# Fundamentals of Data Engineering - Reis & Housley

## Part I. Foundation and Building Blocks

* 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 + training did not address: **Data *collection*, *cleansing*, *access*, *transformation*, and *infrastructure***
* **These are problems that data engineering aims to solve**
* This book aims to fill a gap in current data engineering (DE) content + materials and connect the dots of the end-to-end **data lifecycle**
* While there’s no shortage of technical resources that address specific DE tools + tech, **people struggle to understand *how* to assemble these components into a coherent whole that applies in the real world**
* We will see how to stitch together various technologies to serve the needs of downstream data consumers such as analysts, data scientists, + MLEs
* Big idea of this book = the **DE lifecycle: data generation, storage, ingestion, transformation, + serving**
* Since the dawn of data, we’ve seen the rise + fall of innumerable specific technologies + vendor products, but the **DE lifecycle stages have remained essentially unchanged**
* With this framework, a reader will come away with a sound understanding for applying technologies to real-world business problems
* **Goal** = to map out principles that reach across 2 axes:
* **1) Wish to distill DE into *principles* that can encompass *any relevant technology***
* **2) Wish to present principles that will stand the test of time**
* **NOTE: Focus is on a cloud-first approach**
* **Cloud** = a fundamentally transformative development that **will endure for decades**
* Most on-prem data systems and workloads will eventually move to cloud hosting.
* **Assume that infrastructure and systems are ephemeral** **+** **scalable, + that DE’s will lean toward deploying managed services in the cloud**
* Most concepts in this book *will* translate to non-cloud environments
* Primary intended audience = technical practitioners, mid-to-senior-level SWE’s, data scientists, analysts interested in moving into DE, or DE’s working in the guts of specific technologies but wanting to develop a more comprehensive perspective
* Secondary target audience = data stakeholders who work adjacent to technical practitioners (e.g., a data team lead with a technical background overseeing a team of DE’s, or a director of data warehousing wanting to migrate from on-prem technology to a cloud-based solution)
* This book will help you **weave a complete picture of DE *across technologies + paradigms***
* By the end of this book you will understand:
* **How DE impacts your current role (data scientist, SWE, or data team lead)**
* **How to cut through marketing hype + choose the right tech, data architecture, + processes**
* **How to use the DE lifecycle to design + build a robust architecture**
* **Best practices for each stage of the data lifecycle**
* And you will be able to:
* **Incorporate DE principles in your role (data scientist, analyst, SWE, data team lead, etc.)**
* **Stitch together a *variety* of cloud technologies to serve needs of downstream data consumers**
* **Assess DE problems with an end-to-end framework of best practices**
* **Incorporate data governance and security across the DE lifecycle**

### Chapter 1 - Data Engineering Described

* If you work in data or software, you may have noticed DE emerging from the shadows + now sharing the stage with data science
* **DE** = one of the hottest fields in data and tech, + for a good reason 🡪 **builds the foundation for data science and analytics in production**

#### What Is Data Engineering?

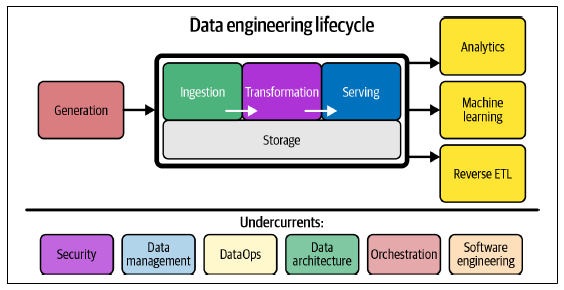
* Despite current popularity of DE, **there’s a lot of confusion about what DE means + what DE’s do**
* DE = existed in *some* form since companies started doing things w/ data (predictive + descriptive analytics, reports, etc.) + came into sharp focus alongside the rise of data science in the 2010s
* Endless definitions of DE exist
* Early 2022 Google exact-match search for “what is data engineering?” = 91,000+ unique results
* Few examples of how some experts in the field define it:
* From “*Data Engineering and Its Main Concepts*” by AlexSoft: DE is a **set of operations aimed at creating interfaces + mechanisms for the flow + access of information**
* It takes dedicated specialists (DE’s) to **maintain data so that it remains available + usable by others**
* In short, **DE’s set up + operate the organization’s data infrastructure, preparing it for further analysis by data analysts + scientists**
* Jesse Anderson, “*The Two Types of Data Engineering*”:
* The **first type of DE is SQL-focused**
* **The work + primary storage of the data is in RDB’s**
* **All the data processing is done w/ SQL or a SQL-based language**
* Sometimes, this data processing is done with **an ETL tool**
* The **second type of DE is Big Data-focused**
* The **work + primary storage of the data is in Big Data technologies** like Hadoop, Cassandra, HBase, etc.
* **All the data processing is done in Big Data frameworks** like MapReduce, Spark, Flink
* **While SQL is used, the *primary processing is done w/ programming languages*** like Java, Scala, + Python
* Lewis Gavin “*What Is Data Engineering*“: DE is all about the **movement, manipulation, + management of data**
* Maxime Beauchemin “The Rise of the Data Engineer“: In relation to previously existing roles, the DE field could be thought of as **a superset of BI + DW-ing that brings more elements from SWE**
* This discipline **also integrates specialization around the operation of so-called “big data” distributed systems**, along w/ **concepts around the extended Hadoop ecosystem, stream processing, and in computation at scale**
* That’s only a handful of definitions, containing an enormous range of opinions about the meaning of DE

##### Data Engineering Defined

* An obvious pattern emerges: **a DE gets data, stores it, + prepares it for consumption by data scientists, analysts, + others**
* **DE** **= the development, implementation, + maintenance of systems + processes that take in raw data + produce high-quality, consistent information that supports downstream use cases, such as analysis + ML**
* **Intersection of security, data management, DataOps, data architecture, orchestration, + SWE**
* A **data engineer****manages the DE lifecycle, beginning w/ getting data from source systems + ending w/ serving data for use cases, such as analysis or ML**

##### The Data Engineering Lifecycle

* It is all too **easy to fixate on tech and miss the bigger picture myopically**
* **Data Engineering Lifecycle gives DE’s the holistic context to view their role:**



