O'REILLY®

Compliments of Qubole

Operationalizing the Data Lake

Building and Extracting Value from Data Lakes with a Cloud-Native Data Platform

Holden Ackerman
& Jon King

Operationalizing the Data Lake

Building and Extracting Value from Data Lakes with a Cloud-Native Data Platform

Holden Ackerman and Jon King

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Operationalizing the Data Lake
by Holden Ackerman and Jon King

Printed in the United States of America.

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

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

Editor: Nicole Tache
Production Editor: Deborah Baker
Copyeditor: Octal Publishing, LLC
Proofreader: Christina Edwards

Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Rebecca Demarest

June 2019: First Edition

Revision History for the First Edition
2019-04-29: First Release

This work is part of a collaboration between O'Reilly and Qubole. See our *statement of editorial independence*.

978-1-492-04948-7

[LSI]

Table of Contents

Acknowledgments

In a world in which data has become the new oil for companies, building a company that can be driven by data and has the ability to scale with it has become more important than ever to remain competitive and ahead of the curve. Although many approaches to building a successful data operation are often highly customized to the company, its data, and the users working with it, this book aims to put together the data platform jigsaw puzzle, both pragmatically and theoretically, based on the experiences of multiple people working on data teams managing large-scale workloads across use cases, systems, and industries.

We cannot thank Ashish Thusoo and Joydeep Sen Sarma enough for inspiring the content of this book following the tremendous success of *Creating a Data-Driven Enterprise with DataOps* (O'Reilly, 2017) and for encouraging us to question the status quo every day. As the cofounders of Qubole, your vision of centralizing a data platform around the cloud data lake has been an incredible eye-opener, illuminating the true impact that information can have for a company when done right and made useful for its people. Thank you immensely to Kay Lawton as well, for managing the entire book from birth to completion. This book would have never been completed if it weren't for your incredible skills of bringing everyone together and keeping us on our toes. Your work and coordination behind the scenes with O'Reilly and at Qubole ensured that logistics ran smoothly. Of course, a huge thank you to the Qubole Marketing leaders, Orlando De Bruce, Utpal Bhatt, and Jose Villacis, for all your considerations and help with the content and efforts in readying this book for publication.

We also want to thank the entire production team at O'Reilly, especially the dynamic duo: Nicole Tache and Alice LaPlante. Alice, the time you spent with us brainstorming, meeting with more than a dozen folks with different perspectives, and getting into the forest of diverse technologies operations related to running cloud data lakes was invaluable. Nicole, your unique viewpoint and relentless efforts to deliver quality and context have truly sculpted this book into the finely finished product that we all envisioned.

Holistically capturing the principles of our learnings has taken deep consideration and support from a number of people in all roles of the data team, from security to data science and engineering. To that effect, this book would not have happened without the incredibly insightful contributions of Pradeep Reddy, Mohit Bhatnagar, Piero Cinquegrana, Prateek Shrivastava, Drew Daniels, Akil Murali, Ben Roubicek, Mayank Ahuja, Rajat Venkatesh, and Ashish Dubey. We also wanted to give a special shout-out to our friends at Ibotta: Eric Franco, Steve Carpenter, Nathan McIntyre, and Laura Spencer. Your contributions in brainstorming, giving interviews, and editing imbued the book with true experiences and lessons that make it incredibly insightful.

Lastly, thank you to our friends and families who have supported and encouraged us month after month and through long nights as we created the book. Your support gave us the energy we needed to make it all happen.

Foreword

Today, we are rapidly moving from the information age to the age of intelligence. Artificial intelligence (AI) is quickly transforming our day-to-day lives. This age is powered by data. Any business that wants to thrive in this age has no choice but to embrace data. It has no choice but to develop the ability and agility to harness data for a wide variety of uses. This need has led to the emergence of data lakes.

A data lake is generally created without a specific purpose in mind. It includes all source data, unstructured and semi-structured, from a wide variety of data sources, which makes it much more flexible in its potential use cases. Data lakes are usually built on low-cost commodity hardware, which makes it economically viable to store terabytes or even petabytes of data.

In my opinion, the true potential of data lakes can be harnessed only through the cloud—this is why we founded Qubole in 2011. This opinion is finally being widely shared around the globe. Today, we are seeing businesses choose the cloud as the preferred home for their data lakes.

Although most initial data lakes were created on-premises, movement to the cloud is accelerating. In fact, the cloud market for data lakes is growing two to three times faster than the on-premises data lake market. According to a 2018 survey by Qubole and Dimensional Research, 73% of businesses are now performing their big data processing in the cloud, up from 58% in 2017. The shift toward the cloud is needed in part due to the ever-growing volume and diversity of data that companies are dealing with; for example, 44%

of organizations now report working with massive data lakes that are more than 100 terabytes in size.

Adoption of the cloud as the preferred infrastructure for building data lakes is being driven both by businesses that are new to data lakes and adopting the cloud for the first time as well as by organizations that had built data lakes on-premises, but now want to move their infrastructures to the cloud.

The case for building a data lake has been accepted for some years now, but why the cloud? There are three reasons for this.

First is agility. The cloud is elastic, whereas on-premises datacenters are resource-constrained. The cloud has virtually limitless resources and offers choices for adding compute and storage that are just an API call away. On the other hand, on-premises datacenters are always constrained by the physical resources: servers, storage, and networking.

Think about it: data lakes must support the ever-growing needs of organizations for data and new types of analyses. As a result, data lakes drive demand for compute and storage that is difficult to predict. The elasticity of the cloud provides a perfect infrastructure to support data lakes—more so than any on-premises datacenter.

The second reason why more data lakes are being created on the cloud than in on-premises datacenters is innovation. Most next-generation data-driven products and software are being built in the cloud—especially advanced products built around AI and machine learning. Because these products reside in the cloud, their data stays in the cloud. And because the data is in the cloud, data lakes are being deployed in the cloud. Thus, the notion of "data gravity"—that bodies of data will attract applications, services, and other data, and the larger the amount of data, the more applications, services, and other data will be attracted to it—is now working in favor of the cloud versus on-premises datacenters.

The third reason for the movement to the cloud is economies of scale. The market seems to finally realize that economics in the cloud are much more favorable when compared to on-premises infrastructures. As the cloud infrastructure industry becomes increasingly competitive, we're seeing better pricing. Even more fundamentally, the rise of cloud-native big data platforms is taking advantage of the cloud's elasticity to drive heavily efficient usage of

infrastructure through automation. This leads to better economics than on-premises data lakes, which are not nearly as efficient in how they use infrastructure.

If you combine all of these things together, you see that an on-premises infrastructure not only impedes agility, but also is an expensive choice. A cloud-based data lake, on the other hand, enables you to operationalize the data lakes at enterprise scale and at a fraction of the cost, all while taking advantage of the latest innovations.

Businesses follow one of two different strategies when building or moving their data lake in the cloud. One strategy is to use a cloud-native platform like Qubole, Amazon Web Services Elastic MapReduce, Microsoft Azure HDInsight, or Google Dataproc. The other is to try to build it themselves using open source software or through commercially supported open source distributions like Cloudera and buy or rent server capacity.

The second strategy is fraught with failures. This is because companies that follow that route aren't able to take advantage of all the automation that cloud-native platforms provide. Firms tend to blow through their budgets or fail to establish a stable and strong infrastructure.

In 2017, I published a book titled *Creating a Data-Driven Enterprise with DataOps* that talked about the need to create a DataOps culture before beginning your big data journey to the cloud. That book addressed the technological, organizational, and process aspects of creating a data-driven enterprise. A chapter in that book also put forth a case of why the cloud is the right infrastructure for building data lakes.

Today, my colleagues are continuing to explore the value of the cloud infrastructure. This book, written by cloud data lake experts Holden Ackerman and Jon King, takes that case forward and presents a more in-depth look at how to build data lakes on the cloud. I know that you'll find it useful.

— Ashish Thusoo
Cofounder and CEO, Qubole
May 2019

Introduction

Overview: Big Data's Big Journey to the Cloud

It all started with the data. There was too much of it. Too much to process in a timely manner. Too much to analyze. Too much to store cost effectively. Too much to protect. And yet the data kept coming. Something had to give.

We generate 2.5 quintillion bytes of data each day (one quintillion is one thousand quadrillion, which is one thousand trillion). A NASA mathematician puts it like this: "1 million seconds is about 11.5 days, 1 billion seconds is about 32 years, while a trillion seconds is equal to 32,000 years." This would mean one quadrillion seconds is 32 *billion* years—and 2.5 quintillion would be 2,500 times that.

After you've tried to visualize that—you can't, it's not humanly possible—keep in mind that 90% of all the data in the world was created in just the past two years.

Despite these staggering numbers, organizations are beginning to harness the value of what is now called *big data*.

Almost half of respondents to a recent McKinsey Analytics study, Analytics Comes of Age, say big data has "fundamentally changed" their business practices. According to NewVantage Partners, big data is delivering the most value to enterprises by cutting expenses (49.2%) and creating new avenues for innovation and disruption (44.3%). Almost 7 in 10 companies (69.4%) have begun using big data to create data-driven cultures, with 27.9% reporting positive results, as illustrated in Figure I-1.

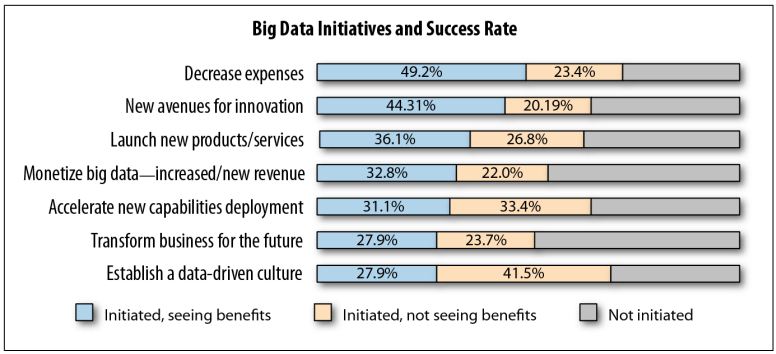

Figure I-1. The benefits of deploying big data

Overall, 27% of those surveyed indicate their big data projects are already profitable, and 45% indicate they're at a break-even stage.

What's more, the majority of big data projects these days are being deployed in the cloud. Big data stored in the cloud will reach 403 exabytes by 2021, up almost eight-fold from the 25 exabytes that was stored in 2016. Big data alone will represent 30% of data stored in datacenters by 2021, up from 18% in 2016.

My Journey to a Data Lake

The journey to a data lake is different for everyone. For me, Jon King, it was the realization that I was already on the road to implementing a data lake architecture. My company at the time was running a data warehouse architecture that housed a subset of data coming from our hundreds of MySQL servers. We began by extracting our MySQL tables to comma-separated values (CSV) format on our NetApp Filers and then loading those into the data warehouse. This data was used for business reports and ad hoc questions.

As the company grew, so did the platform. The amount, complexity, and—most important—the types of data also increased. In addition to our usual CSV-to-warehouse extract, transform, and load (ETL) conversions, we were soon ingesting billions of complex JSON-formatted events daily. Converting these JSON events to a relational database management system (RDBMS) format required significantly more ETL resources, and the schemas were always evolving based on new product releases. It was soon apparent that our data warehouse wasn't going to keep up with our product roadmap. Storage and compute limitations meant that we were having to con-

stantly decide what data we could and could not keep in the warehouse, and schema evolutions meant that we were frequently taking long maintenance outages.

At this point, we began to look at new distributed architectures that could meet the demands of our product roadmap. After looking at several open source and commercial options, we found Apache Hadoop and Hive. The nature of the Hadoop Distributed File System (HDFS) and Hive's schema-on-read enabled us to address our need for tabular data as well as our need to parse and analyze complex JSON objects and store more data than we could in the data warehouse. The ability to use Hive to dynamically parse a JSON object allowed us to meet the demands of the analytics organization.

Thus, we had a cloud data lake, which was based in Amazon Web Services (AWS). But soon thereafter, we found ourselves growing at a much faster rate, and realized that we needed a platform to help us manage the new open source tools and technologies that could handle these vast data volumes with the elasticity of the cloud while also controlling cost overruns. That led us to Qubole's cloud data platform—and my journey became much more interesting.

A Quick History Lesson on Big Data

To understand how we got here, let's look at Figure I-2, which provides a retrospective on how the big data universe developed.

