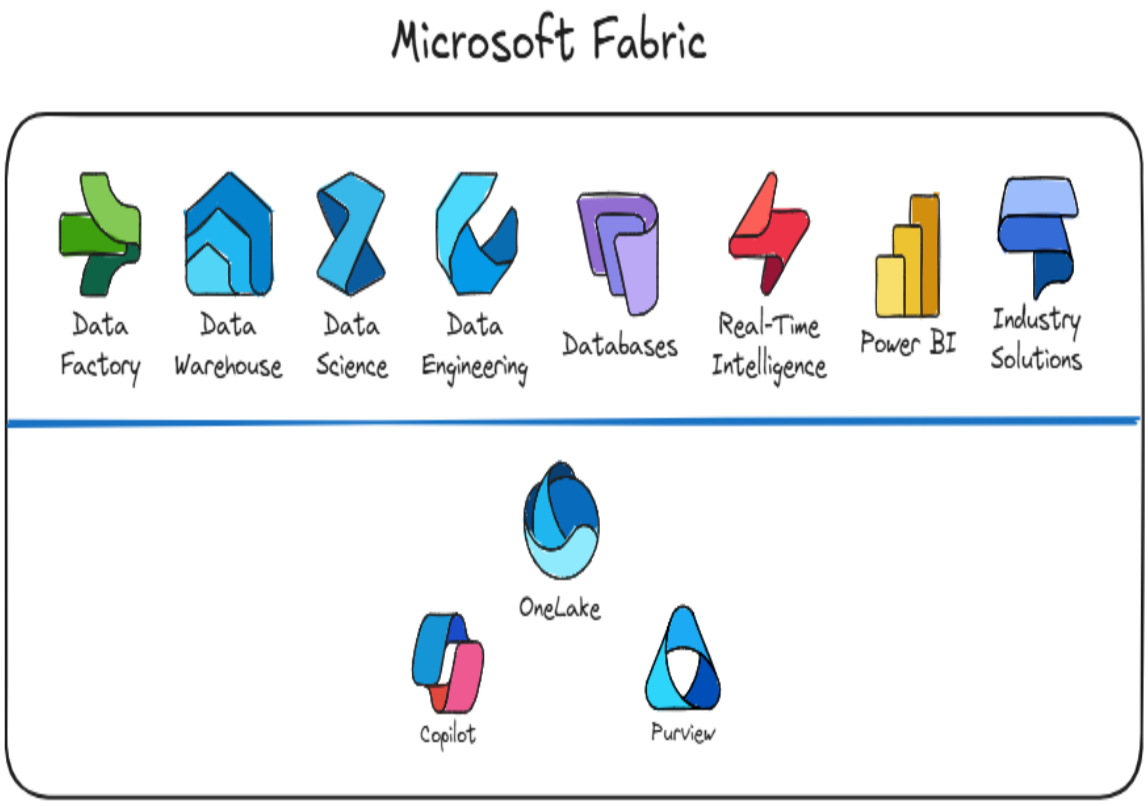

Figure 2-1. A high level overview of core Fabric components.

Fabric is designed to be straightforward to use but powerful enough for big enterprise needs. It enables organizations to create and share reports at scale by providing a clear, high-level view of the business. Even though Fabric is an enterprise-grade analytics platform, it is also suitable for small business workloads, thanks to its price and architecture flexibility where you can choose what parts you want to work with.

One key attribute of Fabric is that it has a Software as a Service model (SaaS). This is critical because IT administrators don't need to set up servers, infrastructure, and networking which is the case for on-premises systems or even to some degree for Platform as a Service (PaaS) products in the cloud. This gives organizations the opportunity to be agile and invest their time and efforts on data vs. infrastructure or networking. Even though a few of the Fabric tools were previously available in Microsoft Azure with a PaaS model; many organizations still found it difficult to stitch these tools together. On the other hand, the SaaS model of Power BI was a huge

contributor to its success a decade earlier. Microsoft followed the same path for Microsoft Fabric's adoption of the SaaS model and added many more capabilities next to it,

In the rest of this chapter, we will first review fundamental Fabric concepts such as Tenant, Capacities, Workspaces Then we will provide a high-level introduction for each component and share the pricing model.

<note>Please note that while Purview is not a component of Fabric, it is listed here since it plays an important role in governance in Fabric. </note>

Fabric Core Concepts

In order to understand how Fabric works, it is important that you understand the concepts of tenant, capacity, and workspaces. Figure 2-2 shows the relationship between a tenant, Fabric capacities, and Fabric workspaces. In the next sections, we will explain each of these concepts.