* DE lifecycle shifts the conversation away from tech + ***toward the data itself* + the end goals that it must serve**
* **Stages** of the DE lifecycle: **Generation**, **Storage**, **Ingestion**, **Transformation**, and **Serving**
* DE lifecycle also has a notion of **undercurrents** **(critical ideas across the entire lifecycle**): **security**, **data management**, **DataOps**, **data architecture**, **orchestration**, and **SWE**
* DE lifecycle is essential to the definition of DE

##### Evolution of the Data Engineer

* **Understanding DE today and tomorrow requires a context of how the field evolved**
* **A common theme constantly reappears: what’s old is new again.**

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

* Birth of the DE arguably has its **roots in data warehousing**, dating as far back as the ‘70s, w/ the **business data warehouse**taking shape in the ‘80s + Inmon officially coining the term **data warehouse (DW)**in 1989
* IBM developed the RDB + SQL, + Oracle popularized the technology
* **As nascent data systems grew, businesses needed dedicated tools + data pipelines for reporting and BI**
* To help people **correctly model business logic in a DW, Kimball + Inmon developed respective eponymous data-modeling techniques + approaches, which are *still widely used today***
* **Data warehousing** ushered in the first age of **scalable analytics**, w/ **new massively parallel processing (MPP) databases** that **use multiple processors to crunch large amounts of data coming on the market + supporting unprecedented volumes of data**
* Roles such as BI engineer, ETL developer, + DW engineer addressed the various needs of the DW
* **DW + BI engineering were a precursor to today’s data engineering + still play a central role in the discipline**
* Internet went mainstream around the mid-1990s, creating a whole new generation of web-first companies (AOL, Yahoo, Amazon, etc.)
* Dot-com boom spawned a ton of activity in web apps + their supporting backend systems (servers, databases, storage, etc.)
* Much of the infrastructure was expensive, monolithic, + heavily licensed
* 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

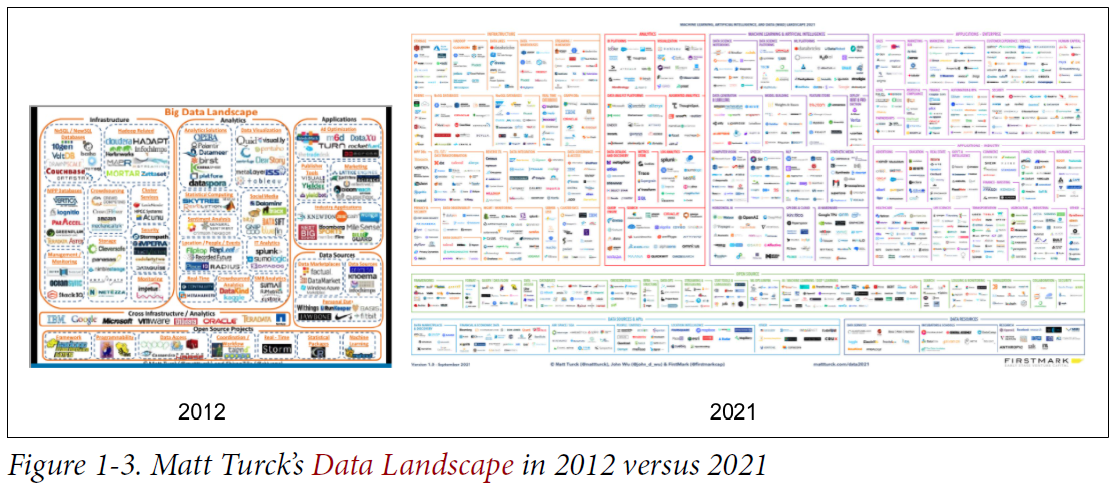
* Early 2000s: dot-com boom of the late ‘90s went bust, leaving behind a tiny cluster of survivors.
* Some survivors (Yahoo, Google, Amazon, etc.) would grow into powerhouse tech companies.
* **Initially, these companies continued to rely on the traditional, monolithic RDB’s and DW’s** of the ‘90s, pushing these systems to the limit
* As **systems buckled, updated approaches were needed to handle data growth**
* The new generation of the systems must be **cost-effective, scalable, available, + reliable**
* **Coinciding with the explosion of data, commodity hardware (such as servers, RAM, disks, + flash drives) also became cheap + ubiquitous**
* Several innovations allowed **distributed computation** and **storage on massive computing clusters** at **a vast scale**
* These **innovations started decentralizing + breaking apart traditionally monolithic services**
* “Big data” era had begun.
* “Big Data” = extremely large data sets that may be analyzed computationally to reveal patterns, trends, + associations, especially relating to human behavior + interactions
* The 3 V’s of data: **velocity, variety, and volume**
* Google had a 2003 paper on the **Google File System**, + in 2004 a paper on **MapReduce, an ultra-scalable data-processing paradigm**
* Big data has earlier antecedents in MPP DW’s + data management for experimental physics projects, but Google’s publications constituted a “big bang” for data technologies + the cultural roots of DE as we know it today
* Google papers inspired engineers at Yahoo to develop + later open-source Apache Hadoop in 2006
* **Hard to overstate the impact of Hadoop**
* SWE’s interested in large-scale data problems were drawn to the possibilities of this new open-source tech ecosystem
* As **companies of all sizes + types saw their data grow into many TB + even PB, the era of the big data engineer was born**
* Around the same time, Amazon had to keep up w/ its own exploding data needs + 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), + many other core data building blocks
* **Amazon elected to offer these services for internal + external consumption through AWS, becoming the 1st popular public cloud**
* AWS **created an ultra-flexible pay-as-you-go resource marketplace by virtualizing + reselling vast pools of commodity hardware**
* **Instead of purchasing hardware for a data center, developers could simply rent compute + storage from AWS**
* AWS became a highly profitable growth engine for Amazon, + other public clouds would soon follow (Google Cloud, Microsoft Azure, DigitalOcean, etc.)
* **The public cloud is arguably one of the most significant innovations of the 21st century + spawned a revolution in the way software + data applications are developed + deployed**
* The **early big data tools + public cloud laid the foundation for today’s data ecosystem**
* The modern data landscape (+ DE as we know it now) would not exist w/out these innovations