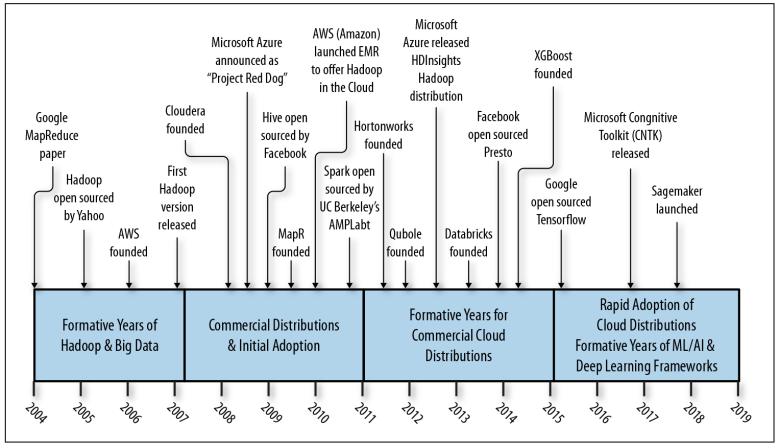

Figure I-2. The evolution of big data

Even now, the big data ecosystem is still under construction. Advancement typically begins with an innovation by a pioneering organization (a Facebook, Google, eBay, Uber, or the like), an innovation created to address a specific challenge that a business encounters in storing, processing, analyzing, or managing its data. Typically, the intellectual property (IP) is eventually open sourced by its creator. Commercialization of the innovation almost inevitably follows.

A significant early milestone in the development of a big data ecosystem was a 2004 whitepaper from Google. Titled "MapReduce: Simplified Data Processing on Large Clusters," it detailed how Google performed distributed information processing with a new engine and resource manager called *MapReduce*.

Struggling with the huge volumes of data it was generating, Google had distributed computations across thousands of machines so that it could finish calculations in time for the results to be useful. The paper addressed issues such as how to parallelize the computation, distribute the data, and handle failures.

Google called it MapReduce because you first use a `map()` function to process a key and generate a set of intermediate keys. Then, you use a `reduce()` function that merges all intermediate values that are associated with the same intermediate key, as demonstrated in Figure I-3

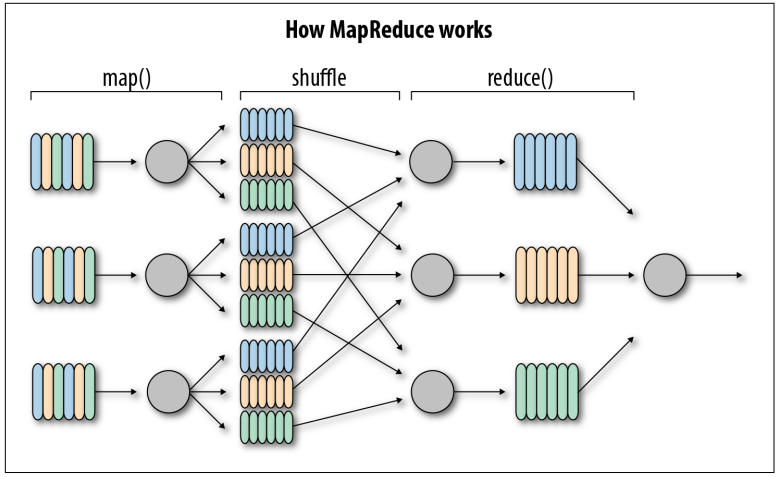

Figure I-3. How MapReduce works

A year after Google published its whitepaper, Doug Cutting of Yahoo combined MapReduce with an open source web search engine called Nutch that had emerged from the Lucene Project (also open source). Cutting realized that MapReduce could solve the storage challenge for the very large files generated as part of Apache Nutch's web-crawling and indexing processes.

By early 2005, developers had a working MapReduce implementation in Nutch, and by the middle of that year, most of the Nutch algorithms had been ported using MapReduce. In February 2006, the team moved out of Nutch completely to found an independent subproject of Lucene. They called this project *Hadoop*, named for a toy stuffed elephant that had belonged to Cutting's then-five-year-old son.

Hadoop became the go-to framework for large-scale, data-intensive deployments. Today, Hadoop has evolved far beyond its beginnings in web indexing and is now used to tackle a huge variety of tasks across multiple industries.

"The block of time between 2004 and 2007 were the truly formative years," says Pradeep Reddy, a solutions architect at Qubole, who has been working with big data systems for more than a decade. "There was really no notion of big data before then."

The Second Phase of Big Data Development

Between 2007 and 2011, a significant number of big data companies —including Cloudera and MapR—were founded in what would be the second major phase of big data development. "And what they essentially did was take the open source Hadoop code and commercialize it," says Reddy. "By creating nice management frameworks around basic Hadoop, they were the first to offer commercial flavors that would accelerate deployment of Hadoop in the enterprise."

So, what was driving all this big data activity? Companies attempting to deal with the masses of data pouring in realized that they needed *faster time to insight*. Businesses themselves needed to be more agile and support complex and increasingly digital business environments that were highly dynamic. The concept of lean manufacturing and just-in-time resources in the enterprise had arrived.

But there was a major problem, says Reddy: "Even as more commercial distributions of Hadoop and open source big data engines began

to emerge, businesses were not benefiting from them, because they were so difficult to us. All of them required specialized skills, and few people other than data scientists had those skills." In the O'Reilly book *Creating a Data-Driven Enterprise with DataOps*, Ashish Thusoo, cofounder and CEO of Qubole, describes how he and Qubole cofounder Joydeep Sen Sarma together addressed this problem while working at Facebook:

> I joined Facebook in August 2007 as part of the data team. It was a new group, set up in the traditional way for that time. The data infrastructure team supported a small group of data professionals who were called upon whenever anyone needed to access or analyze data located in a traditional data warehouse. As was typical in those days, anyone in the company who wanted to get data beyond some small and curated summaries stored in the data warehouse had to come to the data team and make a request. Our data team was excellent, but it could only work so fast: it was a clear bottleneck.
>
> I was delighted to find a former classmate from my undergraduate days at the Indian Institute of Technology already at Facebook. Joydeep Sen Sarma had been hired just a month previously. Our team's charter was simple: to make Facebook's rich trove of data more available.
>
> Our initial challenge was that we had a nonscalable infrastructure that had hit its limits. So, our first step was to experiment with Hadoop. Joydeep created the first Hadoop cluster at Facebook and the first set of jobs, populating the first datasets to be consumed by other engineers—application logs collected using Scribe and application data stored in MySQL.
>
> But Hadoop wasn't (and still isn't) particularly user friendly, even for engineers. It was, and is, a challenging environment. We found that the productivity of our engineers suffered. The bottleneck of data requests persisted. [See Figure I-4.]
>
> SQL, on the other hand, was widely used by both engineers and analysts, and was powerful enough for most analytics requirements. So Joydeep and I decided to make the programmability of Hadoop available to everyone. Our idea: to create a SQL-based declarative language that would allow engineers to plug in their own scripts and programs when SQL wasn't adequate. In addition, it was built to store all of the metadata about Hadoop-based datasets in one place. This latter feature was important because it turned out indispensable for creating the data-driven company that Facebook subsequently became. That language, of course, was Hive, and the rest is history.

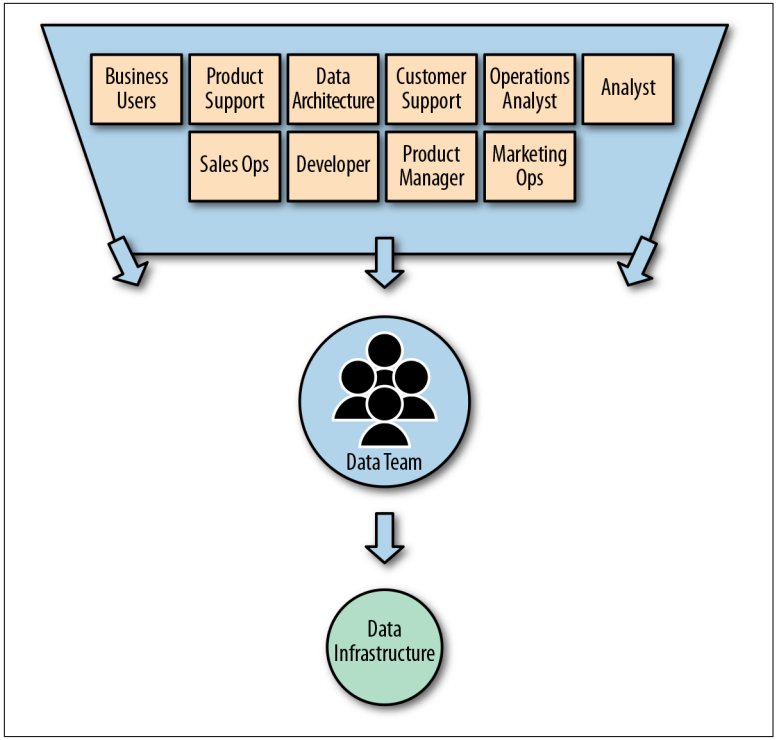

Figure I-4. Human bottlenecks for democratizing data

Says Thusoo today: "Data was clearly too important to be left behind lock and key, accessible only by data engineers. We needed to democratize data across the company—beyond engineering and IT."

Then another innovation appeared: Spark. Spark was originally developed because though memory was becoming cheaper, there was no single engine that could handle both real-time and batch-advanced analytics. Engines such as MapReduce were built specifically for batch processing and Java programming, and they weren't always user-friendly tools for anyone other than data specialists such as analysts and data scientists. Researchers at the University of California at Berkeley's AMPLab asked: is there a way to leverage memory to make big data processing faster?

Spark is a general-purpose, distributed data-processing engine suitable for use in a wide range of applications. On top of the Spark core data-processing engine lay libraries for SQL, machine learning, graph computation, and stream processing, all of which can be used

together in an application. Programming languages supported by Spark include Java, Python, Scala, and R.

Big data practitioners began integrating Spark into their applications to rapidly query, analyze, and transform large amounts of data. Tasks most frequently associated with Spark include ETL and SQL batch jobs across large datasets; processing of streaming data from sensors, Internet of Things (IoT), or financial systems; and machine learning.

In 2010, AMPLab donated the Spark codebase to the Apache Software Foundation, and it became open source. Businesses rapidly began adopting it.

Then, in 2013, Facebook launched another open source engine, Presto. Presto started as a project at Facebook to run interactive analytic queries against a 300 PB data warehouse. It was built on large Hadoop and HDFS-based clusters.

Prior to building Presto, Facebook had been using Hive. Says Reddy, "However, Hive wasn't optimized for fast performance needed in interactive queries, and Facebook needed something that could operate at the petabyte scale."

In November 2013, Facebook open sourced Presto on its own (versus licensing with Apache or MIT) with Apache, and made it available for anyone to download. Today, Presto is a popular engine for large scale, running interactive SQL queries on semi-structured and structured data. Presto shines on the compute side, where many data warehouses can't scale out, thanks to its in-memory engine's ability to handle massive data volume and query concurrency Hadoop.

Facebook's Presto implementation is used today by more than a thousand of its employees, who together run more than 30,000 queries and process more than one petabyte of data daily. The company has moved a number of their large-scale Hive batch workloads into Presto as a result of performance improvements. "[Most] ad hoc queries, before Presto was released, took too much time," says Reddy. "Someone would hit *query* and have time to eat their breakfast before getting results. With Presto you get subsecond results."

"Another interesting trend we're seeing is machine learning and deep learning being applied to big data in the cloud," says Reddy. "The field of artificial intelligence had of course existed for a long time, but beginning in 2015, there was a lot of open source invest-

ments happening around it, enabling machine learning in Spark for distributed computing." The open source community also made significant investments in innovative frameworks like TensorFlow, CNTK, PyTorch, Theano, MXNET, and Keras.

During the Deep Learning Summit at AWS re:Invent 2017, AI and deep learning pioneer Terrence Sejnowski notably said, "Whoever has more data wins." He was summing up what many people now regard as a universal truth: machine learning requires big data to work. Without large, well-maintained training sets, machine learning algorithms—especially deep learning algorithms—fall short of their potential.

But despite the recent increase in applying deep learning algorithms to real-world challenges, there hasn't been a corresponding upswell of innovation in this field. Although new "bleeding edge" algorithms have been released—most recently Geoffrey Hinton's milestone capsule networks—most deep learning algorithms are actually decades old. What's truly driving these new applications of AI and machine learning isn't new algorithms, but bigger data. As Moore's law predicts, data scientists now have incredible compute and storage capabilities that today allow them to make use of the massive amounts of data being collected.

Weather Update: Clouds Ahead

Within a year of Hadoop's introduction, another important—at the time seemingly unrelated—event occurred. Amazon launched AWS in 2006. Of course, the cloud had been around for a while. Project MAC, begun by the Defense Advanced Research Projects Agency (DARPA) in 1963, was arguably the first primitive instance of a cloud, "but Amazon's move turned out to be critical for advancement of a big data ecosystem for enterprises," says Reddy.