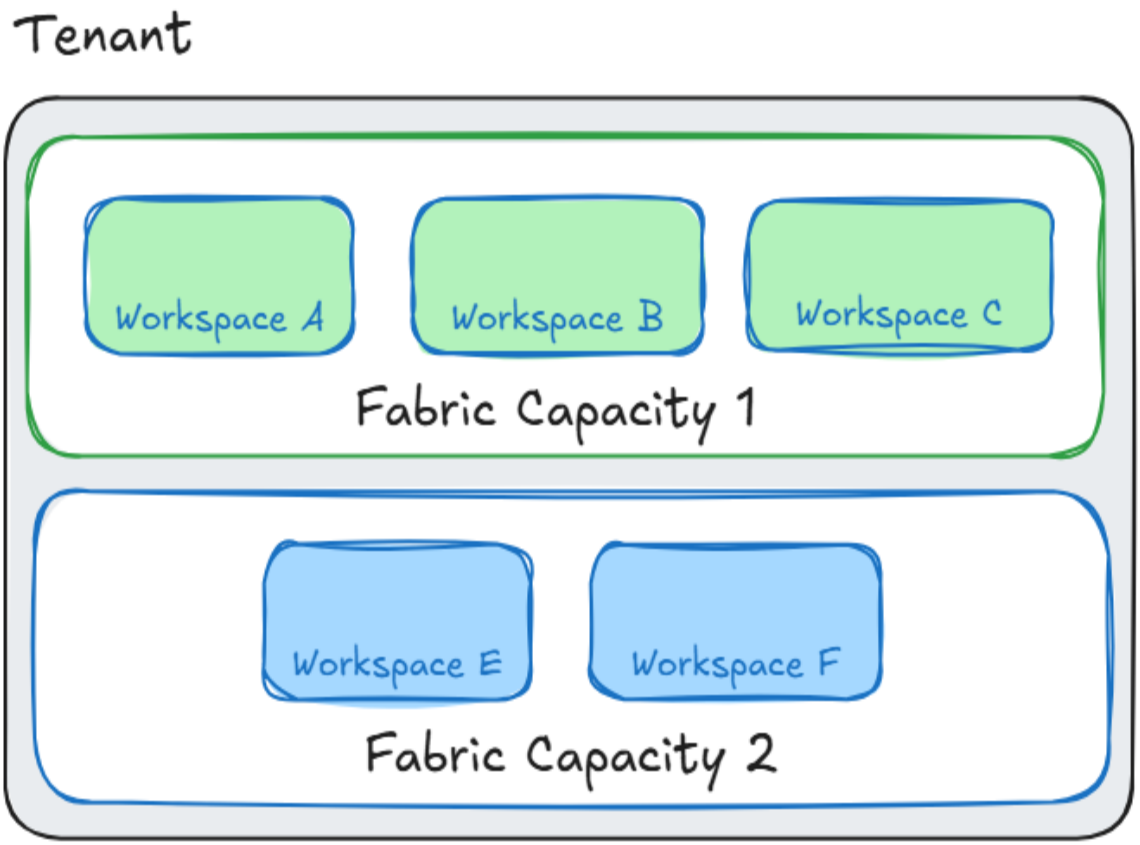

Figure 2-2. Tenant, capacities and workspaces.

Tenant

Fabric works inside a Microsoft Entra[1] tenant. A tenant is a secure container or home for your organization's Microsoft services. Microsoft's Entra ID, licenses, Azure subscriptions, and Fabric all live inside the tenant.

If your organization doesn't already have a tenant, Microsoft creates one automatically when you sign up for any Microsoft cloud service such as Microsoft 365, Azure, or Fabric (including a free trial). By default, the new tenant will get a placeholder domain that ends in *.onmicrosoft.com*. For example, a company called Contoso would get contoso.onmicrosoft.com.

You can keep using this default domain or attach your own registered domain, for example contoso.com. If you attach your own domain, your users can sign in with emails like username@contoso.com instead of

username@contoso.onmicrosoft.com. You can learn more about using your organization's domain here.

Once the tenant is in place, you can start adding resources such as Fabric capacities.

Capacity

Capacities are collections of compute (CPU), memory, and storage resources that you purchase from Microsoft to run Fabric workloads. A capacity's compute power is measured in an abstract unit of Capacity Units (CU). The more CUs a capacity has, the more compute power it has.

Once you have a capacity, you can have multiple Fabric items in that capacity. For example, you can have many Power BI reports, Power BI semantic models, Warehouses, data pipelines, and many more items, all powered by the same capacity.

Depending on the size of your organization and workloads, you may only need one capacity or you may choose to work with several capacities. Capacities are transparent to most users. It means that regular users other than tenant or capacity admins don't need to know or worry about which capacity is being used.

You can purchase Fabric capacities from Azure's portal. Capacities are labeled using SKUs (Stock Keeping Units). They start from F2 and move all the way to F2024. The higher the CU, the higher computer power they have.

> **TIP**
>
> SKU is a unique code or label used to identify and track a specific product or item in a store or system. In the case of Fabric, think of Fabric SKU as the Fabric capacity your purchasing department buys from Microsoft.

Table 2-1 lists Fabric SKUs along with their corresponding Power BI SKUs.[2]

Table 2-1. Fabric Capacities (SKUs)

SKU	Capacity Units (CU)	Power BI SKU	Power BI v-cores
F2	2	-	0.25
F4	4	-	0.5
F8	8	EM/A1	1
F16	16	EM2/A2	2
F32	32	EM3/A3	4
F64	64	P1/A4	8
Trial	64	-	8
F128	128	P2/A5	16
F256	256	P3/A6	32
F512	512	P4/A7	64
F1024	1024	P5/A8	128
F2048	2048	-	256

Workspace

In Fabric, workspaces are logical grouping of Fabric items. They live inside a capacity. Workspaces act as containers for Fabric items *and* you use them

to control who can access and do what with those items. Figure 2-3 shows a Fabric capacity with various Fabric items created across three workspaces.

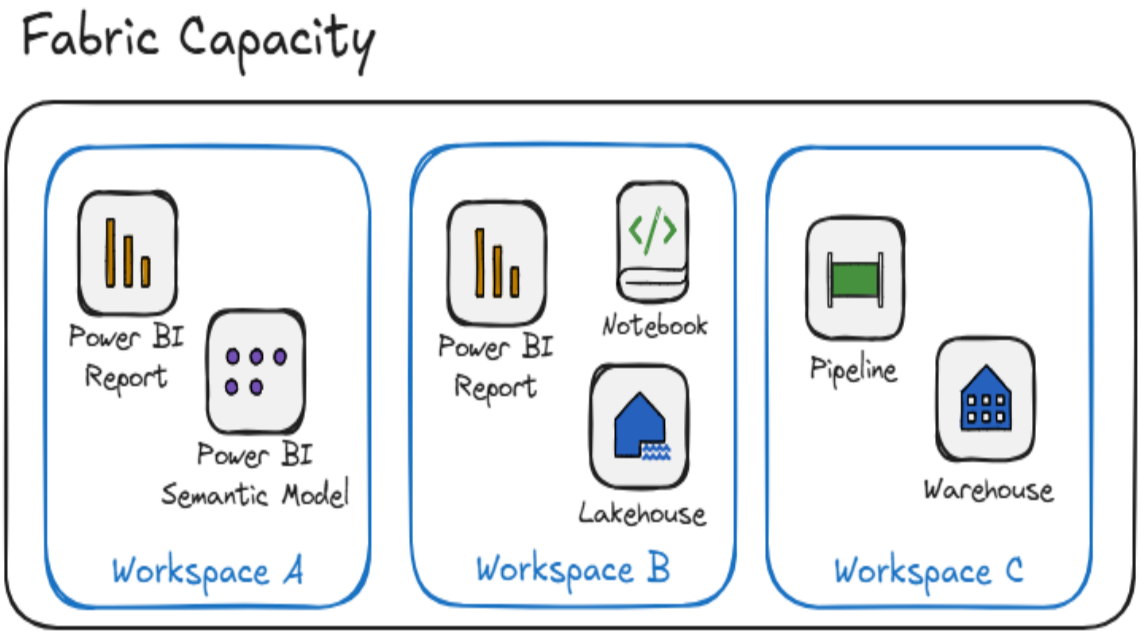

Figure 2-3. Workspaces within a capacity with various Fabric items.