###### The 2000s and 2010s: Big data engineering

* Open-source big data tools in the Hadoop ecosystem rapidly matured + 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**
* **Engineers could choose the latest + greatest** (Hadoop, Apache Pig, Apache Hive, Dremel, Apache HBase, Apache Storm, Apache Cassandra, Apache Spark, Presto, + numerous other new tech) that came on the scene
* Traditional enterprise-oriented + GUI-based data tools suddenly felt outmoded, + **code-first engineering was in vogue with the ascendance of MapReduce**
* The explosion of data tools in the late 2000s and 2010s ushered in the **big data engineer**
* To effectively use these tools + techniques (namely, the Hadoop ecosystem including Hadoop, YARN, HDFS, + MapReduce), **big data engineers had to be proficient in SWE + low-level infrastructure hacking, but w/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 🡪 a buzzword that gained popularity during the early ‘00s through the mid-‘10s + captured the imagination of companies trying to make sense of the ever-growing volumes of data + endless barrage of marketing from companies selling big data tools + services
* Immense hype = common to see **companies using big data tools for small data problems**, sometimes standing up a Hadoop cluster to process just a few GB
* Seemed like everyone wanted in on the big data action.
* **Big data has lost steam due to simplification**
* Despite the power + sophistication of open-source big data tools, **managing them was a lot of work + required constant attention**
* Often, companies employed entire *teams* of big data engineers, costing millions of dollars a year, to babysit these platforms + often **spend excessive time maintaining complicated tooling + arguably not as much time delivering the business’s insights + value**
* **Open-source developers, clouds, + 3rd-parties started looking for ways to abstract, simplify, + make big data available *w/out the high admin overhead + cost of managing their clusters, and installing, configuring, + upgrading their open-source code***
* “Big data” =essentially a relic to describe a particular time + approach to handling large amounts of data
* Today, data is moving faster than ever + 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

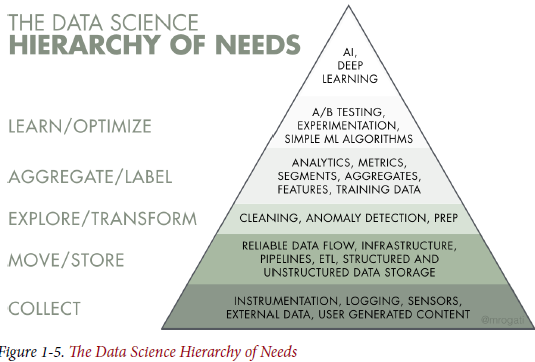
* **DE role is evolving rapidly for the foreseeable future**
* Whereas DE’s **historically tended to the low-level details of monolithic frameworks** such as Hadoop, Spark, or Informatica, the **trend is moving toward decentralized, modularized, managed, and highly abstracted tools**
* Data **tools have proliferated at an astonishing rate**:



* Popular trends in the early 2020s include the **modern data stack**, representing **a collection of off-the-shelf open source + 3rd-party products assembled to make analysts’ lives easier**
* At the **same time, data sources + data formats are growing both in variety + size**
* **DE is increasingly a discipline of interoperation + connecting various technologies** like LEGO bricks, **to serve ultimate business goals**
* DE can be described more precisely as a **data lifecycle engineer**
* **With greater abstraction + simplification, a data lifecycle engineer is no longer encumbered by the gory details of yesterday’s big data frameworks**
* **While DE’s maintain skills in low-level data programming + use these as required, they increasingly find their role focused on things *higher in the value chain*: security, data management, DataOps, data architecture, orchestration, + general data lifecycle management**
* As tools + workflows simplify, we’ve seen a **noticeable shift in the attitudes of DE’s:**
* Instead of focusing on who has the “biggest data,” **open-source projects + services are increasingly concerned w/ managing + governing data, making it easier to use + discover, + improving its quality**
* Now conversant in acronyms like CCPAand GDPR, + **as** **they engineer pipelines, they concern themselves w/ privacy, anonymization, data garbage collection, + compliance w/ regulations**
* ***What’s old is new again***
* While “enterprise-y” stuff like data management (including data quality + governance) was common for large enterprises in 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, + packaged, technologists + entrepreneurs have shifted focus back to the “enterprise-y” stuff, but w/ an emphasis on decentralization and agility, which contrasts w/ the traditional enterprise command-and-control approach**
* The present (2022-2023) = a golden age of data lifecycle management
* **DE’s managing the DE lifecycle have better tools + techniques than ever before**

##### Data Engineering and Data Science

* *Where does DE fit in with data science?*
* Some argue DE is a subdiscipline of data science, but some say **DE is *separate* from data science and analytics** (They ***complement* each other, but they are distinctly different**)
* **DE sits *upstream* from data science** (**provide the inputs used by data scientists**, who are downstream from DE + **convert these inputs into something useful**)



* Although many data scientists are eager to build + tune ML models, the joke is that an estimated 70%-80% of their time is spent toiling in the bottom 3 parts of the hierarchy (gathering, cleaning, + processing data) + only a tiny slice of their time on analysis + ML
* Many argue that **companies need to build a solid data foundation (bottom 3 levels of the hierarchy) before tackling areas such as AI and ML**
* **Data scientists aren’t typically trained to engineer production-grade data systems, + they end up doing this work haphazardly b/c they lack the support + resources of a DE**
* In an ideal world, data scientists should spend > 90% of their time focused on the top layers of the pyramid: **analytics, experimentation, + ML**
* **When DE focus on these bottom parts of the hierarchy, they build a solid foundation for data scientists to succeed**
* With data science driving advanced analytics + ML, **DE straddles the divide between getting data and getting value from data**
* **DE is of equal importance + visibility to data science, w/ DE’s engineers playing a vital role in making data science successful in production**

#### Data Engineering Skills and Activities

* The skill set of a DE encompasses the “**undercurrents**” of DE: **security, data management, DataOps, data architecture, + SWE**
* **This skill set requires an understanding of how to evaluate data tools + how they fit together across the DE lifecycle**
* It’s **also critical to know *how* data is produced in source systems + how analysts + data scientists will consume + create value after processing + curating data**
* Finally, a **DE juggles a lot of complex moving parts + must constantly optimize along the axes of cost, agility, scalability, simplicity, reuse, + interoperability**
* **Recent past = a DE was expected to know + understand how to use a small handful of powerful + monolithic technologies** (Hadoop, Spark, Teradata, Hive, + many others) to create a data solution
* Utilizing these technologies often requires a sophisticated understanding of SWE, networking, distributed computing, storage, or other low-level details
* Work would be devoted to cluster admin + maintenance, managing overhead, + writing pipeline + transformation jobs, among other tasks
* ***Nowadays*, the data-tooling landscape is dramatically less complicated to manage + deploy**
* **Modern data tools considerably abstract + simplify workflows**
* As a result, **DE’s are now focused on balancing the simplest + most cost-effective, best-of-breed services that deliver value to the business**
* The DE is **also expected to create agile data architectures that evolve as new trends emerge**
* What are some things a data engineer does *NOT* do?
* Typically, **does not directly build ML models, create reports or dashboards, perform data analysis, build KPIs, or develop software applications**
* But a **DE *should* have a good functioning understanding of such areas in order to serve stakeholders best**