Google, naturally, wasn't far behind. According to "An Annotated History of Google's Cloud Platform," in April 2008, App Engine launched for 20,000 developers as a tool to run web applications on Google's infrastructure. Applications had to be written in Python and were limited to 500 MB of storage, 200 million megacycles of CPU, and 10 GB bandwidth per day. In May 2008, Google opened signups to all developers. The service was an immediate hit.

Microsoft tried to catch up with Google and Amazon by announcing Azure Cloud, codenamed Red Dog, also in 2008. But it would take years for Microsoft to get it out the door. Today, however, Microsoft Azure is growing quickly. It currently has 29.4% of application workloads in the public cloud, according to a recent Cloud Security Alliance (CSA) report. That being said, AWS continues to be the most popular, with 41.5% of application workloads. Google trails far behind, with just 3% of the installed base. However, the market is still considered immature and continues to develop as new cloud providers enter. Stay tuned; there is still room for others such as IBM, Alibaba, and Oracle to seize market share, but the window is beginning to close.

Bringing Big Data and Cloud Together

Another major event that happened around the time of the second phase of big data development is that Amazon launched the first cloud distribution of Hadoop by offering the framework in its AWS cloud ecosystem. Amazon Elastic MapReduce (EMR) is a web service that uses Hadoop to process vast amounts of data in the cloud. "And from the very beginning, Amazon offered Hadoop and Hive," says Reddy. He adds that though Amazon also began offering Spark and other big data engines, "2010 is the birth of a cloud-native Hadoop distribution—a very important timeline event."

Commercial Cloud Distributions: The Formative Years

Reddy calls 2011–2015 the "formative" years of commercial cloud Hadoop platforms. He adds that, "within this period, we saw the revolutionary idea of separating storage and compute emerge."

Qubole's founders came from Facebook, where they were the creators of Apache Hive and the key architects of Facebook's internal data platforms. In 2011, when they founded Qubole, they set out on a mission to create a cloud-agnostic, cloud-native big data distribution platform to replicate their success at Facebook in the cloud. In doing so, they pioneered a new market.

Through the choice of engines, tools, and technologies, Qubole caters to users with diverse skillsets and enables a wide spectrum of

big data use cases like ETL, data prep and ingestion, business intelligence (BI), and advanced analytics with machine learning and AI.

Qubole incorporated in 2011, founded on the belief that big data analytics workloads belong in the cloud. Its platform brings all the benefits of the cloud to a broader range of users. Indeed, Thusoo and Sarma started Qubole to "bring the template for hypergrowth companies like Facebook and Google to the enterprise."

"We asked companies what was holding them back from using machine learning to do advanced analytics. They said, 'We have no expertise and no platform,'" Thusoo said in a 2018 interview with *Forbes*. "We delivered a cloud-based unified platform that runs on AWS, Microsoft Azure, and Oracle Cloud." During this same period of evolution, Facebook's open sourced Presto enabled fast business intelligence on top of Hadoop. Presto is meant to deliver accelerated access to the data for interactive analytics queries.

2011 also saw the founding of another commercial on-premises distribution platform: Hortonworks. Microsoft Azure later teamed up with Hortonworks to repackage Hortonworks Data Platform (HDP) and in 2012 released its cloud big data distribution for Azure under the name HDInsight.

OSS Monopolies? Not in the Cloud

An interesting controversy has arisen in the intersection between the open source software (OSS) and cloud worlds, as captured in an article by Qubole cofounder Joydeep Sen Sarma. Specifically, the AWS launch of Kafka as a managed service seems to have finally brought the friction between OSS and cloud vendors out into the open. Although many in the industry seem to view AWS as the villain, Sarma disagrees. He points out that open source started as a way to share knowledge and build upon it collectively, which he calls "a noble goal." Then, open source became an alternative to standards-based technology—particularly in the big data space. This led to an interesting phenomenon: the rise of the open source monopoly. OSS thus became a business model. "OSS vendors hired out most of the project committers and became de facto owners of their projects," wrote Sarma, adding that of course venture capitalists pounced; why wouldn't they enjoy the monopolies that such arrangements enabled? But cloud vendors soon caught up. AWS in particular crushed the venture capitalists' dreams. No one does

commodity and Software as a Service (SaaS) better than AWS. Taking code and converting it into a low-cost web service is an art form at which AWS excels. To say that OSS vendors were not pleased is an understatement. They began trying to get in the way of AWS and other cloud platform vendors. But customers (businesses in this case) do better when competition exists. As software consumption increasingly goes to SaaS, we need the competition that AWS and others provide. Wrote Sarma, "Delivering highly reliable web services and fantastic vertically integrated product experiences online is a different specialty than spawning a successful OSS project and fostering a great community." He sees no reason why success in the latter should automatically extend to a monopoly in the former. Success must be earned in both markets.

As previously mentioned, in 2012 Microsoft released HDInsight, its first commercial cloud distribution. Then in 2013, another big data platform provider, Databricks, was launched. Founded by the creators of Apache Spark, Databricks aims to help clients with cloud-based big data processing. This marked the beginning of a new era of "born-in-the-cloud" SaaS companies that were aligned perfectly with the operational agility and pricing structure of the cloud.

Big Data and AI Move Decisively to the Cloud, but Operationalizing Initiatives Lag

Since 2015, big data has steadily moved to the cloud. The most popular open source projects (Apache Kafka, ElasticSearch, Presto, Apache Hadoop, Spark, and many others) all have operators built for various cloud commodities (such as storage and compute) and managed services (such as databases, monitoring apps, and more). These open source communities (largely comprising other enterprise practitioners) are also using the cloud in their workplaces, and we're seeing some extraordinary contributions going into these projects from developers worldwide.

"We've seen a lot of enterprise companies moving away from on-premises deployments because of the pain of hitting the wall in terms of capacity," says Reddy, adding that, with the cloud, the notion of multitenancy (or sharing a cluster across many users) came full circle. In the cloud, "it's all about creating clusters for spe-

cific use cases, and right-sizing them to get the most out of them," says Reddy.

Back when Hadoop was in its infancy, Yahoo began building Hadoop on-demand clusters—dedicated clusters of Hadoop that lacked multitenancy. Yahoo would bring these clusters up for dedicated tasks, perform necessary big data operations, and then tear them down.

But since then, most of the advancements around Hadoop have been focused around multitenant capabilities. The YARN (yet another resource negotiator) project was chartered with this as one of its main objectives. YARN delivered and helped Hadoop platforms expand in the enterprises that adopted it early. But there was a problem. The velocity and volume of data was increasing at such an exponential rate that all these enterprises that implemented big data on-premises would soon hit the ceiling in terms of capacity. They'd require multiple hardware refreshes to meet the demand for data processing. Multitenancy on-premises also requires a lot of administration time to manage fair share across the different users and workloads.

Today, in the cloud, we see Hadoop on-demand clusters similar to those we saw when Hadoop was in its infancy. As Reddy said, the focus is more about right-sizing the clusters for specific uses rather than enabling multitenancy. Multitenancy is still very relevant in the cloud for Presto, although not as much for Hive and Spark clusters.

At the present time, cloud deployments of big data represent as much as 57% of all big data workloads, according to Gartner. And global spending on big data solutions via cloud subscriptions will grow almost 7.5 times faster than those on-premises, says Forrester, which found that moving to the public cloud was the number-one technology priority for big data practitioners, according to its 2017 survey of data analytics professionals (see Figure I-5).

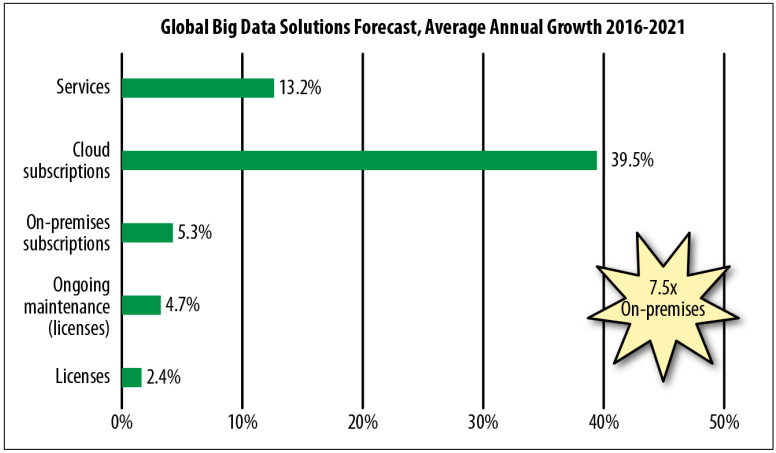

Figure I-5. Growth of big data solutions

This represents just the foundational years of machine learning and deep learning, stresses Reddy. "There is a lot more to come."

By 2030, applying AI technologies such as machine learning and deep learning to big data deployments will be a $15.7 trillion "game changer," according to PwC. Also, 59% of executives say their companies' ability to leverage big data will be significantly enhanced by applying AI.

Indeed, big data and AI are becoming inexorably intertwined. A number of recent industry surveys have unanimously agreed that from the top down, companies are ramping up their investment in advanced analytics as a key priority of this decade. In NewVantage Partners' annual executive survey, an overwhelming 97.2% of executives report that their companies are investing in building or launching combined big data and AI initiatives. And 76.5% said the proliferation of data is empowering AI and cognitive computing initiatives.

One reason for the quick marriage of big data and AI on the cloud is that most companies surveyed were worried that they would be "disrupted" by new market entrants.

AI and Big Data: A Disruptive Force for Good

The technology judged most disruptive today is AI. A full 72% of executives in the NewVantage survey chose it as the disruptive tech-

nology with the most impact. And 73% said they have already received measurable value from their big data and AI projects.

But they also reported having trouble deploying these technologies. When executives were asked to rate their companies' ability to *operationalize* big data—that is, make it a key part of a data-driven organization—the results were somewhat mixed, according to Forrester. Few have achieved all of their goals, as shown in Figure I-6.

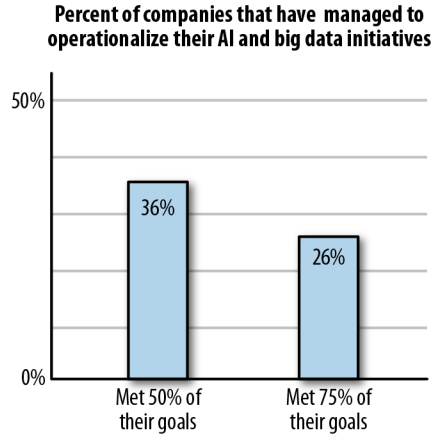

Figure I-6. Few companies have managed to operationalize their AI and big data initiatives

And a global survey by Gartner indicated that the overwhelming majority—91%—of businesses have yet to achieve a "transformational" level of maturity in big data, despite such activities being a number one investment priority for CIOs recently.

In this survey, Gartner asked organizations to rate themselves based on Gartner's big data maturity model—ranging from Level 1 (basic) to Level 2 (opportunistic) to Level 3 (systematic) to Level 4 (differentiating), and to Level 5 (transformational)—and found that 60% placed themselves in the lowest three levels.

We Believe in the Cloud for Big Data and AI

The premise of this book is that by taking advantage of the compute power and scalability of the cloud and the right open source big data and AI engines and tools, businesses can finally operationalize their big data. This will allow them to be more innovative and collabora-

tive, achieving analytical value in less time while lowering operational costs. The ultimate result: businesses will achieve their goals faster and more effectively while accelerating time to market through intelligent use of data.

This book is designed not to be a tow rope, but a guiding line for all members of the data team—from data engineers to data scientists to machine learning engineers to analysts—to help them understand how to operationalize big data and machine learning in the cloud.

Following is a snapshot of what you will learn in this book:

Chapter 1
: You learn why you need a "central repository" to be able to use your data effectively. In short, you'll learn why you need a data lake.

Chapter 2
: You need a data-driven culture, but it can be challenging to get there. This chapter explains how.

Chapter 3
: We show you how to begin to build a data lake.

Chapter 4
: We discuss building the infrastructure for the data lake. What kind of structure do you need to house your big data?

Chapter 5
: Now that you've built the "house" for your data lake, you need to consider governance. In this chapter, we cover three necessary governance plans: data, financial, and security.

Chapter 6
: You'll need some tools to manage your growing data lake. Here, we provide a roundup of those tools.

Chapter 7
: We examine three key considerations for securing a data lake in the cloud.

Chapter 8
: We discuss the role of data engineers, and how they interface with a cloud-native data platform.

Chapter 9

We discuss the role of data scientists, and how they interface with a cloud-native data platform.

Chapter 10

We discuss the role of data analysts, and how they interface with a cloud-native data platform.

Chapter 11

We present a case study from Ibotta, which transitioned from a static and rigid data warehouse to a cost-efficient, self-service data lake using Qubole's cloud-native data platform.

Chapter 12

We conclude by examining why a cloud data platform is a future-proof approach to operationalizing your data lake.