In the following sections, we will briefly introduce all the capabilities of Fabric. While these capabilities have a wide range and enable enterprise-grade end-to-end projects, it does not mean that you have to learn all the capabilities and use them in every project. Don't let the long list of capabilities overwhelm you! Fabric provides a wide variety of tools that you can choose from based on your individual project requirements. Some projects may only use Power BI items, while others may only use Notebooks and Lakehouses, and yet others may use every offering to create a solution. The good news is that no matter which set of items you work with, they have familiar user interfaces, seamlessly work together, and follow the same security and governance model.

The Main Components of Microsoft Fabric

Fabric brings together different types of data tools into one easy-to use SaaS platform. Since OneLake is at the foundation of Fabric, we will start with

OneLake.

OneLake

OneLake is a logical data storage within Fabric which serves as the single storage location for all an organization's data. In fact, OneLake is often referred to as OneDrive for data. Just as an organization has a single OneDrive, it has a single OneLake for all its data.

OneLake is open, scalable, and it supports both structured and unstructured data. Being open means that it supports open standard file formats such as Delta Parquet. This openness ensures flexibility and prevents vendor lock-in, and enables organizations to use their preferred tools. Scalable means that you can add more storage as needed since it is built on Azure Data Lake Storage (ADLS) Gen 2.

Finally, OneLake operates as a unified Data Lake across different Fabric workloads, making it easy to access and share data without duplication. This is possible because all Fabric workloads can read/write Delta Parquet format natively.

> **NOTE**
>
> As you learned in Chapter 2, a Data Lake is a centralized storage system that allows organizations to store large amounts of structured, semi-structured, and unstructured data in its raw and transformed formats. Unlike traditional databases that require data to be pre-processed and structured before storage, a data lake enables organizations to ingest data from multiple sources without predefined schemas. This flexibility makes it ideal for big data analytics, machine learning, and real-time processing.

Data Factory

All data projects start with ingesting or accessing data. Data Factory is the data ingestion, transformation, and orchestration tool within Fabric. Data Factory combines the ease of use and low-code/no-code experience of Power Query with the scale of Azure Data Factory. By doing so, it makes it possible for both low-code/no-code developers (sometimes called citizen

developers) and code-first developers (sometimes called pro developers) to work together in the same environment. Both types of developers are able to ingest and transform data from hundreds of data sources at large scale.

> **TIP**
>
> Orchestration refers to the process of defining, scheduling, and executing a sequence of data ingestion, transformation, and loading activities.

Data Factory has four high level Fabric items that you can choose from based on your use case and skill set. Figure 2-4 provides a high-level overview of what each item is mainly used for and the typical skill sets required to use them. Let's review each item.

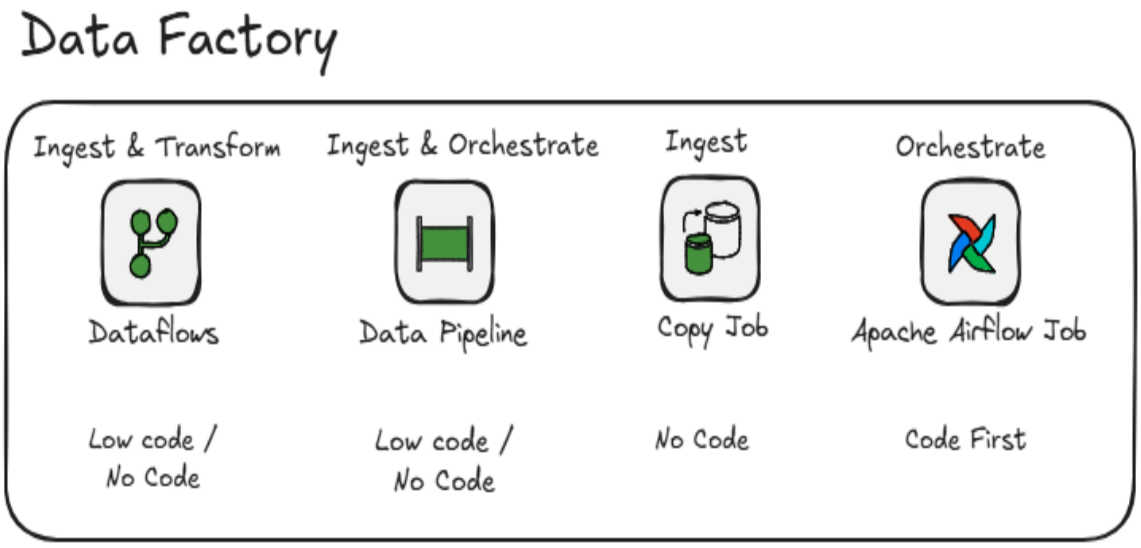

Figure 2-4. Overview of Data Factory components and required skill sets

Dataflows[3] allow you to ingest data from hundreds of data sources and use more than 300+ transformations with the low-code/no-code experience of PowerQuery. These are sometimes referred to as Dataflows Gen 2, compared to Dataflows Gen 1 in Power BI. One major difference is that with Dataflows Gen 1, the data stays in the dataflow whereas with Dataflows (Gen2), you can load the data into multiple destinations such as a

Lakehouse, a Warehouse, an Azure SQL Database, and many more. Plus, Dataflows support parameterization which allows you to dynamically customize dataflows.