##### Data Maturity and the Data Engineer

* **Level of DE complexity w/in a company depends a great deal on a company’s data maturity**
* This **significantly impacts a DE’s day-to-day job responsibilities + *career progression***
* **Data maturity**= **the progression toward higher data utilization, capabilities, + integration across the organization**
* **But does *NOT* simply depend on the age or revenue of a company**
* *An early-stage startup can have greater data maturity than a 100-year-old company w/ annual revenues in the billions*
* **What matters is the way data is leveraged as a competitive advantage**
* Data maturity models have many versions, such as **Data Management Maturity (DMM)** + others, + it’s hard to pick one that is both simple + useful for DE
* We’ll create our own **simplified data maturity model with** **3 stages**: **1)** **starting with data**, **2)** **scaling with data**, and **3) leading with data**

###### Data Maturity Model Stage 1: Starting with data

* **A company getting started with data is, by definition, in the very early stages of its data maturity**
* The company may have fuzzy, loosely defined goals or no goals, data architecture + infrastructure are in the very early stages of planning + development, adoption + utilization are likely low or nonexistent, the data team is small, often w/ a headcount in the single digits
* **At this stage, a DE is usually a generalist + will typically play several other roles, such as data scientist or SWE**
* A DE’s **goal here is to move fast, get traction, + add value**
* The **practicalities of getting value from data are typically poorly understood, but the desire exists**
* **Reports or analyses lack formal structure, + most requests for data are ad hoc**
* While it’s **tempting to jump headfirst into ML at this stage, don’t do it**
* Countless **data teams get stuck + fall short when they try to jump to ML w/out building a solid data foundation** (*not* to say you *can’t* get wins from ML at this stage (it is rare but possible))
* **W/out a solid data foundation, you likely won’t have the data to train reliable ML models nor the means to deploy these models to production in a scalable + repeatable way**
* A **DE should focus on the following in organizations getting started with data:**
* **1) Get buy-in from key stakeholders, including executive management**
* Ideally, the DE should have a **sponsor** for critical initiatives to design + build a data architecture to support the company’s goals
* **2) Define the right data architecture**
* Usually solo, since a **data architect** likely isn’t available
* This means **determining business goals + the competitive advantage you’re aiming to achieve w/ your data initiative**
* Work toward a data architecture that supports these goals
* **3) Identify + audit data that will support key initiatives + operate w/in the data architecture you designed**
* **4) Build a solid data foundation for *future* data analysts + scientists to generate reports + models that provide competitive value**
* In the meantime, you *may also have to generate these reports + models until a team is hired*
* This is a **delicate stage with lots of pitfalls**
* Tips for this stage:
* **Organizational willpower may wane if a lot of visible successes don’t occur w/ data**
* Getting **quick wins will establish the importance of data w/in the organization**
* Just keep in mind that **quick wins will likely create technical debt**
* **Have a plan to reduce this debt, as it will otherwise add friction for future delivery**
* **Talk to people, avoid working in silos**
* Often see the data team working in a bubble, not communicating w/ people outside their departments + getting perspectives + feedback from business stakeholders
* The **danger is you’ll spend a lot of time working on things of little use to people**
* **Avoid undifferentiated heavy lifting**
* **Don’t box yourself in with unnecessary technical complexity**
* **Use off-the-shelf, turnkey solutions wherever possible**
* ***Build custom solutions + code only where this creates a competitive advantage***

###### Data Maturity Model Stage 2: Scaling with data

* At this point, **a company has moved away from ad hoc data requests + has formal data practices**
* Now the **challenge is creating *scalable* data architectures + planning for a future where the company is *genuinely* data-driven**
* **DE roles move from generalists to *specialists*, focusing on particular aspects of the DE lifecycle**
* **Stage 2 of data maturity DE goals**:
* **Establish formal data practices**
* **Create scalable + robust data architectures**
* **Adopt DevOps + DataOps practices**
* **Build systems that support ML**
* ***Continue to avoid undifferentiated heavy lifting + customize only when a competitive advantage results***
* Issues to watch out for include:
* As we grow more sophisticated w/ data, **there’s a temptation to adopt bleeding-edge tech based on social proof from Silicon Valley, which is *rarely a good use of time + energy***
* **Any technology decisions should be driven by the value they’ll deliver to your customers**
* The **main bottleneck for scaling is not cluster nodes, storage, or technology but the DE team**
* **Focus on solutions that are simple to deploy + manage to expand your team’s throughput**
* **Tempted to frame yourself as a technologist, a data genius who can deliver magical products**
* **Shift focus instead to pragmatic leadership + begin transitioning to the next maturity stage**
* **Communicate w/ other teams about the practical utility of data**
* **Teach the organization how to consume + leverage data**

###### Data Maturity Model Stage 3: Leading with data

* At this stage, the **company is data-driven*: Automated* pipelines + systems created by DE’s allow people w/in the company to do self-service analytics + ML**
* **Introducing new data sources is seamless**, + **tangible value is derived**
* **DE’s implement proper controls + practices to ensure data is always available to people + systems**
* **DE roles continue to specialize more deeply than in stage 2**
* Stage **3 of data maturity DE will continue building on prior stages, plus they will do the following**:
* **Create automation for the seamless introduction + usage of new data**
* **Focus on building custom tools + systems that leverage data as a competitive advantage**
* **Focus on the “enterprise-y” aspects of data, such as data management (including data governance + quality) and DataOps**
* **Deploy tools that expose + disseminate data throughout the organization, including data catalogs, data lineage tools, + metadata management systems**
* **Collaborate efficiently w/ SWE’s, MLE’s, analysts, + others**
* **Create a community + environment where people can collaborate + speak openly, no matter their role or position**
* Issues to watch out for:
* At this stage, **complacency is a significant danger**
* **Orgs. must constantly focus on maintenance + improvement or risk falling to a lower stage**
* **Technology distractions** are a more significant danger here than in the other stages
* **Temptations to pursue expensive hobby projects that don’t deliver value to the business**
* ***AGAIN, utilize custom-built technology ONLY where it provides a competitive advantage***