CHAPTER 1

The Data Lake: A Central Repository

In the introduction, we examined how companies are beginning to recognize the value of their data. Every business with information systems essentially uses a data lake, whether it is a staging environment for a data mart or a temporary storage layer for downstream processes like loading data warehouses or delivering to core systems.

These businesses are attempting to "activate" the data—that is, put it in the hands of the business users who need it—by taking advantage of the power of data lakes to *operationalize* their information in a way that can grow with the company.

These users almost immediately encounter several significant problems. Different users or teams might not be using the same versions of data. This happens when a dataset—for example, quarterly sales data—is split into or distributed to different data marts or other types of system silos in different departments. The data typically is cleaned, formatted, or changed in some way to fit these different types of users. Accounts payable and marketing departments may be looking at different versions of sales results. Each department may have their unconscious assumptions and biases that cause them to use the data in different ways. And sometimes the data itself is biased, which we look at more closely in the sidebar that follows.

The Four (Theoretical) Evils of Analytics

The four primary types of biases are information, confirmation, interpretation, and prediction. One or more of these biases might come into play during the data life cycle. Here are a few examples:

Information bias
: This refers to bias regarding the origin of the data—the "when" and "where." For example, take flu statistics. Where is the data coming from? Is the data coming in evenly from across the world? Or is it skewed to a few countries or cities? Is the data being collected in a standardized way?

Confirmation bias
: This bias is probably the most common of the four. We, as humans, subconsciously make inferences about data and look for evidence to support those inferences. Data analysts must understand their own biases, be ready to reexamine the data, and put aside any preconceived notions or views.

Interpretation bias
: This bias comes into play when framing the data for analysis. Subtle stimuli in the framing of a question can change the bias of the analysis. For example, take these two survey questions: "What speed do you think the cars were going when they collided?" versus "What speed do you think the cars were going when they smashed?" By using a more violent word, *smashed*, the second question tempts the survey subject to provide a higher number, thus skewing the results.

Prediction bias
: Attempting to predict future events from past ones is always difficult. Relying too much on certain data inputs to make such predictions runs the risk of allocating resources to an area where they are not actually needed and thus wasted. To decrease such biases, data must always be evaluated by a human who can make the subtle differentiations that machines are not yet capable of doing.

Companies commonly run into a "data whitespace" problem because of unstructured data. This happens when you can't see all of your data. Traditional tools such as PostgreSQL, MySQL, and Oracle are all good for storing and querying structured data. You also have

unstructured data like log files, video files, and audio files that you can't fit into databases. You end up with data, but you can't do anything with it. This leaves holes in your ability to "see" your business.

Enter the data lake. The idea behind a data lake is to have one place where all company data resides. This raw data—which implies an exact copy of data from whatever source it came from—is an immutable record that a business can then utilize to *transform* data, so that it can be used for reporting, visualization, analytics, machine learning, and business insights.

What Is a Data Lake?

A data lake is a central repository that allows you to store all your data—structured and unstructured—in volume, as shown in Figure 1-1. Data typically is stored in a raw format (i.e., as is) without first being structured. From there it can be scrubbed and optimized for the purpose at hand, be it dashboards for interactive analytics, downstream machine learning, or analytics applications. Ultimately, the data lake enables your data team to work collectively on the same information, which can be curated and secured for the right team or operation.

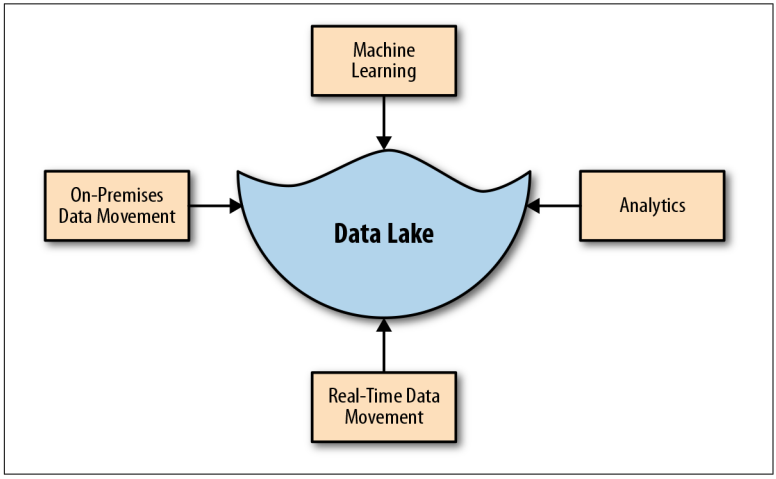

Figure 1-1. What is a data lake?

Although this book is about building data lakes in the cloud, we can also build them on-premises. However, as we delve further into the topic, you will see why it makes sense to build your data lake in the

cloud. (More on that in Chapter 3.) According to Ovum ICT Enterprise Insights, 27.5% of big data workloads are currently running in the cloud.

The beauty of a data lake is that everyone is looking at and operating from the same data. Eliminating multiple sources of data and having a referenceable "golden" dataset in the data lake leads to alignment within the organization, because any other downstream repository or technology used to access intelligence in your organization will be synchronized. This is critical. With this centralized source of data, you're not pulling bits of data from disparate silos; everyone in the organization has a single source of truth. This directly affects strategic business operations, as everyone from the C-suite on down is making important strategic decisions based upon a single, immutable, data source.

Data Lakes and the Five Vs of Big Data

The ultimate goal of putting your data in a data lake is to reduce the time it takes to move from storing raw *data* to retrieving actionable and valuable *information*. But to reach that point, you need to have an understanding of what big data is across your data team. You need to understand the "five Vs" model of big data, as shown in Figure 1-2. Otherwise, your data lake will be a mess—commonly known as a *data swamp*.

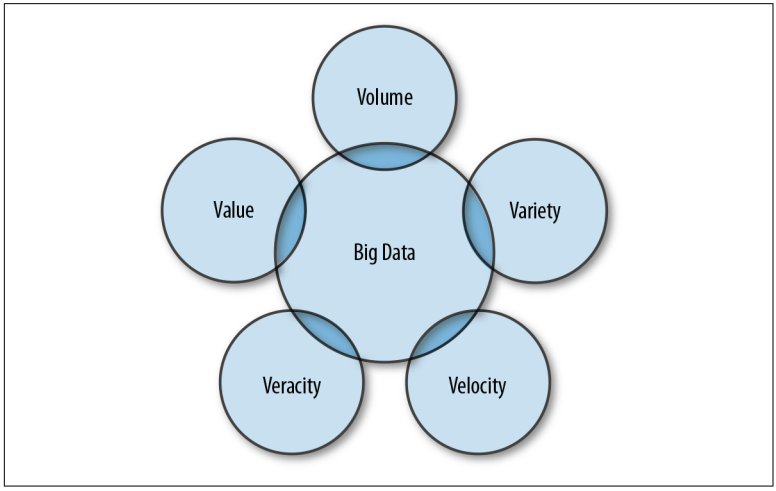

Figure 1-2. The five Vs model of big data

Big data, by definition, describes three different types of data: structured, semi-structured, and unstructured. The complexity of the resulting data infrastructure requires powerful management and technological solutions to get value out of it.

Most data scientists define big data as having three key characteristics: volume, velocity, and variety. More recently, they've added two other qualities: veracity and value. Let's explore each in turn:

Volume

Big data is big; this is its most obvious characteristic. Every second, almost inconceivable amounts of data are generated from financial transactions, ecommerce transactions, social media, phones, cars, credit cards, sensors, video, and more. The data has in fact become so vast that we can't store or process it using traditional databases. Instead, we need distributed systems in which data is stored across many different machines—either physical (real) or virtual—and managed as a whole by software. IDC predicts that the "global datasphere" will grow from 33 zettabytes (ZB) in 2018 to 175 ZB by 2025. One ZB is approximately equal to a thousand exabytes, a billion terabytes, or a trillion gigabytes. Visualize this: if each terabyte were a kilometer, a ZB would be equivalent to 1,300 round trips to the moon.

Velocity

Next, there's the velocity, or *speed*, of big data. Not only is the volume huge, but the rate at which it is generated is blindingly fast. Every minute, the Weather Channel receives 18 million forecast requests, YouTube users watch 4.1 million videos, Google delivers results for 3.6 million searches, and Wikipedia users publish 600 new edits. And that's just the tip of the iceberg. Not only must this data be analyzed, but access to the data must also be instantaneous to allow for applications like real-time access to websites, credit card verifications, and instant messaging. As it has matured, big data technology has allowed us to analyze extremely fast data even as it is generated, even without storing it in a database.

Variety

Big data is made up of many different types of data. No longer having the luxury of working with structured data that fits cleanly into databases, spreadsheets, or tables, today's data teams work with semi-structured data such as XML, open-

standing JSON, or NoSQL; or they must contend with completely unstructured data such as emails, texts, and human-generated documents such as word processing or presentation documents, photos, videos, and social media updates. Most—approximately 85%—of today's data is unstructured. Previously, this data was not considered usable. But modern big data technologies have enabled all three types of data to be generated, stored, analyzed, and consumed simultaneously, as illustrated in Figure 1-3.

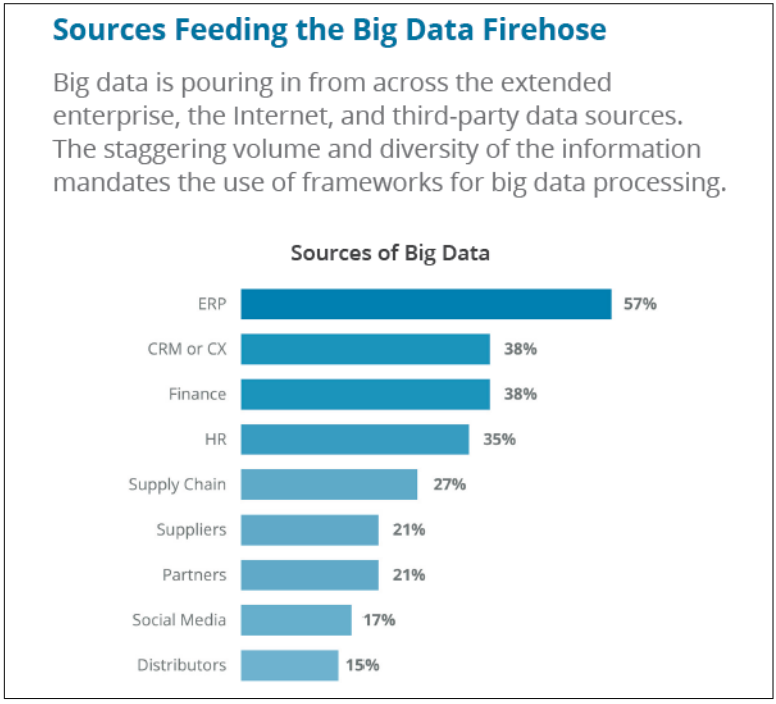

Figure 1-3. The variety of sources in big data deployments

Veracity

Veracity, the *quality* of the data, is a recent addition to the original three attributes of the big data definition. How accurate is your data? Can you trust it? If not, analyzing large volumes of data is not only a meaningless exercise, but alsoa dangerous one given that inaccurate data can lead to wrong conclusions.

Value

Finally, there's the big question: what is the data worth? Generating or collecting massive volumes of data is, again, pointless if you cannot transform it into something of value. This is where financial governance of big data comes in. Researchers have found a clear link between data, insights, and profitability, but businesses still need to be able to calculate the relative costs and benefits of collecting, storing, analyzing, and retrieving the data to make sure that it can ultimately be monetized in some way.

Data Lake Consumers and Operators

Big data stakeholders can be loosely categorized as either operators or consumers. The consumer category can be further divided into internal and external users. (We provide more granular definitions of roles in the following section.) Both camps have different roles and responsibilities in interacting with each of the five Vs.

Operators

Data operators include data engineers, data architects, and data and infrastructure administrators. They are responsible for dealing with the volume, velocity, variety, and veracity of the data. Thus they must ensure that large amounts of information, no matter the speed, arrive at the correct data stores and processes on time. They're responsible for ensuring that the data is clean and uncorrupted. In addition, the operators define and enforce access policies—policies that determine who has access to what data.

Indeed, ensuring veracity is probably the biggest challenge for operators. If you can't trust the data, the source of the data, or the processes you are using to identify which data is important, you have a veracity problem. These errors can be caused by user entry errors, redundancy, corruption, and myriad other factors. And one serious problem with big data is that errors tend to snowball and become worse over time. This is why an operator's primary responsibility is to catalog data to ensure the information is well governed but still accessible to the right users.

In many on-premises data lakes, operators end up being the bottlenecks. After all, they're the ones who must provision the infrastructure, ingest the data, ensure that the correct governance processes are in place, and otherwise make sure the foundational infrastruc-

ture and data engineering is robust. This can be a challenge, and resources are often limited. Rarely is there is enough compute, memory, storage—or even people—to adequately meet demand.