Data Pipelines are the main cloud-scale orchestration tool within Fabric. With a straightforward interface, you can chain various activities, including data ingestion, data transformation, and control flow activities together and schedule them to run. Some of the data transformation activities include Dataflows, Notebooks, SQL Scripts, Stored Procedures, and more. Pipelines themselves have minimal data transformation capabilities. Data pipelines support on-premises data gateways and Virtual Network (VNet) data gateways. Note that while pipelines themselves are considered a low code/no code tool, they can execute Notebooks and SQL scripts that are code-first tools.

Copy Job which was added to Data Factory in 2025 enables quick and easy data movement with minimal learning curve for new users. While there is a Copy Activity available in a data pipeline, Microsoft introduced the stand-alone copy job for those who needed a quick option without going through the full learning curve of data pipelines. Copy Job is super powerful and supports incremental data loads.

Apache Airflow Job enables you to define your data orchestration using Directed Acyclic Graphs (DAGs) in Python. It is meant for those who are already familiar with Apache Airflow or prefer a code-first approach to data orchestration.

> **TIP**
>
> If you are not familiar with Apache Airflow, it is an open source tool created to programmatically create, schedule and monitor workflows. It integrates with many third party projects including Fabric and is one the most robust tools used by Data Engineers for data orchestration. See *https://airflow.apache.org/* for more information.

Data Warehouse

Fabric Data Warehouse is a high-performance, lake-centric SQL solution designed for enterprise-scale analytics. It requires minimal setup and offers automatic workload optimization, *ACID* (Atomicity, Consistency, Isolation, Durability) transaction compliance, and T-SQL support, including full DDL (Data Definition Language) and DML (Data Modification Language) operations.

Even though Fabric Warehouse may look like the traditional on-premises SQL Server, it is quite different. Fabric Warehouse uses massively parallel processing (MPP) to handle large workloads efficiently while separating storage from compute to maximize scalability.

It stores data in Delta/Parquet open-source format within OneLake. This allows other Fabric workloads to access the data without the need to duplicate it. Storing data in an open-standard format guarantees no vendor lock and enables 3rd party tool integration.

Fabric Warehouse is optimized for structured and semi-structured data, and it provides an intuitive easy-to-use experience for both beginner and experienced data professionals.

There are several options to load data into a warehouse. You can populate a warehouse with Pipelines, Dataflows, or SQL commands such as CREATE TABLE AS (CTAS), SELECT INTO, and COPY INTO.

Figure 2-5 shows different options you can use to load data into a Fabric Warehouse.

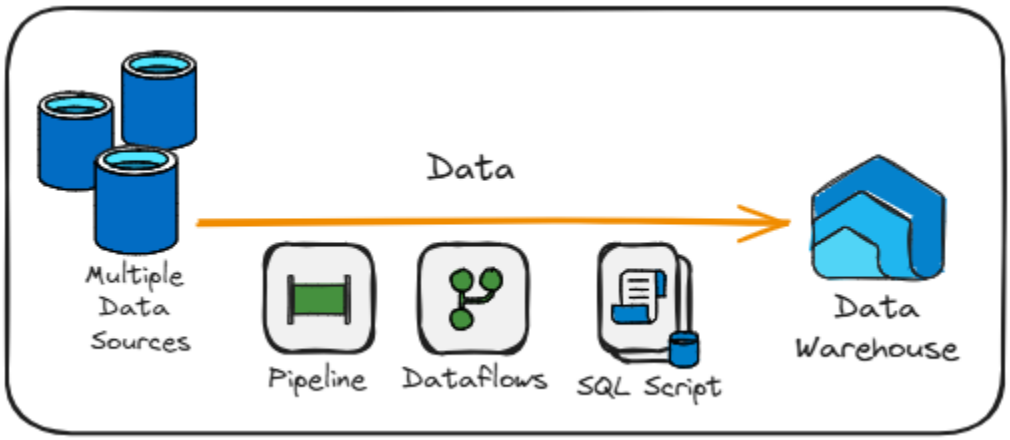

Figure 2-5. Various options for loading data into a Warehouse

A Data Warehouse includes robust security features to protect sensitive data, offering both high-level and granular access control such as object/column/row level security.

Finally, each Warehouse comes with a default Power BI semantic model for further data analysis.

You will learn more about Data Warehouse in Chapter 7, “ Working with the warehouse in MS Fabric”.

Data Engineering

Data Engineering provides a powerful, code-first environment for building and managing large-scale data transformations and analysis. Data Engineering is centered around a Lakehouse. It uses Apache Spark, notebooks, and Delta Lake to support advanced data exploration and cleansing.

The target users of Data Engineering are those who prefer to work with code-first tools and work with languages such as Python, SQL, Scala, and R to ingest, transform, and prepare data.

Data Engineering has three main items in Fabric: **Lakehouse**, **Notebook**, and **Spark Job Definition**.

Figure 2-6 shows some of the main items that are commonly used when you work as a Data Engineer.

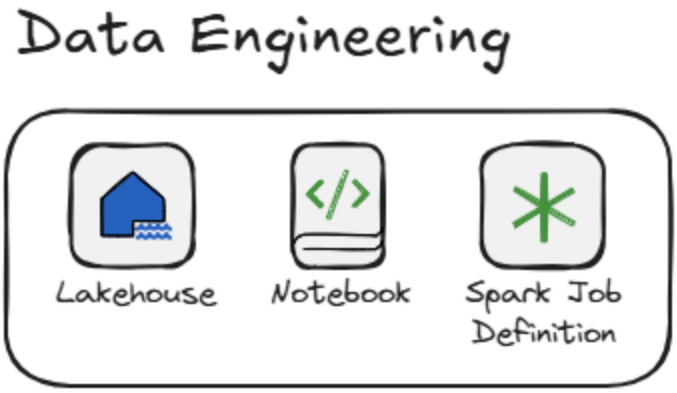

Figure 2-6. The Fabric items commonly used in Data Engineering

Let's briefly look at the three main Fabric items.

Lakehouse is a modern way to store and work with data that resides in OneLake. It combines the best parts of data lakes and data warehouses. It makes it easier to manage and analyze both structured data (like tables) and unstructured data (like text files or images) at scale.

While data lakes are great for storing lots of different kinds of data, most users find it easier to work with structured data when creating reports or running queries. For example, tabular data (organized in rows and columns) is much easier to search and analyze than raw text files.