##### The Background Skills of a Data Engineer

* DE = a fast-growing field, and a lot of questions remain about *how to become one*
* Because DE is a relatively new discipline, **little formal training is available to enter the field**
* Universities don’t have a standard DE path + although a handful of boot camps + online tutorials cover random topics, **a common curriculum for the subject doesn’t yet exist**
* People entering DE arrive with varying backgrounds in education, career, and skill set
* **Everyone entering the field should expect to invest a significant amount of time in self-study**
* **Transition = easiest when moving from an adjacent field, (SWE, ETL development, DBA, data science or analysis)**
* These disciplines tend to be “data aware” + provide good context for data roles in an org.
* They also equip folks w/ the relevant technical skills + context to solve DE problems
* Despite the lack of a formalized path, **a requisite body of knowledge exists that a data engineer *should* know to be successful**
* By definition, a DE **must understand *both* data + tech**
* W/ respect to **data**: entails knowing about **various best practices around data management**
* W/ respect to **tech**: must be aware of **various tool options, + their interplay + trade-offs**
* This requires **a good understanding of SWE, DataOps, + data architecture**
* Zooming out, a DE must **also understand the requirements of data consumers (data analysts and data scientists) and the broader implications of data across the organization**
* **DE = *holistic* practice (best DE’s view their responsibilities through business *and* technical lenses)**

##### Business Responsibilities

* The following **macro-responsibilities** aren’t exclusive to DE’s but are **crucial for anyone working in a data or tech field**
* **1) Know how to communicate w/ non-technical *and* technical people.**
* **Communication is key** = **need to be able to establish rapport + trust w/ people across an org.**
* **Pay close attention to organizational hierarchies, who reports to whom, how people interact, and which silos exist**, as **these observations will be invaluable to your success**
* **2) Understand how to scope + gather business *and* product requirements.**
* **Need to know *what* to build + ensure that stakeholders agree w/ your assessment**
* In addition, **develop a sense of how data + technology decisions impact the business**
* **3) Understand the cultural foundations of Agile, DevOps, + DataOps**
* Many technologists mistakenly believe these practices are solved through technology, + *this is dangerously wrong*
* **Agile, DevOps, + DataOps = fundamentally *cultural*, requiring buy-in across the organization**
* **4) Control costs**
* **Successful = you can keep costs low while providing outsized value**
* Know how to **optimize for time to value**, the **total cost of ownership**, + **opportunity cost**
* Learn to **monitor costs to avoid surprises**
* **5) Learn continuously**
* The data field feels like it’s changing at light speed
* **People who succeed are great at picking up new things *while sharpening their fundamental knowledge***
* They’re also good at **filtering** **= determining which new developments are most relevant to their work, which are still immature, + which are just fads**
* Stay abreast of the field and ***learn how to learn***
* **A successful DE always zooms out to understand the big picture + how to achieve outsized value for the business**
* **Communication is vital**, both for technical + non-technical people
* Often see data teams succeed based on their communication w/ other stakeholders, as **success or failure is rarely a tech issue**
* **Knowing how to navigate an organization, scope + gather requirements, control costs, + continuously learn will set you apart from DE’s who rely solely on technical abilities to carry their career**

##### Technical Responsibilities

* You **must understand how to build architectures that optimize performance + cost at a high level, using prepackaged or homegrown components**
* Ultimately, **architectures + constituent technologies are building blocks to serve the DE lifecycle**
* Recall the **Stages of the DE lifecycle**: **Generation, Storage, Ingestion, Transformation, Serving**
* **Undercurrents of the DE lifecycle**: **Security, Data management, DataOps, Data architecture, Orchestration, SWE**
* **Should a DE know how to code? Short answer: yes** 🡪 Should have ***production-grade* SWE chops**
* The **nature of SWE projects undertaken by DE’s has changed fundamentally in the last few years**
* Fully managed services now replace a great deal of low-level programming effort previously expected of engineers, who now use managed, open-source, simple plug-and-play SaaS offerings
* Ex: DE’s now focus on high-level abstractions or writing pipelines as code w/in an **orchestration framework**
* **Even in a more abstract world, SWE best practices provide a competitive advantage, + DE’s who can dive into deep architectural details of a codebase give companies an edge when specific technical needs arise**
* A ***DE who can’t write production-grade code will be severely hindered*, + we don’t see this changing anytime soon**
* **DE remain SWE’s, in addition to their many other roles**
* ***What languages should a data engineer know?***
* **Primary languages of DE = SQL, Python, a Java Virtual Machine (JVM) language (usually Java or Scala), + bash:**
* **SQL** = most common interface for databases + data lakes
* After briefly being sidelined by the need to write custom MapReduce code for big data processing, **SQL (*in various forms*) has reemerged as the lingua franca of data**
* The Unreasonable Effectiveness of SQL
* The advent of MapReduce + the big data era relegated SQL to passe status
* Since then, **various developments have dramatically enhanced the utility of SQL in the DE lifecycle**
* **Spark SQL, Google BigQuery, Snowflake, Hive**, **+ many other data tools** can process massive amounts of data via declarative, **set-theoretic SQL semantics**
* SQL is **also supported by many streaming frameworks** (Flink, Beam, Kafka, etc.)
* **Competent DE’s should be highly proficient in SQL**
* **NOT a be-all + end-all language**
* **A powerful tool that can quickly solve complex analytics + data transformation problems**
* Given that **time is a primary constraint for DE team throughput, engineers should embrace tools that combine simplicity + high productivity**
* DE’s also do well to **develop expertise in *composing SQL w/ other operations***
* Either within frameworks such as Spark and Flink or by using orchestration to combine multiple tools
* DE’s should also **learn modern SQL semantics for dealing with JSON parsing + nested data, + consider leveraging a SQL management framework such as dbt**
* A **proficient DE also recognizes when SQL is *NOT* the right tool for the job + can choose + code in a suitable alternative**
* A SQL expert could likely write a query to stem + tokenize raw text in an NLP pipeline, but would also recognize coding in native Spark is a far superior alternative
* **Python** = **bridge language between DE and data science**
* A growing number of DE tools are **written in Python *or* have Python APIs**
* “2nd-best language at everything” that underlies popular data tools such as pandas, NumPy, Airflow, sci-kit learn, TensorFlow, PyTorch, and PySpark
* **Python is the glue between underlying components + is frequently a first-class API language for interfacing w/ a framework**
* **JVM languages such as Java and Scala**
* Prevalent for Apache open-source projects such as Spark, Hive, Druid, etc.
* **JVM is generally more performant than Python + may provide access to lower-level features than a Python API** (ex: This is the case for Apache Spark and Beam)
* Understanding Java/Scala = **beneficial if using a popular open-source data framework**
* **bash** = the CLI for Linux OS’s
* **Knowing bash commands + being comfortable using CLIs will significantly improve productivity + workflow when you need to script or perform OS operations**
* Even today, DE frequently use CLI tools like `awk` or `sed` to process files in a data pipeline or call bash commands from orchestration frameworks
* **If using Windows, feel free to substitute PowerShell for bash**
* DE’s may also need to develop proficiency in **secondary programming languages**: including **R, JavaScript, Go, Rust, C/C++, C#,** and **Julia**
* **Developing in these languages is often necessary when popular across the company or used with domain-specific data tools**
* Ex: JavaScript has proven popular as a language for user-defined functions in cloud data warehouses
* At the same time, C# and PowerShell = essential in companies that leverage Azure + the Microsoft ecosystem
* **Keeping Pace in a Fast-Moving Field**
* *How do you keep skills sharp in a rapidly changing field like DE? Should you focus on the latest tools or deep dive into fundamentals?*
* **Focus on the fundamentals to understand what’s *NOT* going to change** and **pay attention to ongoing developments to know where the field *IS* going**
* New paradigms + practices are introduced all the time, + it’s **incumbent on *you* to stay current**
* **Strive to understand how new technologies will be helpful in the lifecycle**