But in the cloud, with the scalability, elasticity, and tools that it offers, operators have a much easier time managing and operationalizing a data lake. They can allocate a budget and encourage consumers to figure things out for themselves. That's because after a company has placed the data in the data lake (the source of truth) and created a platform that can easily acquire resources (compute), project leaders can more effectively allocate a budget for the project. After the budget is allocated, the teams are now empowered to decide on and get the resources themselves and do the work within the given project timeframe.

Consumers (Both Internal and External)

Consumers are the data scientists and data analysts, employee citizen scientists, managers, and executives, as well as external field workers or even customers who are responsible for drawing conclusions about what's in the data. They are responsible for finding the value, the fifth V, in it. They also must deal with the volume of data when it comes to the amount of information that they need to sift through as well as the frequency of requests they get for ad hoc reports and answers to specific queries. These teams are often also working with a variety of unstructured and structured datasets to create more usable information. They are the ones who analyze the data produced by the operators to create valuable and actionable insights. The consumers use the infrastructure managed by the operators to do these analyses.

There are internal and external users of the data lake. Internal users are those developing tools and recommendations for use within your company. External users are outside your company; they want limited access to data residing in the data lake. Google AdSense is an example of this: Google is developing tools and insights for its internal use at a global level. At the same time, it is developing portals for external entities to gain insights into their advertising companies. If I'm the Coca-Cola Company, for instance, I would want to know the success rate of my ads targeting Pepsi users, and whether I can use my advertising dollars better.

Challenges in Operationalizing Data Lakes

In 2017, Gartner estimated that perhaps 60% of all big data projects fail. That sounds grim; but the reality is worse. According to a Twitter post by Gartner analyst Nick Heudecker, Gartner was "too conservative" with this estimate. Heudecker in 2018 estimated that the real failure rate was "closer to 85%." But the problem wasn't a technical one; it was because of the humans—the operators and consumers—who were necessarily involved in the process.

McKinsey came to the same pessimistic conclusion. Though research done by the McKinsey Global Institute (a sister organization) generated a lot of excitement by predicting that retailers using big data and analytics could improve their operating margins by more than 60%, when McKinsey recently gathered analytics leaders from leading large enterprises and asked about the revenue or cost benefits they'd received from big data, 75% said it had been less than 1%. Why was this?

Here are the three most common reasons why big data projects fail:

Shortage of resources and expertise
: Data science jobs are already difficult to fill, and according to *The Quant Crunch* report, demand is expected to rise 28% by 2020. Businesses eager to begin big data projects are often frustrated by the lack of data science skills in their talent pool, as depicted in Figure 1-4, and mistakes made by relative newbies to big data can cause major setbacks.

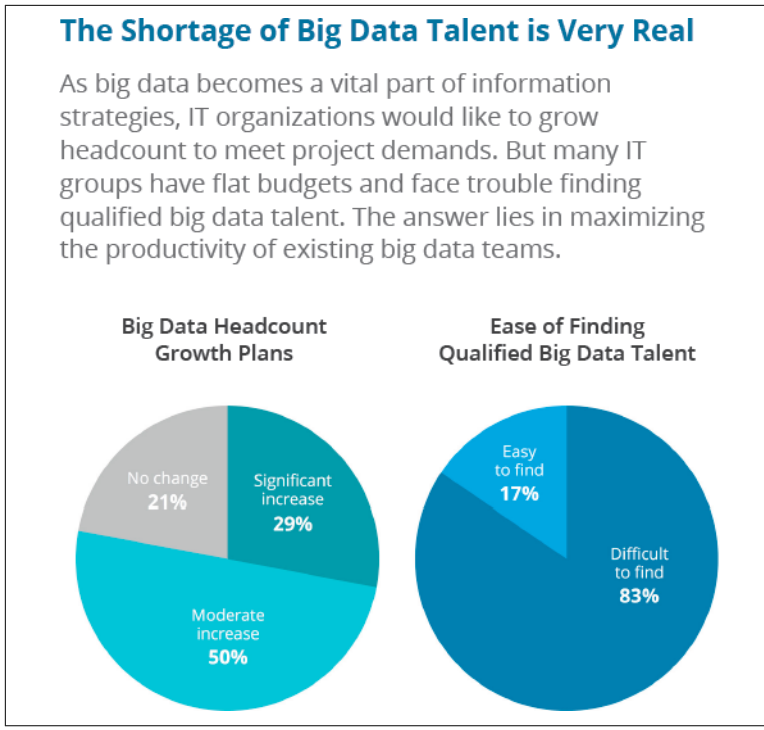

Figure 1-4. Shortage of big data talent is very real

Costs are too high

Talent is not only scarce, it's expensive. And many businesses are dependent on third-party consulting firms to successfully complete projects, which adds to the cost.

In addition, data lakes consume a lot of infrastructure (compute and storage). If proper financial governance is not put in place, companies risk incurring runaway infrastructure costs with no visibility on their ROI.

At least part of the problem is that organizations don't provide sufficient resources to their big data projects. According to a Qubole survey, a full 75% of companies identified a gap between their big data resources and the potential value of the project(s), as shown in Figure 1-5.

Figure 1-5. The gap between big data needs and resources

Cloud-native platforms like Qubole intelligently optimize resources, as demonstrated in Figure 1-6. They automatically assign more capacity when needed and release resources when workloads require less capacity by doing intelligent workload-aware autoscaling. This is a huge game changer for organizations that pay only for what they use rather than preemptively ordering capacity and hiring teams to provision and maintain that technology.

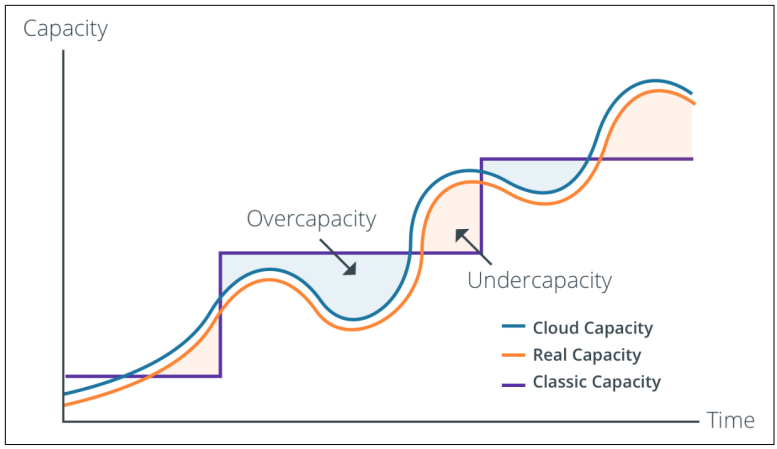

Figure 1-6. How cloud-native big data platforms can intelligently optimize resources

It takes too long to realize value

Because of the relative newness of the technologies, coupled with the difficulty of finding expertise, many big data projects fail to deliver within expected timeframes, as shown in Figure 1-7.

Given these challenges, should companies build a solution or buy one? This question of "build or buy" inevitably comes up in a big data project. It's important to not conflate technical and business issues when making this decision. When faced with a technical challenge, don't hesitate: buy your way out of it. If someone has already created a solution, purchase it.

What we've seen is that most do-it-yourself projects return less-than-expected value because the teams spend most (75%) of the allocated time simply acquiring sufficient resources. This leaves less than 25% of the project time to realize value from the data. Now imagine a project timeline for which technology resources can be acquired in minutes, leaving more than 90% of the project timeframe to find and acquire data. Being able to easily spin up an engine provides faster iteration, and you'll get answers to queries in seconds versus minutes. Having the right platform and engine in a matter of minutes means that projects are better positioned to find significant value from data in less time.

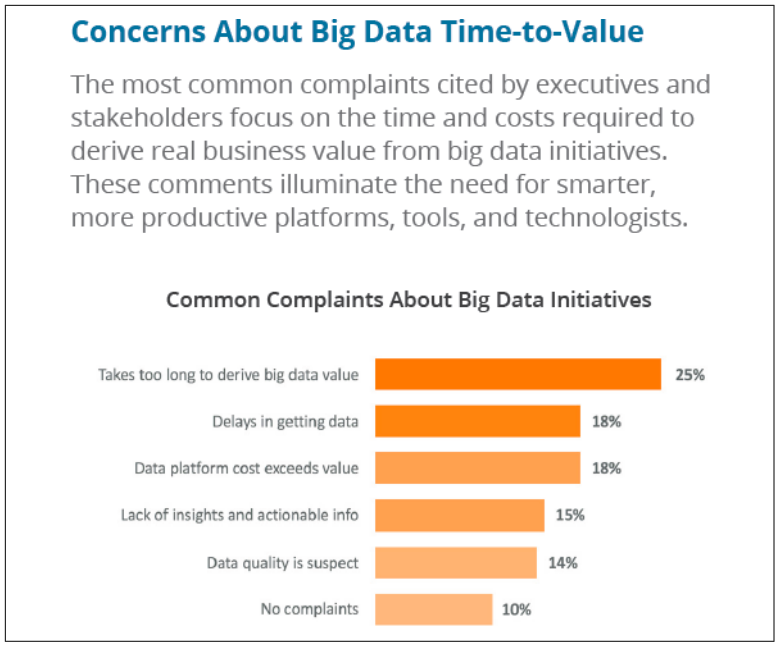

Figure 1-7. Concerns about big data time to value

The SaaS model also provides advantages over licensed distributions for which you buy software by the node for a yearly license, because the latter method is too fixed and outdated to keep up with data growth or the speed of innovation.

CHAPTER 2
The Importance of Building a Self-Service Culture

Before we can talk about how to build a data lake, we need to discuss the culture of the company using that data lake and, more specifically, the mental shift required for organizations to fully embrace the value of a data lake. In more traditional organizations, the DataOps team stands between the data and the business users. Traditionally, when a user needs data, they approach a data analyst or data scientist and make a request. The data team then responds. It's common for the data team to build dashboards and have a set of prebuilt reports that it refreshes or sends out periodically, but so-called ad hoc requests are usually handled on a case-by-case basis.

This gatekeeper approach inevitably causes bottlenecks, as shown in Figure 2-1. Users who need data for a key presentation or to make a strategic decision are forced to wait for their turns in the queue. Often, they give up, and make the decision without having the data to back it up. And it becomes difficult, if not impossible, for an organization to extract the full value from its data using this paradigm. Because of this, a *self-service culture* is essential if your company is going to get the most value from your data and, eventually, become a true *data-driven organization*.

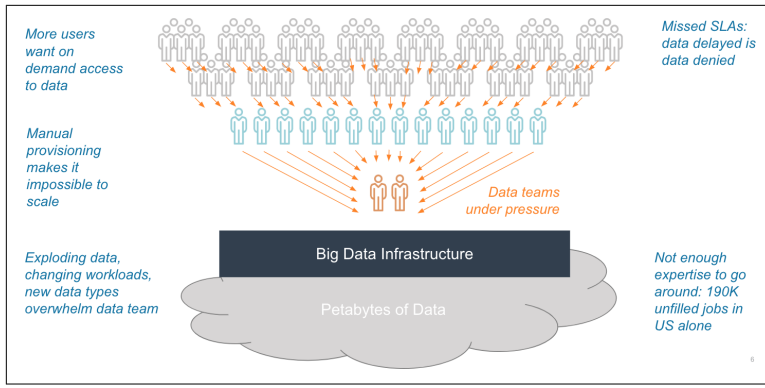

Figure 2-1. The importance of building a self-service culture

The End Goal: Becoming a Data-Driven Organization

A data-driven business is one in which decisions are powered by data as opposed to intuition or even personal experience. It's one in which people who tend to "think from their gut" are encouraged to use hard empirical evidence to back up what they say and do. In a data-driven company like Facebook, for example, no one would think of showing up to an important meeting without quantifiable facts to back up their position.

According to Forrester, data-driven companies grow eight times faster than those that work from intuition or speculation. Insights-driven businesses grow on average more than 30% annually and are on track to earn $1.8 trillion by 2021.

Obviously, a first step toward being data driven is to make data readily accessible to everyone in the organization—that is, to *democratize* the data. And this means having a self-service data culture.

Numerous—if not most—companies today have announced their intentions of becoming data driven. Many have already started down this road. The basic premise has long been that deploying a data warehouse, populating it with company data, and hiring a team of intelligence analysts will lead in no time to data-driven nirvana.

But it's not happening very quickly. Virtually all respondents to a recent NewVantage survey said that their firms are trying to make the shift, but only about a third have succeeded.

A data-driven organization should possess three things, according to Ashish Thusoo, cofounder and CEO of Qubole, as documented in the O'Reilly book *Creating a Data-Driven Enterprise with DataOps*:

- A culture in which everyone buys into the idea of using data to make business decisions
- An organizational structure that supports a data-driven culture
- Technology that makes data self-service

We are focusing on the last point in this book, but let's go over all three requirements nonetheless.