Fabric Lakehouse helps with this by converting raw data into Delta tables, which can be easily accessed using languages like SQL. It also supports ACID (Atomicity, Consistency, Isolation, Durability) transaction compliance which ensures that data remains accurate and reliable when it's updated or changed.

Fabric provides a built-in Lakehouse explorer that allows data engineers to interact with the lakehouse. Notebooks, pipelines and dataflows, and Apache Spark Jobs are other mechanisms to interact with a lakehouse.

Each lakehouse comes with a SQL Analytics Endpoint and a default Power BI semantic model for further data analysis.

Notebooks are a web-based interactive coding tool used for data ingestion, data preparation and transformation, exploratory analysis, machine learning, and reporting. They are formatted into cells of code or text. This gives them an excellent interactive structured format. Fabric notebooks support languages such as Python, SQL, Scala, and R.You can schedule Notebooks to run independently or as part of a pipeline.

Apache Spark Jobs are designed for high-performance, automated data processing. They are pieces of code that allow you to submit batch or streaming jobs to Spark clusters where tasks are split across multiple nodes using distributed computing. Unlike interactive notebooks, which allow you to run code step by step, display results, and make adjustments, Spark jobs execute in a single run without manual interaction from you..

You will learn more about Data Engineering and Lakehouse later in this book.

Data Science

Data Science capabilities in Fabric allow Data Scientists to enrich organizational data by running machine learning models and saving the resulting predictive data into OneLake, which analysts in turn can seamlessly include in their reports.

Figure 2-7 shows some of the main Fabric items that Data Scientists use the most.



Figure 2-7. Fabric items used by Data Scientists

Data Scientists can access organizational data stored in OneLake through Lakehouse and Notebooks to explore, visualize, and prepare data. Within the notebooks, they can use PySpark/Python, Scala, SparkR/SparklyR, as well as many powerful open-source visualization libraries. They also have access to a notebook based tool called Data Wrangler for data exploration and preparation using Python.

Databases

Fabric SQL Database is a developer-friendly, fully transactional database designed to support OLTP (Online Transaction Processing) workloads. In contrast, Fabric Warehouse and Lakehouse are optimized for OLAP (Online Analytical Processing) workloads, focusing on large-scale analytics.

Fabric SQL Database is built on the same code base as Azure SQL Database and wrapped in a SaaS model for simplified access and ease of use. It has features such as Automatic index creation and the Automatic Tuning enabled by default which ensure optimal performance without manual intervention.

Figure 2-8 shows some of Fabric items that are commonly used along with SQL Databases.

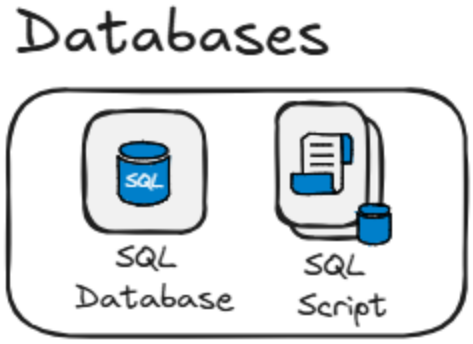

Figure 2-8. Fabric items used with SQL Databases.

Data stored in a Fabric SQL Database is automatically replicated to OneLake which makes it accessible through a read-only SQL analytics endpoint (read-only warehouse) for analytical queries. Once the data is in

OneLake, it is fully available across all Fabric workloads, enhancing interoperability.

Finally, every SQL Database in Fabric comes with a default semantic Power BI model for quick analytics.

Real-Time Intelligence

Real-Time Intelligence (RTI) in Fabric is a powerful solution for managing event-driven streaming and log data. It excels at processing structured and semi-structured data, such as JSON and text, at petabyte scale, making it the most efficient Fabric workload for handling data in motion. Unlike traditional batch processes, RTI allows data to stay in motion, enabling real-time actions without requiring immediate storage.

RTI provides a comprehensive data pipeline, covering ingestion, transformation, storage, modeling, analytics, visualization, AI-driven insights, and tracking. It consists of several key items: Eventstream, Eventhouse, KQL Database, KQL Queryset, RTI Dashboard, and Activator. Figure 2-9 shows the main items that are commonly used as part of RTI in Fabric.

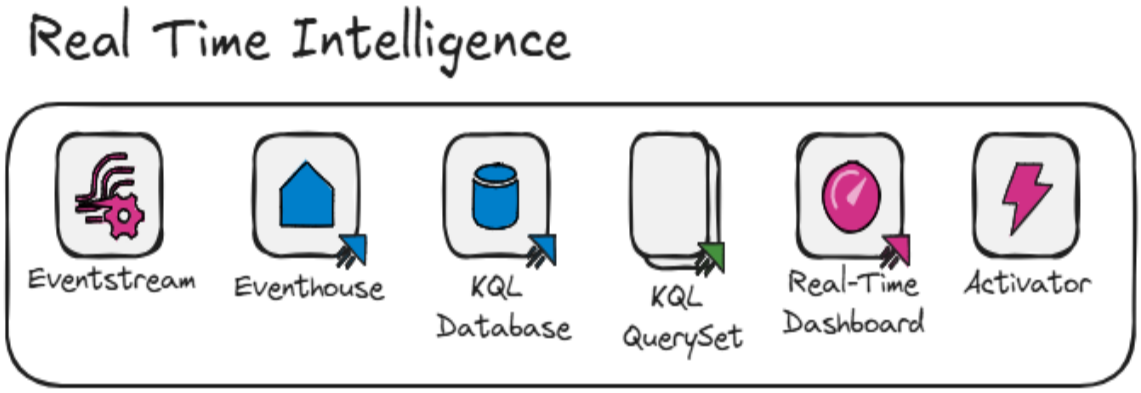

Figure 2-9. Commonly used Fabric items for RTI

> **TIP**
>
> KQL stands for Kusto Query Language. KQL is a powerful query language designed for fast and efficient data exploration on large volumes of structured, semi-structured, and unstructured data.

Let’s review the Fabric items used in RTI.

Eventstreams

Eventstrems are used to ingest, filter, and transform data in motion from various sources like IoT devices or application logs. They can send streamed data to an Eventhouse, which acts as a storage hub optimized for real-time querying and analytics. Alternatively, Eventstreams can send streamed data to a Lakehouse in Delta Parquet format.