##### The Continuum of Data Engineering Roles, From A to B

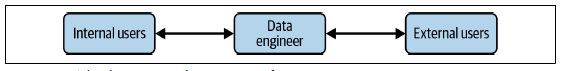
* Although job descriptions paint a DE as a “unicorn” who must possess every data skill imaginable, they **don’t all do the same type of work or have the same skill set**
* **Data maturity is a helpful guide to understanding the types of data challenges a company will face as it grows its data capability**
* It’s beneficial to look at some **critical distinctions in the kinds of work DE do**
* Though these distinctions are simplistic, they clarify what data scientists and DE’s do, + avoid lumping either role into the unicorn bucket
* In data science, there’s the notion of **type A and type B data scientists**
* **Type A (for “analysis”) data scientists**: focus on **understanding + deriving insight from data**
* **Type B (for “build”) data scientists**: share similar backgrounds as type A data scientists + **possess strong programming skills** (build systems that make data science work in production)
* Borrowing from this data scientist continuum:
* **Type A (for “abstraction”) data engineers: avoids undifferentiated heavy lifting, keeping data architecture as abstract + straightforward as possible, and not reinventing the wheel**
* Manages DE lifecycle mainly **via** **entirely off-the-shelf products, managed services, + tools**
* **Works at companies across industries + at all levels of data maturity**
* **Type B (for “build”) data engineers: build data tools + systems that scale + leverage a company’s core competency + competitive advantage**
* More **commonly found at stage 2 or 3 data maturity companies** (scaling or leading w/ data)
* *Or* when **an initial data use case is so unique + mission-critical that custom data tools are required to get started**
* **Type A and type B DE’s may work in the same company and may even be the same person**
* More **commonly**, a **type A** data engineer is **first hired to set the foundation**, with **type B** data engineer **skill sets either learned or hired as the need arises** w/in a company

#### Data Engineers Inside an Organization

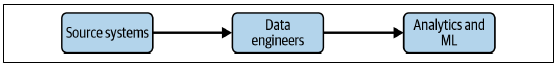
* **DE’s don’t work in a vacuum: *Depending on what they’re working on*, they’ll interact w/ technical *and* non-technical people + face different directions (internal *and* external)**

##### Internal vs. External-Facing Data Engineers

* A DE **serves several end users + faces many internal + external directions**
* Since **not all DE workloads + responsibilities are the same**, it’s **essential to understand *whom* the DE serves**
* ***Depending on the end-use cases*, a DE’s primary responsibilities = external-facing, internal-facing, or a *blend* of the two (2 sets of users with *very* different requirements for query concurrency, security, + more):**



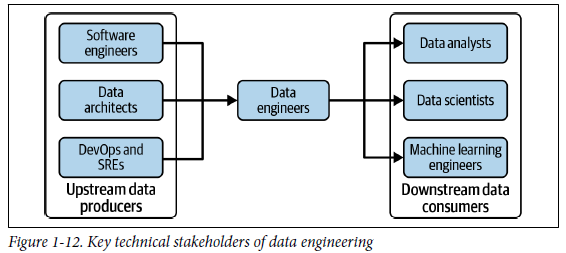
* **External-facing DEtypically aligns w/ the users of external-facing applications (social media apps, IoT devices, e-commerce platforms, etc.)**
* **Architects, builds, + manages systems that collect, store, + process transactional data *and* event data from these applications**
* **Systems** built by these DE’s have a **feedback loop** **from the application used by external users to the data pipeline, + then back to the application**
* ***External-facing*** DE comes with a unique **set of problems**
* **External-facing query engines often handle much larger concurrency loads than internal-facing systems**
* DE’s also need to **consider putting tight limits on queries that users can run to limit the infrastructure impact of any single user**
* Also, **security = much more complex + sensitive problem for external queries, especially if data being queried is multitenant (from many customers + housed in a single table)**



* **Internal-facing DE typically focuses on activities crucial to the needs of the business + internal stakeholders**
* Ex: **Creating + maintaining data pipelines + DW’s** for BI dashboards, reports, business processes, data science, + ML models, etc.
* **External-facing and internal-facing responsibilities are often blended**
* **In practice, internal-facing data is usually a prerequisite to external-facing data**

##### Data Engineers and Other Technical Roles

* **In practice**, the **DE lifecycle cuts across many domains of responsibility**
* **DE’s sit at the nexus of various roles, directly or through managers, interacting w/ many organizational units**
* Let’s look at **whom a DE may impact, *in terms of technical roles***



* The **DE = a hub between** **data producers** (SWE’s, data architects, + DevOps or site-reliability engineers (SREs), etc.) **and** **data consumers** (data analysts, data scientists, MLE’s, etc.)
* In addition, **DE’s will interact with those in operational roles, such as DevOps engineers**
* *Given the pace at which new data roles come into vogue (analytics and MLE’s come to mind), this is by no means an exhaustive list*