Foster a Culture of Data-Driven Decision Making

Whether you know it or not, your business already has a decision-making culture. The problem is that your culture might not be geared toward a data-driven approach. It might be ingrained in the culture of your organization, like many others, that the "HIPPO" (highest-paid person in the office) is the last stop in the decision-making process, and the senior person in the meeting gets to make the final choice. Unfortunately, the HIPPO can at times be very wrong. But unless you have the data to use as evidence for your arguments (and your company culture supports arguing with the top brass in the organization), their decision usually stands.

To successfully become data driven, your employees should *always* use data to start, continue, or conclude every single business decision, no matter how major or minor.

You probably need to start the shift from the top. For example, many marketing departments are becoming more data driven. A chief marketing officer (CMO) can set the right tone by making it mandatory to experiment with and test new creative initiatives and campaigns to gather data on their impact, as opposed to simply relying on gut feeling and intuition. That message gives primacy to data, and that sentiment then flows to the rest of the marketing team.

It's important to understand that different employees within your organization will have different reasons for buying into using data in their day-to-day jobs. You first must identify who all of these stakeholders are. Then, you must understand what will motivate them to begin using data to make decisions. And then you must make it easy

for them to serve themselves, organizationally and technically, a topic we address in the next two sections.

Build an Organizational Structure That Supports a Self-Service Culture

Organizationally, how do you support a data-driven company? Most successful data-driven entities have created a central data platform team that publishes data and manages the necessary technical infrastructure. Although some organizations establish multiple data teams and embed them in different departments, each catering to the needs of that particular department, this is typically less useful if you hope to get to data-driven nirvana because it ends up creating a number of isolated data silos. Therefore, a strong, functional, central data team is extremely important. As we discussed in Chapter 1, this provides a single source of truth that underpins all data analyses.

Your next step is to embed business function–specific analysts within each department, staff specially designated to help users in that department extract value from your company's data. This works best because those analysts possess the domain-specific knowledge about the business function—the marketing data analyst understands campaigns, and the finance data analyst understands accounts payables—while also having intimate knowledge of the data. They can convert the language of the data systems to the language of the business, as illustrated in Figure 2-2.

This is critical because the two languages are very different. The business wants to ask questions such as the following:

- Which geographic regions of my business are the best to invest in?
- What is the size of the market?
- Who is the competition?
- What are the best market opportunities today?

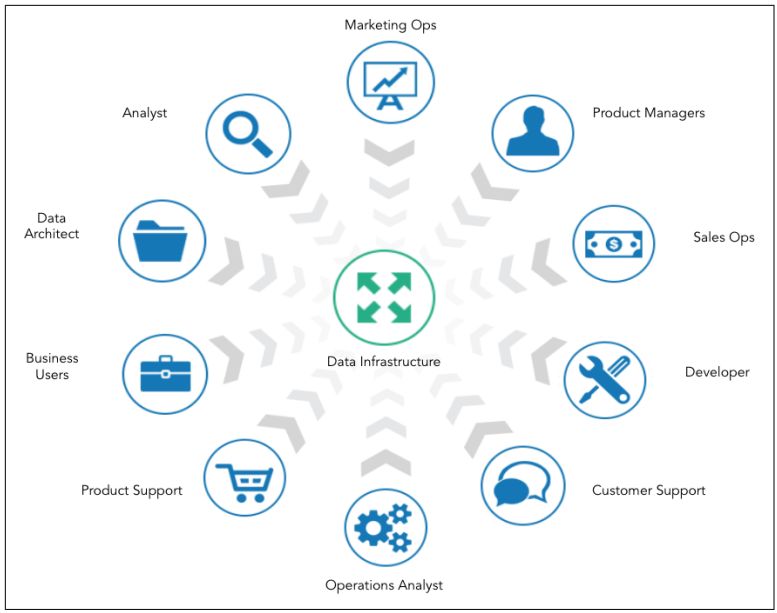

Figure 2-2. The hub-and-spoke data organizational model

According to Thusoo, Facebook quickly understood that it needed a centralized data team. Then, it embedded analysts in every product team. "We also took care that all the analysts had a central forum at which they could meet and communicate what they were doing, allowing data intelligence to flow through the entire organization," he says.

Essentially, this model transmitted the data-driven DNA of the self-service culture throughout the company. If any link in this "value chain" of data is weak, you face barriers to developing a true data-driven culture. Chief data officers usually carry the ultimate responsibility of nurturing and growing this value chain of data.

Putting a Self-Service Technological Infrastructure in Place

All data-using employees must be supported by the central data team if you hope to achieve self-service. This team is ultimately responsible for maintaining the infrastructure. Their job is to deploy whatever people, processes, and technologies are necessary to make data available to everyone who needs it, on a self-service basis.

In a recent survey by Qubole, 70% of organizations said a self-service environment was either already in place or planned, as shown in Figure 2-3.

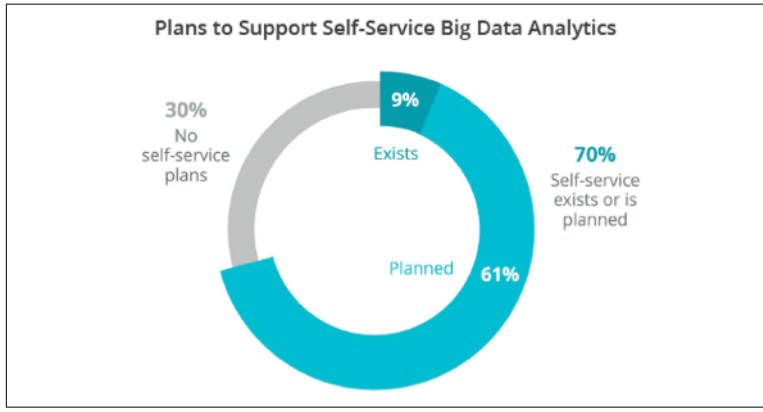

Figure 2-3. Plans to build a self-service environment

By empowering users to explore data easily and with the tools they're comfortable with, data teams can keep staffing low and raise the productivity and effectiveness of business users throughout the organization. The key here is that data engineering teams need to be able to automate the underlying infrastructure to be fully aligned with key business initiatives. If they don't do that, they'll waste time providing information to other data teams rather than doing infrastructure setup and maintenance.

You'll know that your company is genuinely data driven when "bottom-up" demand for self-service data access emerges. When this happens, you must ensure that the tools and mechanisms are there to support this bottom-up interest among employees. For example, with self-service tools and processes in place, employees in marketing could themselves find and analyze previous campaigns' datasets—which are stored in the centralized data lake—to come up with ideas for successful future campaigns and messaging.

Challenges of Building a Self-Service Infrastructure

When it comes to creating a data-driven infrastructure that includes a data lake, organizations most often face the following four challenges:

- Lack of specialized expertise
- The disparity and distribution of data
- Organizational resistance
- Reluctance to commit to open source

Let's examine each of these a bit more closely.

Lack of Specialized Expertise

Building a data lake for truly big data requires specialized technologies and skillsets. In fact, the primary big data challenge in the Qubole survey was lack of experience, as shown in Figure 2-4.

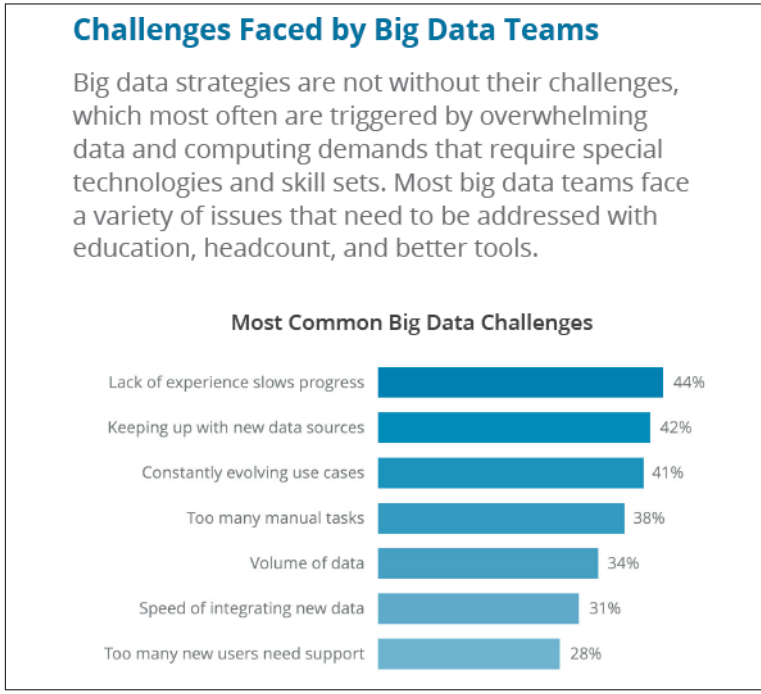

Figure 2-4. Challenges faced by big data teams

The 80/20 rule appears to be true in the big data world, in which only 20% of the technology workforce has adequate hands-on experience building big data platforms that can scale as needed. This raises urgent issues, because data in corporate data lakes is growing year after year, as depicted in Figure 2-5.

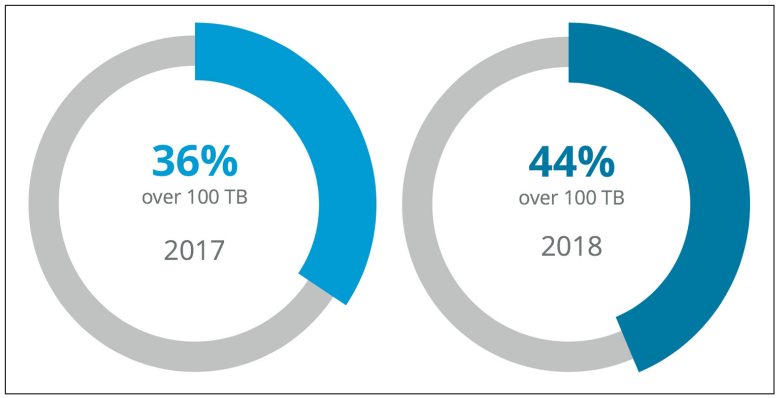

Figure 2-5. The growth of corporate data stores

During the initial exploratory phases, users might experiment with data on their local laptops. This works for the short term because the datasets don't exceed the resource limits of their system. However, as soon as data begins to grow, either in volume or complexity, the physics required to process it begins to change. It requires more compute resources to process. This is when organizations begin to look at cluster computing engines like Hadoop, Hive, Spark, or Presto, depending on the particular use case.

Also, most IT engineers only have experience with static, monolithic applications that have fixed resources, not clusters and workloads, which are more ephemeral.

The LinkedIn Workforce Report for US (August 2018) states that "demand for data scientists is off the charts," with data science skills shortages present in almost every large US city, as demonstrated in Figure 2-6.

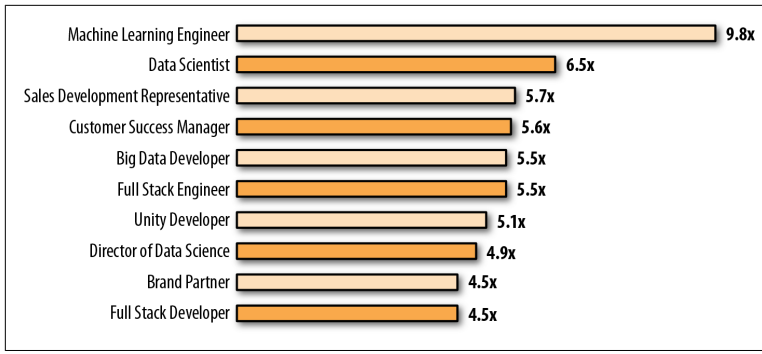

Figure 2-6. Job growth in big data space

Moreover, the technology in the big data space is still young and thus constantly evolving. Though Spark was the most used framework in 2017, as Figure 2-7 highlights, Flink and Presto were catching up with extraordinary market-share growth of 125% and 63%, respectively.