Eventhouse

Eventhouse is a storage hub for streamed data. Instead of forcing data into traditional batch processing, Eventhouse keeps it ready for real-time querying, making it easy to analyze and respond to petabytes of data quickly. Once data is stored in Eventhouse, you can optionally[4] choose to replicate it to OneLake. In this scenario, you will also be able to access the data from OneLake with a read-only SQL analytics endpoint (read-only warehouse) or a default Power BI semantic model. This enables seamless integration with the rest of Fabric workloads.

KQL Database

KQL Database is a database within an Eventhouse. An Eventhouse can contain multiple KQL databases which can store and analyze real-time streaming data.

KQL Queryset

KQL Queryset is a set of predefined KQL queries. Instead of writing queries from scratch, you can use these to speed up searches and

analysis.

Real Time Intelligence Dashboard

Real Time Intelligence Dashboard is where you can visualize and track their real-time data! The dashboard provides intuitive charts and live updates, making it easy to monitor trends and performance at a glance.

Activator

Activator enables automated actions based on real-time data insights, helping organizations respond instantly to critical events. For example, if an unusual pattern is detected or certain thresholds are reached, Activator can trigger alerts or workflows without waiting for manual intervention.

You'll get to learn more about Real-Time Intelligence later in the book.

Power BI

Power BI has been repeatedly recognized as a leader in the Gartner's Magic Quadrant for Analytics and Business Intelligence Platforms.

Power BI empowers you to create semantic models from data, enabling powerful analytical reporting. When Fabric workloads are enabled in a tenant, Power BI provides business analysts and stakeholders seamless access to all Fabric data. With innovations such as Direct Lake storage mode which delivers blazing fast data, Power BI continues to lead the way in modern analytics. Figure 2-10 shows Power BI items.

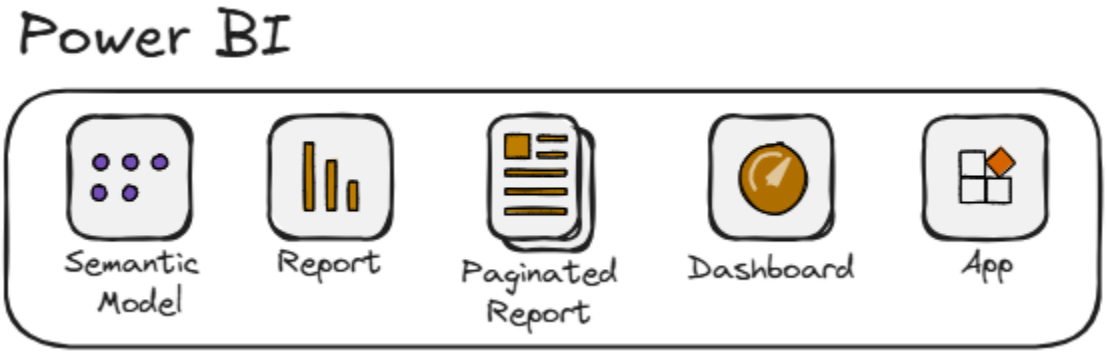

Figure 2-10. Power BI items

The main components of Power BI are Semantic Models, Reports, Paginated Reports, Dashboards, and Apps. Let’s look at each one.

Semantic Models

Semantic Models are the foundation of Power BI, providing a structured way to connect and organize data. Power BI modelers typically use Power BI Desktop to define relationships between multiple data tables and create calculations that enhance business reporting and insights. By establishing clear connections between data sources, semantic models make it easy for users to explore, analyze, and visualize data without requiring deep technical expertise.

Reports

Reports are Interactive data presentations that bring insights to life with charts and graphs, allowing users to filter and drill down into key business metrics.

Paginated Reports

Paginated Reports are printer-friendly, formatted reports for structured documents like invoices or financial statements. Unlike interactive reports, paginated reports are designed for precise page layouts and can be exported to PDF or printed.

Dashboards

Dashboards provide a high-level overview of your most important metrics, pulling insights from multiple reports into a single, easy-to-read visual display. Dashboards help users track performance and make quick, informed decisions.

Apps

Apps are collections of reports that make it simple to share insights with teams.

You'll get to dive deeper into Power BI in Chapter 10, " Understanding Power BI core components."

Industry Solutions and Partner Workloads

Industry Solutions in Fabric provide prebuilt templates for specific industries like manufacturing, retail, healthcare, etc. For example, healthcare organizations can use Fabric to transform patient data, while retailers can leverage it for inventory optimization and sales forecasting. The industry solutions themselves are usually free to access and deploy, but you will be charged for the underlying Fabric resources they use, such as compute, storage, and capacity.

Partner Workloads in Fabric allow independent software vendors (ISVs) and Microsoft partners to build custom solutions on Fabric. Partners can interoperate, develop, or extend Fabric with specialized workloads, helping businesses accelerate transformation and enhance data-driven decision-making. As time goes by, more and more Partner Workloads are becoming available in Fabric.

CoPilot

Copilot is an AI powered assistant that helps you get things done faster by writing code, creating dataflows, summarizing reports, or even building visuals, all from plain natural human language instructions.

In 2025, Microsoft made Copilot available in all paid Fabric capacities starting from F2. This change is helping drive its adoption by more users who use it in tools including Power BI, Data Factory, and Notebooks. You will learn more about CoPilots and AI capabilities in Fabric in Chapter 19, “AI Analytics Engineer”.

Now that you learned about the main components of Fabric, let’s check out how the pricing works.

Fabric’s Cost Model

Microsoft Fabric pricing includes four areas: compute, user licensing, storage, and a fourth bucket we loosely define as the other (catchall) category.

Let’s start with Compute pricing. As you learned earlier in this chapter, in Fabric, compute is measured in capacity power or CUs. Fabric capacities have either a fixed monthly cost or a Pay as You go price model. Let us start by explaining the fixed monthly pricing.

Fixed Monthly Capacity Pricing

In the fixed monthly price model, you pay the same amount each month depending on the size of the capacity (CUs). The type and number of Fabric workloads that you run on a capacity, does not change the monthly price. Of course, as your workloads increase over time, you may eventually decide to move to a higher capacity which will have a higher monthly price. Fabric Admins can adjust compute capacity by scaling up and down.