###### Upstream stakeholders

* To be successful as a DE, you ***need* to understand the data architecture you’re using or designing + the source systems producing the data you’ll need**
* A few familiar **upstream stakeholders: data architects, SWE’s, + DevOps engineers/SRE’s**
* **Data architects**
* **Function at a level of abstraction one step removed from DE’s**
* **Design the blueprint for organizational data management, mapping out processes + overall data architecture + systems**
* <https://www.dataversity.net/data-architect-vs-data-engineer/>
* Also serve as a **bridge between an organization’s technical + non-technical sides**
* Successful data architects generally have “battle scars” from **extensive engineering experience, allowing them to guide + assist engineers while successfully communicating engineering challenges to non-technical business stakeholders**
* **Implement policies for managing data across silos and business units, steer global strategies such as data management and data governance, + guide significant initiatives**
* **Often play a central role in cloud migrations and greenfield (i.e., ground-up) cloud design**
* **Advent of the cloud has shifted the boundary between data architecture and DE**
* **Cloud data architectures are much more fluid than on-prem systems**, so **architecture decisions that traditionally involved extensive study, long lead times, purchase contracts, + hardware installation are *now often made during the implementation process*, just one step in a larger strategy**
* Nevertheless, **data architects will remain influential visionaries in enterprises, working hand in hand with DE’s to determine the big picture of architecture practices and data strategies**
* **Depending on a company’s data maturity + size, a DE may overlap w/ or assume the responsibilities of a data architect**
* So, a **DE should have a good understanding of architecture best practices + approaches**
* **Note:** We are talking aboutdata architects in the ***upstream* stakeholders**section
* Data **architects** **often help design application data layers that are source systems for DE’s**
* **May also interact w/ DE at various other stages of the DE lifecycle**
* **SWE’s**
* **Build the software + systems that run a business + are largely responsible for generating the internal datathat DE’s will consume and process**
* The **systems built by SWE’s typically generate application event data + logs, which are significant assets in their own right**
* Internal data **contrasts w/ external datapulled from SaaS platforms or partner businesses**
* **DE’s must work closely with the SWE’s**
* **In well-run technical organizations, SWE’s and DE’s coordinate *from the inception* of a new project to design application data for consumption by analytics + ML applications**
* **DE’s should work together with SWE’s to understand the applications that generate data, the volume, frequency, + format of the generated data, + anything else that will impact the DE lifecycle, such as data security + regulatory compliance**
* **Might mean setting upstream expectations on what data SWE’s need to do their jobs**
* **DevOps engineers and site-reliability engineers (SRE’s)**
* **Often produce data through operational monitoring**
* We classify them as upstream of DE’s, but they **may also be downstream, consuming data through dashboards or interacting w/ DE’s directly in coordinating operations of data systems**

###### Downstream stakeholders

* **DE exists to serve downstream data consumers + use cases**
* **Data scientists**
* **Build forward-looking models to make predictions + recommendations, which are then evaluated on live data to provide value in various ways**
* Ex: Model scoring might determine automated actions in response to real-time conditions, recommend products to customers based on the browsing history in their current session, or make live economic predictions used by traders
* Common industry folklore: data scientists spend 70%-80% of their time collecting, cleaning, + preparing data, but **these numbers often reflect immature data science + DE practices**
* **Many popular data science frameworks become bottlenecks if not scaled up appropriately**
* Data scientists who **work exclusively on a single workstation** force themselves to **downsample** data, **making data prep significantly more complicated + potentially compromising the quality of the models they produce**
* Furthermore, **locally-developed code + environments are often difficult to deploy in production, + a lack of automation significantly hampers data science workflows**
* If **DE’s do their job + collaborate successfully, data scientists shouldn’t spend their time collecting, cleaning, + preparing data after initial EDA**
* **DE’s should automate this work as much as possible**
* The **need for production-ready data science is a significant driver behind the emergence of the entire DE profession**
* DE’s should **help data scientists to enable a path to production**
* **DE’s work to provide the data automation and scale that make data science more efficient**
* **Data (or Business) analysts**
* **Seek to understand business performance + trends**
* Whereas **data scientists are *forward*-looking**, **data analysts typically focus on past or present**
* Usually **run SQL queries in a DW or a data lake, + may also utilize spreadsheets for computation + analysis along with various BI tools** (PowerBI, Looker, Tableau, etc.)
* **Domain experts in the data they work w/ frequently + will become intimately familiar w/ data definitions, characteristics, + quality problems**
* **Their typical downstream customers are business users, management, + executives**
* **DE’s work w/ data analysts to build pipelines for new data sources required by the business**
* **Data analysts’ subject-matter expertise (SME) is invaluable in improving data quality**, + they frequently collaborate w/ DE’s in this capacity
* **MLE’s**
* **Overlap with DE’s and data scientists**
* **Develop advanced ML techniques, train models, and design + maintain the infrastructure running ML processes in a scaled production environment**
* Often have **advanced working knowledge of ML + DL techniques + frameworks** (PyTorch, TensorFlow, etc.)
* Also **understand the hardware, services, + systems required to run these frameworks, both for model training + model deployment at a production scale**
* **Common for ML flows to run in a cloud environment where MLE’s can spin up + scale infrastructure resources on demand or rely on managed services**
* ***Boundaries between MLE, DE, + data science are blurry***
* DE’s may have some operational responsibilities over ML systems, + data scientists may work closely with MLE in designing advanced ML processes
* The **MLE world is snowballing + parallels a lot of the same developments occurring in DE**
* Whereas several years ago, the **attention of ML was focused on how to build models, MLE now increasingly emphasizes incorporating best practices of MLOps + other mature practices previously adopted in SWE + DevOps**
* **AI researchers**
* **Work on new, advanced ML techniques**
* May work inside large tech companies, specialized intellectual property startups (OpenAI, DeepMind, etc.), or academic institutions
* Some are dedicated to part-time research in conjunction with MLE responsibilities inside a company
* Those working inside specialized ML labs are often 100% dedicated to research
* **Research problems may target immediate practical applications or more abstract demonstrations of AI**
* ML research projects examples: DALL-E, Gato AI, AlphaGo, and GPT-3/GPT-4
* Given pace of advancements in ML, these will very likely be quaint in a few years’ time
* AI researchers in well-funded organizations are highly specialized + operate w/ supporting teams of engineers to facilitate their work
* MLE’s in academia usually have fewer resources but rely on teams of graduate students, postdocs, + university staff to provide engineering support
* MLE’s who are partially dedicated to research often rely on the same support teams for research and production