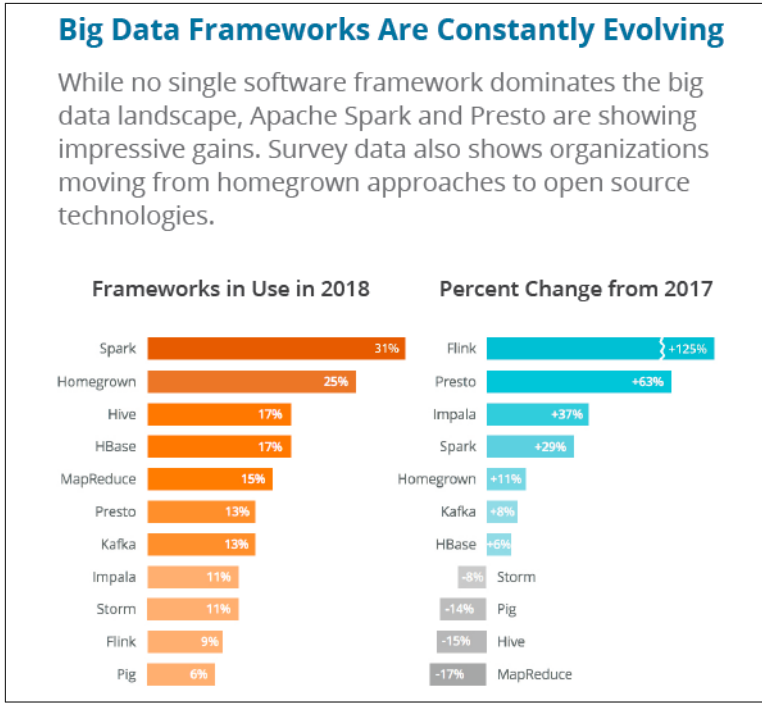

Figure 2-7. Frameworks continually evolving

Disparity and Distribution of Data

Next, the disparity of data is a challenge. The centralized data team needs to be able to capture *all* of the data in the organization. But the number of sources is mind-boggling—and growing every month. Moreover, much of the data is siloed. You might need to find and capture data from within different business applications, product applications, public and private customer interaction points, monitoring systems, third-party data providers, and many other sources.

Also, many enterprise data systems are primarily set up for operational reasons. Collecting data from them is usually an afterthought. The natural inclination of your business is probably not to go out of its way to capture this data, much less expend the effort to consolidate it in one place. Potentially valuable data from all of these sources remains in silos. In the process, the organization loses many opportunities to derive insights or discover optimizations by putting data from different sources together.

Deciding to create a data lake can raise all sorts of questions:

- Where do I store the data?
- In what format should I store it?
- How long is that data useful?
- How do I make it easy for users to find data?
- Who has access to the data?
- What are our permission rights to the data?

The first step toward overcoming these challenges and answering these questions is to take an inventory of all of your data sources and create a company-wide data-capture infrastructure that lays out the correct way to capture and log the data. Everyone in the organization should adhere to those standards.

The next step is to consolidate all of the data in the centralized data lake so that every data consumer in the organization knows where to find it.

Organizational Resistance

Change is difficult—especially a change as massive as moving to a self-service, data-driven company.

The first step is to get the hub-and-spoke data organization in place and ensure that it is aligned with the business requirements. This in and of itself can prove challenging, and given how slowly companies change—especially larger enterprises—it could take many weeks or months.

Next, you need to ensure that you've put the right tools in place to enable self-service for your users. Depending on how sophisticated your users are, these tools can range from canned dashboards and reports to ad hoc querying tools, all the way to fully customizable data platforms. And you must stay on top of whether these tools are actually being used—there's no point in having them if no one pays attention to them.

Your users' level of *data literacy*—their ability to find, work with, analyze, and argue using data—is critical to building a self-service, data-driven culture. Airbnb serves as an example of what you can do to improve enterprise-wide data literacy. Airbnb had a data literacy problem, despite having built a centralized data lake and populated it with massive datasets that the company thought would be useful to employees in the 22 countries where it operates. At the beginning of Q3 2016, only 30% of Airbnb employees used this data platform at least once weekly.

To remedy this, Airbnb built its Data University, with the mission of educating everyone in the company on how to use data effectively. Engineers, product managers, designers—all employees, really—were taught how to use data to unearth insights that would help them make better decisions.

Data University proved a great success, and has transformed Airbnb's culture to one that is both self-service and data driven. Employees learned how to handle ad hoc data requests themselves, and within a year, 60% of them were using the Airbnb data platform at least weekly to make data-driven decisions.

Finally, consider creating some internal centers of excellence around big data analytics. Some of the best data-driven companies constantly reach out to various teams and ask, "How are you doing? What are you using to solve problems?"

You might have centers that are really good at Spark, whereas others are really good at analyses. You might have datacenters that are really good at data governance. With these centers of excellence in place, you can quickly overcome many roadblocks.

Reluctance to Commit to Open Source

It's taken a while, but *open source* is no longer such a dirty word with mainstream enterprises, even fairly traditional ones like financial services or health care companies.

Figure 2-8 shows that "open source software programs play an important role in how DevOps and open source best practices are adopted by organizations," according to a survey conducted by the New Stack and the Linux Foundation.

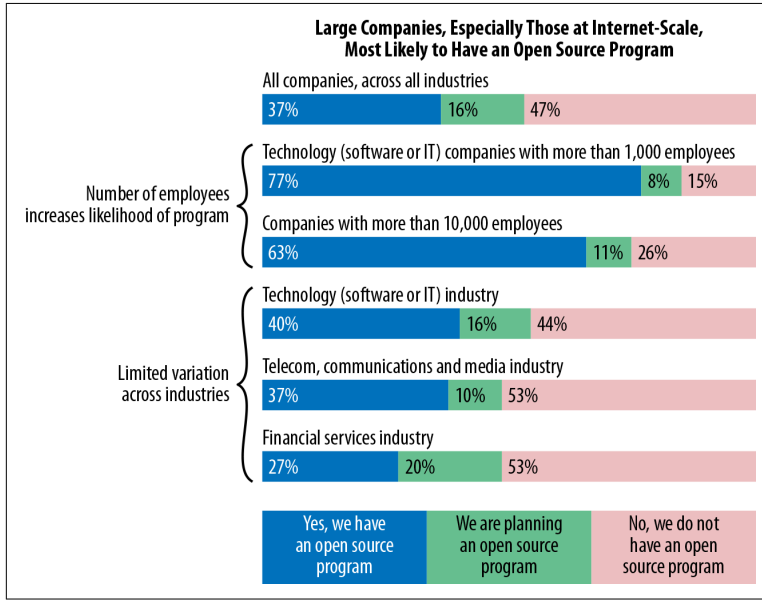

Figure 2-8. Large companies most likely to use open source

In the survey results, more than half of respondents (53%) across all industries say their organizations use an open source software program or have plans to use one. Large companies are almost twice as likely to run an open source program than smaller companies (63% versus 37%). And the number of large companies using open source programs is expected to triple by 2020.

Companies with open source programs also see more benefits from open source code and community participation. As Figure 2-9 highlights, 44% of companies with open source programs contribute code upstream, whereas only 6% of other companies do so.

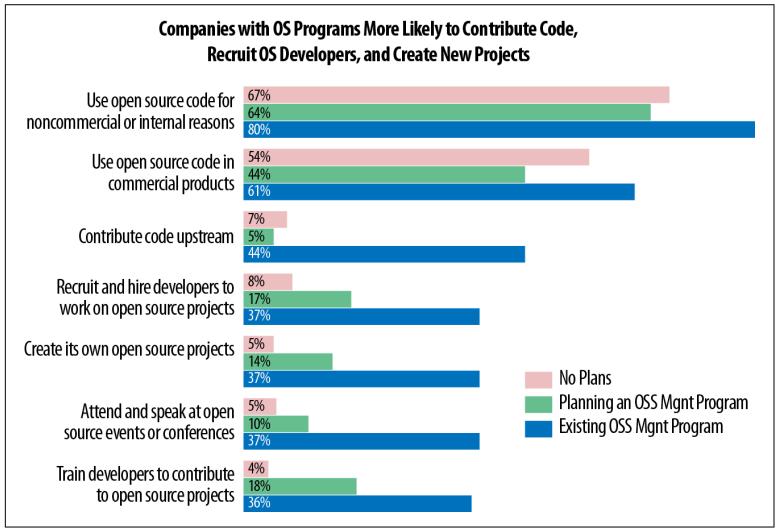

Figure 2-9. Companies with open source programs are more likely to benefit

There are three key arguments for adopting open source software:

- Your company is on the cutting edge of software innovation.
- Your company is able to tap into a huge community of support.
- Your company can "fork" your own version of open source software to build into your applications.

Going the other way—to closed, proprietary big data solutions—you might have clearer roadmaps. You certainly have more structure. And you're paying for stability and enterprise-grade support if something goes wrong. The latter is probably the number one reason: if something goes wrong, you know who to yell at. You have a contract that says the issue will be fixed within four hours. More risk-averse companies tend to stick with proprietary systems. But even that is changing, as big conservative financial institutions like JPMorgan have embraced Hadoop and other open source big data tools.

CHAPTER 3

Getting Started Building Your Data Lake

By now you're probably thinking, "How does the cloud fit in? Why do organizations decide to build their data lakes in the cloud? How could this work in my environment?" As it turns out, there isn't a one-size-fits-all reason for building a data lake in the cloud. But many of the requirements for getting value out of a data lake can be satisfied only in the cloud.

In this chapter, we answer the following questions:

- As your company's data initiatives mature, at what point should you think about moving to a data lake?
- Why should you move your data into the cloud? What are the benefits?
- What are the security concerns with moving data into the cloud?
- How can you ensure proper governance of your data in the cloud?

The Benefits of Moving a Data Lake to the Cloud

The Enterprise Strategy Group asked companies what the most important attributes were when working with big data and analytics.

Not surprisingly, virtually all of the attributes listed were those found when building big data lakes in the cloud, as depicted in Figure 3-1.

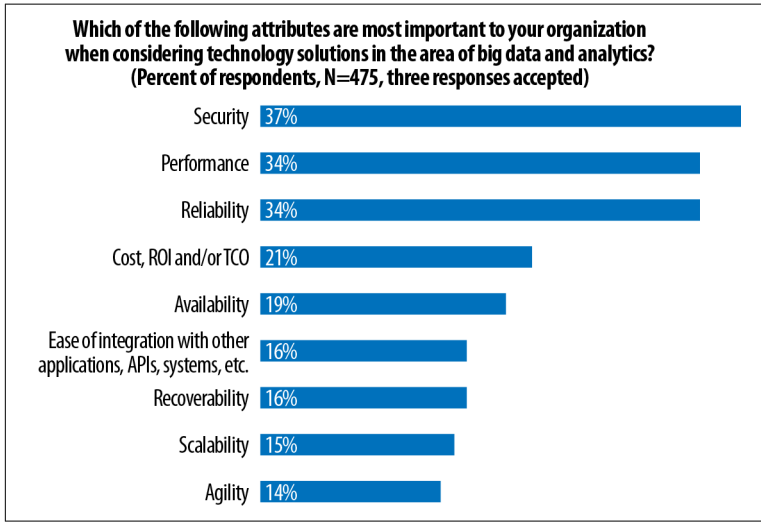

Figure 3-1. Most important attributes when building a data lake

With the cloud come the following key benefits:

Built-in security
: When it comes to security, cloud providers have collected knowledge and best practices from all of their customers and have learned from the trials and errors of literally thousands of other companies. What's more, they have dedicated security professionals—the best in the industry—working on continually improving the security of their platforms. After all, trust in their ability to keep their customers' data safe is key to their success.

High performance
: The resources available from cloud providers are virtually infinite, giving you the ability to scale out performance as well as a broad range of configurations for memory, processors, and storage options.

Greater reliability
: As in your on-premises datacenters, cloud providers have layers of redundancy throughout their entire technology stacks. Service interruptions are extraordinarily rare.

Cost savings
: If you do it well, you'll get a lower total cost of ownership (TCO) for your data lake than if you remained on-premises. You're paying only for the compute you need, and the tools you use are built specifically for the cloud architecture. Automation technologies like Qubole prevent failure and thus lower the risks of operating on big data at scale, which offers a huge value for the DevOps team, who are no longer called in the middle of the night when things break.

A lot of these benefits come from not having to reinvent and maintain the wheel when it comes to your infrastructure. As we've said before, you need to buy only as much as you use, and build only what you need to scale out based on demand. The cloud is perfect for that. Rather than spending all your time and energy spinning up clusters as your data grows and provisioning new servers and storage, you can increase your resources with just a few clicks in the cloud.

Key Benefit: The Ability to Separate Compute and Storage

One requirement inexorably pushing companies toward the cloud is that to work with truly big data, you must separate compute from storage resources. And this type of architecture is possible only in the cloud, making it one of the biggest differences from data platforms built for on-premises infrastructure. Two technologies make this separation possible: *virtualization* and *object stores*.

Virtualization makes it possible for you to provision compute in the cloud on demand. That's because compute in the cloud is ephemeral: you can instantaneously provision and deprovision virtual machines (VMs).

Precisely because compute is ephemeral, the separation of compute and storage is critical for storage of "persistent" assets such as data. This separation is achieved by different storage technologies in the cloud that are persistent: *block stores* and *object stores*. For large datasets, object stores are especially well suited for data storage.