For example, if you purchase an F64 (equivalent to a P1 Power BI capacity), you get 8 capacity units (CUs) and up to 25 GB of memory allowance for each Power BI semantic model in Power BI service. This capacity will be used by all your reports and users. The monthly cost is a fixed amount no matter how many people access your reports, how often they access them, or how complex your reports are. At some point, you may

add more users and Fabric workloads so that an F64 may not meet your Compute needs anymore. In that case, you can scale up to F128.

Fabric capacities start with F2 which only starts at about $263 per month[5]. The cost is even lower if you get a yearly reservation. With reservations you commit to paying a fixed monthly price for the duration of a year. For F2, this brings down the monthly price to around $156 per month.

> **TIP**
>
> For the most recent Fabric capacity pricing see this link: *https://azure.microsoft.com/en-us/pricing/details/microsoft-fabric*.

Pay as You Go Capacity Pricing

In addition to the fixed monthly capacity pricing we explained above, Fabric offers a flexible pay as you go pricing model. This pricing model provides yet another level of flexibility that makes it more feasible for many more organizations to use Fabric. This pricing model provides an hourly cost for a specific CU. For example, an F2 will cost $0.36 per hour. With the Pay-as-You go model, you can pause a Fabric capacity when you don't need it. Even when you pause a Fabric Capacity, you are still charged for storage which is for the data stored in OneLake.

Storage Costs

All Fabric items store their data in OneLake. You pay for each gigabyte of data stored in OneLake. This is independent of the Capacity (Compute) price. Currently, OneLake pricing starts at $0.023 per GB. The storage for Power BI semantic models is part of the capacity pricing and is not billed under OneLake storage.

User Licenses

User licenses are only required for working with Power BI content under certain circumstances as we will explain shortly. The rest of Fabric items are covered under the capacity model.

Power BI Developers need a Power BI Pro license to publish their reports to any Power BI workspace in Power BI service. This is independent of which type of capacity a workspace is part of. On the other hand, when it comes to consuming Power BI reports and dashboards, the capacity level defines whether or not the end users also need a Power BI Pro license. If a workspace is hosted on an F64 (P1) or higher capacity, then the consumers don't need any Power BI pro licenses. If a workspace is hosted on an F32 or lower, every consumer who accesses a Power BI report or dashboard must also have a Pro Power BI license. We will explain Power BI licenses in more detail in chapter 10.

Other costs

Mirroring is a technology in Fabric that allows for seamless and continuous data replication from outside resources. Each Fabric capacity comes with a predefined amount Mirrored storage that is included in the capacity price. For example, an F2 comes with 2TB of included storage for mirroring. An F256 comes with 256 TB of included storage for mirroring. Any storage beyond the allotted included mirroring limit for each capacity, will be billed at regular OneLake pricing.

Networking charges may be added in the future. Microsoft has committed to giving a 90 days notice before these charges take effect. Network charges apply to data transfers from outside sources to Fabric (Ingress) and from Fabric to outside sources (Egress). You can learn more about Azure Ingress and Egress charges here.

Getting Started with Fabric

By this point, you are likely eager to see what Microsoft Fabric looks like in action! You can get to Fabric's portal by navigating to *https://app.fabric.microsoft.com/*. When you visit this link for the first time, you are asked to provide your email. If you don't already have a work account with access to Microsoft Fabric, you can create a free Fabric account and start a free 60 day trial of Microsoft Fabric. Figure 2-11 shows the dialog window you get when you navigate to the portal for the first time.