##### Data Engineers and Business Leadership

* Discussed technical roles w/ which a DE interacts, but **DE’s also operate more broadly as organizational connectors, often in a non-technical capacity**
* **Businesses have come to rely increasingly on data as a core part of many of their products (*or as a product itself*)**
* **DE’s will now**:
* **Participate in strategic planning + lead key initiatives that extend beyond the boundaries of IT**
* **Often support data architects by acting as glue between the business + data science/analytics**
* **Data in the C-suite**
* **C-level executives = increasingly involved in data + analytics, as these are recognized as significant assets for modern businesses**
* Ex:CEOs now concern themselves w/ initiatives that were once the exclusive province of IT, such as cloud migrations or deployment of a new customer data platform
* **CEOs**:
* CEO’s at *non*-tech companies generally don’t concern themselves w/ the nitty-gritty of data frameworks + software
* Instead, they **define a vision in collaboration with technical C-suite roles + company data leadership**
* **DE’s provide a window into what’s possible w/ data, + DE’s + their managers maintain a map of what data is available to the organization (both internally + from 3rd-parties) in what time frame**
* DE’s are **also tasked to study primary data architectural changes in collaboration w/ other engineering roles**
* Ex: DE’s are **often heavily involved in cloud migrations, migrations to new data systems, or deployment of streaming technologies**
* **CIO**
* Senior C-suite executive **responsible for IT within an organization**, an **internal-facing role**. A
* **Must possess deep knowledge of IT *and* business processes, either alone is insufficient**
* **Direct the IT organization, setting ongoing policies while also defining + executing significant initiatives** under the direction of the CEO
* **Often collaborate w/ DE leadership in organizations with a well-developed data culture**
* If not very high in data maturity, a CIO typically **helps shape organization’s** **data culture**
* Will **work w/ engineers + architects to map out major initiatives + make strategic decisions on adopting major architectural elements**
* Such as **enterprise resource planning (ERP)** + **customer relationship management (CRM) systems**, **cloud migrations**, **data systems**, + **internal-facing IT**
* **CTO**
* **Similar to a CIO but faces *outward***
* **Owns the key technological strategy + architectures for *external*-facing applications**
* Such as mobile, web apps, + IoT (all critical data sources for DE’s)
* **Likely a skilled technologist w/ a good sense of SWE fundamentals + system architecture**
* **In some organizations *without a CIO*, CTO (or sometimes the COO) plays the role of CIO**
* **DE’s often report directly or indirectly through a CTO**
* **Chief Data Officer (CDO)**
* Capital One 2002: **created** **to recognize the growing importance of data as a business asset**
* **Responsible for a company’s data assets + strategy**
* **Focused on data’s business utility but should have a strong technical grounding**
* **Oversee data products, strategy, initiatives, + core functions such as master data management (MDM) + privacy**
* **Occasionally**, CDOs **manage business analytics + DE**
* **Chief Analytics Officer (CAO)**
* **Variant of the CDO role**
* At places where *both* roles exist, **CDO focuses on the tech + organization required to *deliver* data**, while **CAO is responsible for analytics, strategy, + decision-making for the business**
* ***May* oversee data science + ML**, though this *largely depends on whether the company has a CDO or CTO role*
* **Chief Algorithms Officer (CAO-2)**
* A recent innovation in the C-suite
* **Highly technical role focused specifically on data science + ML**
* **Typically have experience as individual contributors + team leads in data science or ML projects, + frequently have a background in ML research and a related advanced degree**
* Expected to be **conversant in current ML research + have deep technical knowledge of their company’s ML initiatives**
* In addition to creating business initiatives, they **provide technical leadership, set research and development agendas, + build research teams**
* **Data engineers and *Project* Managers (PM’s)**
* **DE’s often work on significant initiatives, potentially spanning many years**
* As of 2021-2023, many **DE’s are now working on cloud migrations (migrating pipelines + warehouses to the next generation of data tools)**
* Other DE’s are starting **greenfield projects** **(assembling new data architectures from scratch by selecting from an astonishing number of best-of-breed architecture + tooling options)**
* Such **large initiatives will often benefit from** ***project* management**(in contrast to *product* management)
* Whereas **DE’s function in an infrastructure + service delivery capacity**, a **PM directs traffic + serves as a gatekeeper**
* Most PM’s operate according to some variation of **Agile and Scrum, w/ Waterfall still appearing occasionally**
* **Business never sleeps, + business stakeholders often have a significant backlog of things they want to address + new initiatives they want to launch**
* **PM’s must filter a long list of requests + prioritize critical deliverables to keep projects on track + better serve the company**
* **DE’s interact w/ PM’s, often planning sprints for projects + ensuing standups related to sprints**
* **Feedback goes both ways**
* **DE’s informing PM’s + other stakeholders about progress + blockers**
* **PM’s balancing cadence of tech teams against the ever-changing needs of the business**
* **Data engineers and *Product* Managers**
* ***Product* managers oversee product development**, often **owning product lines**
* In the context of DE’s, “**data products**” **= either built from the ground up or are incremental improvements upon existing products**
* **DE’s interact more frequently w/ product managersas the corporate world has adopted a data-centric focus**
* ***Like PM’s*, product managers balance the activity of tech teams against the needs of the customer + business**
* **Data engineers + other management roles**
* De’s interact with various managers beyond project + product managers.
* However, **these interactions usually follow either the services or cross-functional models**
* DE’s either serve a variety of incoming requests as a centralized team or work as a resource assigned to a particular manager, project, or product.
* More information on data teams and how to structure them
* John Thompson’s Building Analytics Teams: <https://www.packtpub.com/product/building-analytics-teams/9781800203167>
* Jesse Anderson’s Data Teams: <https://www.amazon.com/Data-Teams-Management-Successful-Data-Focused/dp/1484262271>
* Both books provide strong frameworks + perspectives on the roles of execs with data, who to hire, + how to construct the most effective data team for a company
* **Companies don’t hire engineers simply to hack on code in isolation.**
* To be worthy of their title, **DE’s should develop a deep understanding of the problems they’re tasked w/ solving, the tech tools at their disposal, + the people they work with + serve**