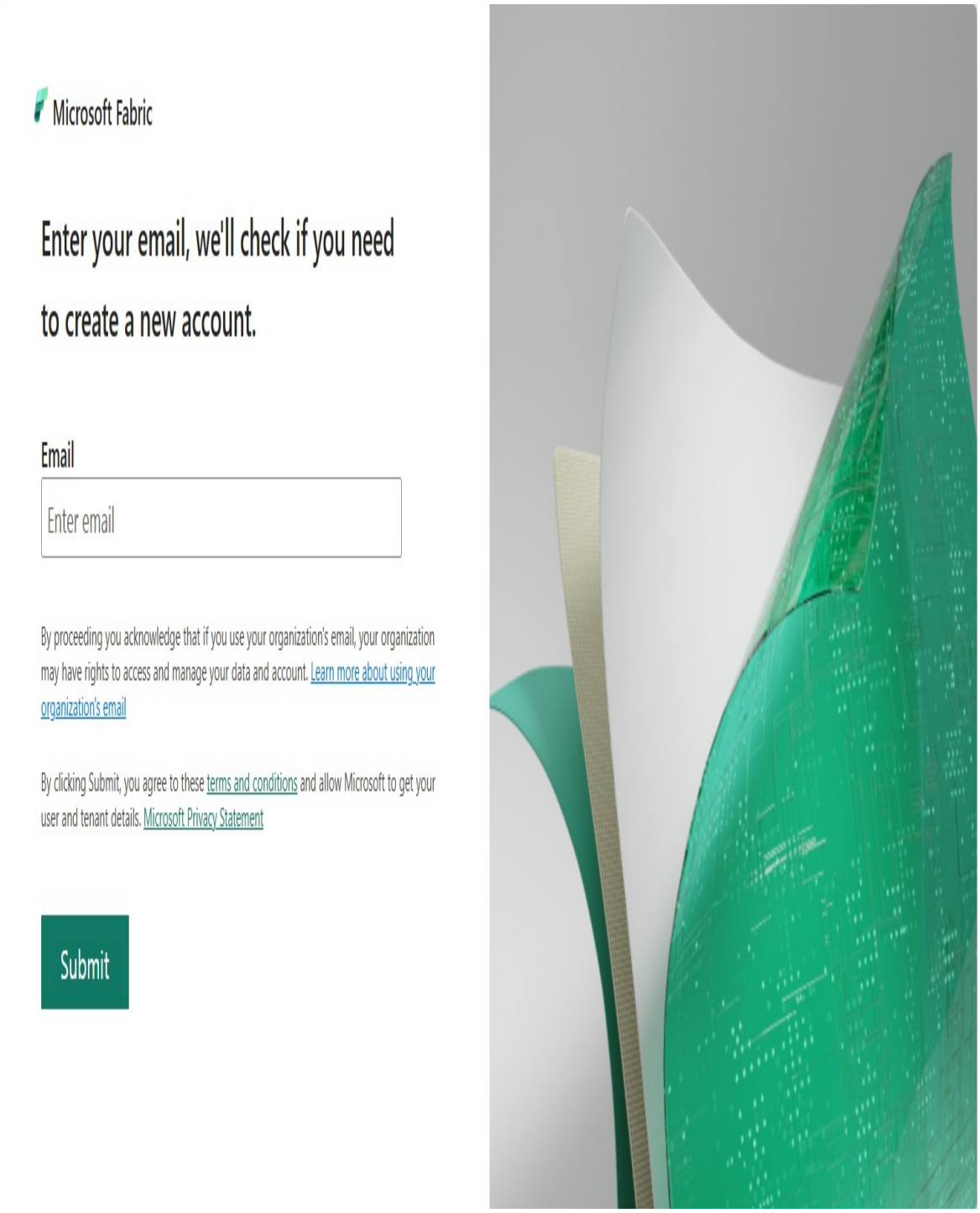

Figure 2-11. Signing into Fabric for the first time

If you sign up for the free Fabric trial, you'll get access to resources roughly equivalent to an F64 capacity, with some limitations. For example, the trial capacity does not give you access to features such as trusted workspace access, private links, CoPilots, or free Mirroring storage. Even with these

limitations, the trial capacity is an amazing opportunity to get started with Fabric for free for 60 days. Individuals often use the trial to learn the platform and keep up with technology, while organizations use the trial to train their teams or test Fabric's capabilities with real data and use cases.

You will get started with some hands-on practices to ingest data into Fabric in Chapter 5, "Data ingestion and data access in MS Fabric".

Following in Power BI's successful footsteps

It is worth taking a few moments and considering the factors that led to the huge success of Power BI given that Microsoft Fabric has adopted the same SaaS model and inherited many of its foundations such as the concepts of capacities and workspaces from Power BI. We think the reason for Power BI's massive success was three fold.

The first part was that Power BI Desktop was provided as a free tool to anyone who wanted to use it. You do not need a license to do analytics on the Desktop. As we explained previously, you only need a paid license if you want to publish your report to Power BI Service to share with other people. This empowered massive numbers of people to try it out for free before they had to pull out their credit cards or get budget approvals at work. By the time the decision makers saw what Power BI Desktop could produce, they were more than willing to invest in the product. We believe that the free Fabric Trial capacity will serve a similar purpose in that it allows organizations to try the product for themselves for free and witness its massive capabilities and powers.

The second part was that Power BI Service, which is the web based portal where you ideally go to share your reports with a wide enterprise level audience, was implemented and provided as a simple SaaS model. That means that once you have a Power BI report, you would just publish it to Power BI service and you do not have to worry about setting up services or servers to share the report. Microsoft takes care of all the infrastructure, provisioning, security, and network for you. This ease of use and efficient pattern was sometimes referred to as "five seconds to sign-up; five minutes

to WOW!". Now with Fabric, you have the same ease of use and agility to get started with the rest of Fabric items.

Finally, the third success factor was the use of bundling compute, memory, and storage resources into Power BI capacities. Power BI capacities simplified pricing and purchasing. Within each capacity, you can have as many Power BI reports and models as you like. As you learned previously in this chapter, Fabric offers a similar capacity pricing model that not only simplest purchasing and pricing but also allows for multiple workloads to combine under the same set of resources. This will result in cost savings vs. if you had to pay for each service separately.

Let's use an analogy to clarify the relationship between Power BI and Microsoft Fabric. As we mentioned earlier, Fabric was built on the foundation of Power BI. The relationship between Power BI and Microsoft Fabric is often described as the relationship between Microsoft Word and Microsoft Office. Some users only use Word out of the Office family of products; while others may use several such as Excel, PowerPoint. No matter which one of the Office tools you use, you have a familiar user interface.

The same is true for Power BI and Fabric. You can continue to only use Power BI features. You don't have to use all the rest of Fabric capabilities. In fact, for existing Power BI customers, a Fabric admin must enable a "Users can create Fabric items" setting in Power BI's tenant settings before users are allowed to create the rest of Fabric items.

The good news is that if you are already familiar with Power BI, you are going to find Fabric's interface and concepts very intuitive. On the other hand, if you have not used Power BI before, getting started with Fabric is not difficult and that is exactly what we are going to help you with in this book!

Summary

In this chapter you learned about the fundamental concepts of Microsoft Fabric and its core components. Fabric provides a comprehensive set of tools that support data ingestion, transformations, orchestration, data science, data engineering, massively parallel processing (MPP) warehouses and lakehouses, real-time intelligence, OLTP databases, analytical reporting, dashboards, and more.

All of these capabilities are implemented with a Software as a Service (SaaS) model with an easy-to-use interface and a straightforward pricing model. Fabric workloads work with data stored in OneLake in an open-standard format which makes it possible for an unlimited third-party tools to integrate with your data in Fabric.

Fabric is continuously evolving, with frequent updates and enhancements. To stay informed about new features and planned developments, please refer to the Fabric roadmap.

In the next chapter, you will dive deeper into Data ingestion and data access.

---

1 Microsoft Entra is Microsoft's cloud-based identity and access management system. Entra was previously called Azure Active Directory (Azure AD).

2 Microsoft has announced the end of life for the Power BI premium capacities (P1 and above). For full details see this announcement from Microsoft.

3 In this book we refer to Dataflows Gen2 in Fabric as Dataflows. At the time of writing this book, these are still called Dataflows Gen2 in Fabric's interface. We expect them to be renamed to Dataflows in near future.

4 To learn more about replicating data from Eventhouse to OneLake, please refer to this link.

5 Prices listed in this section are in US Dollars for the Central US region. Fabric prices may vary based on your region, your purchase agreement type with Microsoft, date of purchase, and currency exchange rates.

OceanofPDF.com