Object storage is an architecture that manages storage as objects instead of as a hierarchy (as filesystems do) or as blocks (as block storage does). In a cloud architecture, you must store data in object

stores and use compute when needed to process data directly from the object stores. The object stores become the place where you dump all of your data in its raw format. Cloud-native big data platforms then create the compute infrastructure needed to process this data on the fly.

Note that this is different from typical on-premises data architectures for big data. Because of the lack of highly scalable object stores that can support thousands of machines reading data from and writing data to them, on-premises data lake architectures stress *convergence* of compute and storage, as illustrated in Figure 3-2. In fact, Hadoop is based on the principle that compute and storage should be converged. The same physical machines that store data also perform the computation for different applications on that data.

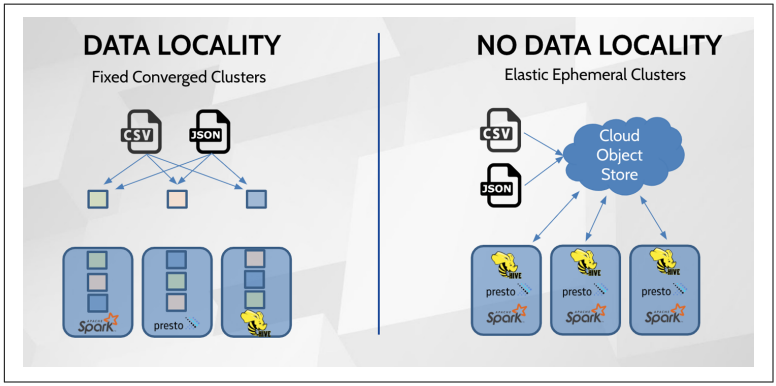

Figure 3-2. The key benefits of cloud-based data platforms: elasticity and separation of compute and storage

Although this is the standard architecture for on-premises big data platforms, lifting and shifting this architecture to the cloud greatly nullifies the elasticity and flexibility benefits of the cloud.

Thus, the cloud allows you to tailor and structure your compute and storage resources exactly the way you need them. This emables you to eliminate worries about resource capacity utilization. It's like using Lyft or Uber rather than owning and maintaining your own car. Instead, a car is a just-in-time resource—just as raw goods are made available just in time in manufacturing—that you can use when you need it. This saves time and money, and reduces risk.

The separation of compute from storage is important because compute and storage have different elasticity requirements. Businesses

don't tend to buy and then throw away storage. If you write something to storage, it's very rarely deleted—not just for compliance reasons (although that's part of it), but because businesses are very reluctant to throw data out.

Compute, on the other hand, is wanted only when needed. You don't want thousands of CPUs sitting idle. That's just wasteful—not just for your organization, but for the environment, too. This is just one of the reasons big data platforms are modernized for the cloud; data inherently has an expiration date on value. You might need to process information only once or a few times. Having elasticity of compute allows for completely efficient use of resources in the analytics life cycle.

This separation of compute and storage also helps you to have true financial governance over your data operations. You can appropriately tune and size your clusters for the given workloads without having to allow for extra compute resources during peak processing times, as you would with an on-premises infrastructure.

NOTE

Some companies are simply lifting and shifting their on-premises Hadoop platforms, such as those distributed by Cloudera or Hortonworks, into the cloud. These companies are actually falling short because they aren't getting the benefits of separating storage from compute and the advantages of being able to spin up ephemeral clusters. Having their platforms in the cloud thus won't improve cost and flexibility results over hosting their data lakes on-premises.

Object storage in the cloud

When it comes to object storage in the cloud, each provider offers its own solution. An object store is a unique way of filing data; whereas traditional systems use blocks of data, object stores make data available in manageable files called objects. Each object combines the pieces of data that compose a file, including the file's relevant metadata, and then creates a custom identifier. This combines the power of a high-performance filesystem with massive scale and economy to help you to speed up your time to insight.

AWS has its *Simple Storage Service* (Amazon S3), which is storage for the internet. Designed to make web-scale computing easier for developers, it has a simple web services interface that can be used to

store and retrieve any amount of data, at any time, from anywhere on the web. You pay only for what you use.

Microsoft offers *Azure BlobStore* as well as Azure Data Lake Storage (ADLS). Azure Blob storage is Microsoft's object storage solution for the cloud; it is optimized for storing massive amounts of unstructured data such as text or binary data. ADLS Gen 2 is a highly scalable and cost-effective data lake solution for big data analytics.

Finally, Google offers *Cloud Storage*. Cloud Storage allows worldwide storage and retrieval of any amount of data at any time. You can use Cloud Storage for a range of scenarios including serving website content, storing data for archival and disaster recovery, or distributing large data objects to users via direct download.

Distributed SQL versus a data warehouse

Another critical decision that organizations face when moving to a cloud data lake is how to migrate their data assets. More often than not, traditional organizations run on online analytical processing (OLAP) systems that have RDBMS and SQL Server backends. These tend to be structured and require data to be persistent with compute. Moving to a cloud object store and then separating data processing from storage can be challenging. Furthermore, certain applications such as enterprise resource planning, financial software, and customer-facing interfaces require that data be highly available.

Data lakes and data warehouses are both widely used for storing big data, but these are not interchangeable terms. Data lakes are vast pools of raw data that have a number of different purposes and can be used for processing later on. Data warehouses (often referred to as *data marts*), on the other hand, store information that has already been processed for a specific purpose; they are intended as repositories for structured and filtered data.

In the strictest sense of the term, a data warehouse is meant to hold a subset of the information stored in the data lake, only in a more optimized and readable format for nontechnical users. Many companies have different data warehouses for finance, manufacturing, ordering, and procurement, for example. These data warehouses are quite often separate from one another because they serve different sectors of the organization. The idea behind a data lake is to bring

these all together in a cohesive system that allows users to query across organizational verticals and obtain more value from the data.

Differences in users

The users of data warehouses and data lakes are also different. Data warehouses are typically operated by less technical users than those using a data lake. That's because data warehouses typically abstract away from the users how data is stored, organized, and optimized.

Data lakes, on the other hand, require much more thought and planning as to where data lives, how it's updated and secured, as well as knowledge of the data's life cycle. Thus, data warehouses are typically used by people who know SQL but could care less about the minutiae of the underlying system. Data lakes are typically managed and used by people who think in computer science terms and are concerned with such things as column-oriented storage, block replication, compression, and the order of data within data files.

In terms of roles and responsibilities, data analysts are typically the target audience of data warehouses. Data engineers and data scientists are the primary users of data lake tools such as Hadoop and Spark. That being said, if a data lake is correctly built, it can serve the needs of all three types of users (analysts, scientists, and data engineers) by supplying an engine for each use case that meets their skillsets and goals.

The data may live in S3, for example, but a cloud-native platform like Qubole offers something for each user with a rich set of engines that can fulfill each user's desired outcome. Data engineers typically use Hadoop, Hive, and Spark; data scientists typically use Spark; and data analysts tend to like Presto. Qubole can support all of these simultaneously.

When Moving from an Enterprise Data Warehouse to a Data Lake

If your business is primarily using an enterprise data warehouse, it is important to ensure that you don't disrupt existing IT operations in the process of building out your data lake platform. A key best practice is to start copying data from your warehouse to your data lake. This creates a foundation for a platform that can scale storage and compute resources separately while figuring out which operations

you can migrate. Your data will then be available across a multitude of use cases while you map out a plan of which teams or data operations make sense to move in the near term. From there, your best next step might be helping to get a successful data product out or creating a new reporting pipeline operation that can demonstrate measurable success to your organization.

How you want to prioritize your plan will be driven by a number of common factors:

Capital expenditure (CapEx) and operational expenditure (OpEx)
: Your business might not want to increase budget on CapEx hardware, so offloading data mining onto the cloud could help mitigate the need to buy more servers.

Analytics bottlenecks
: As more users begin depending on your analytics models to do their jobs, reporting and query volume rapidly increases, which causes bottlenecks on other critical systems or hinders the customer experience. Offloading these users could help stabilize your analytics processes and, subsequently, the business.

Employee resources and expertise
: The company might be hiring more data scientists or analysts, adding to the workloads of the data engineering and DevOps teams who will support them. Focusing on these new and different demands will help you to see the gaps in personnel that you need to fill.

Data analysts (SQL users) are typically the target audience of data warehouses, and data engineers and data scientists are the primary users of data lake tools such as Hadoop and Spark. This said, a well-built data lake will serve the needs of all types of users (analysts, product managers, and business users) by supplying an engine and interface for all use cases that align with users' skillsets and goals.

Deciding whether to start your cloud analytics operations by separating storage from compute will determine whether you go with a cloud data warehouse (such as Redshift) or a distributed SQL engine (such as Presto).

Cloud Data Warehouse

There are two fundamental differences between cloud data warehouses and cloud data lakes: the data types and the processing framework. In a cloud data warehouse model, you need to transform the data into the right structure to make it usable. This is often referred to as *schema-on-write*.

In a cloud-based data lake, you can load raw data—unstructured or structured—from a variety of sources. Most business users can usually analyze only data that has been "cleaned" by formatting it and assigning metadata to it. This is called *schema-on-read*. When you marry this operational model with the cloud's unlimited storage and compute availability, your business can scale its operations with growing volumes of data, a variety of sources, and concurrent querying while paying only for the resources utilized.

With this in mind, some large innovations in data warehouses focus on using data in cloud object storage as the primary repository but can scale out compute nodes horizontally. The most popular of these are Amazon Redshift, Snowflake, Google BigQuery, and Microsoft SQL Data Warehouse.

We describe Redshift in detail, but they all work in a similar manner.

Redshift

Redshift is a managed data warehouse service from Amazon, delivered via AWS. Redshift stores data in local storage distributed across multiple compute nodes, although Redshift Spectrum uses a Redshift cluster to query data stored in S3 instead of local storage.

With Redshift, there is no separation of compute and storage. It does provide "dense-storage" and "dense-compute" options, but storage and compute are still tightly coupled. With Redshift, as your data grows you inevitably buy more resources than you need, increasing the cost of your big data initiatives.

Redshift, originally based on ParAccel, is proprietary and its source code closed. This means that you have no control over what is implemented or what is put in the roadmap. However, on the plus side, with Redshift, you don't need to manage your own hardware. Simply load data from your S3 object store or disk, and you're ready to query from there. This is great for cases in which you are serving

reports or data through a frontend interface from which your users are slicing and dicing different dimensions of the data.

Cloud data warehouse services are great because they're straightforward to use, perform well, and don't require the knowledge and support savviness that is required when you use open source. That being said, ease of use and performance come at a high cost. Given that data remains persistent on disk, adding any new users or data will increase costs exponentially.

Understanding which use cases best fit a distributed SQL technology such as Presto or a data warehouse like Redshift will require you to analyze your needs carefully. You also could find that you have a data pipeline in which your reports are processed by Presto and sent to individual customers via PDF, and then a subset of that data is pushed down to the data warehouse to generate other reports or to feed into customer-facing applications.

Data Lakes and Data Warehouses

We've talked a fair amount about data lakes versus data warehouses. Does this mean that they are mutually exclusive? Absolutely not. More often than not, organizations implement both as part of their overall enterprise data architectures.

A data architect will evaluate the skill level of the teams requiring the data and then build an architecture with the correct engine and data that maximizes the needs and skill levels for the users interacting with that system.

For example, a large conglomerate will have many different organizational units within the company, such as finance, manufacturing, shipping, and legal. Each business unit is typically interested only in the data pertaining to its specific needs.

In the data lake and data warehouse hybrid model, the data lake serves as the primary repository for the organization's data. Because different organizational units within the company need only a subset of the overall data in the lake, data warehouses can be set up to contain just the extracts of data that the business units need. These extracts then can be periodically refreshed from the data lake as needed.

Distributed SQL

If you want the advantages of separating compute and storage, but with a feel that is similar to a data warehouse, Presto is an excellent place to start.

Some important reasons why you might use a distributed SQL engine include:

- You have bursty volumes of users analyzing data regularly.
- Data sizes you're analyzing are unpredictable and large.
- You want to analyze data from multiple sources and data marts.
- You need to join large tables together for further analysis.

In the end, you should have a combination of both. So, your scheduled batches with bursty volumes of data and ad hoc workloads are running in a distributed engine, and your business intelligence analytics or other systems that rely on information are fed into a data warehouse with more uptime.

Presto

Presto is an open source SQL query engine built for running fast, large-scale, analytics workloads distributed across multiple servers. Presto supports standard ANSI SQL and has enterprise-ready distributions made available by services such as Qubole, AWS Athena, GE Digital Predix, and HDInsight. This helps companies on other data warehouses like Redshift, Vertica, and Greenplum to move legacy workloads to Presto.

Presto can plug in to several storage systems such as HDFS, S3, and others. This layer of Presto has an API that allows users to author and use their own storage options as well. As we explained in the previous section, this separation of compute from storage allows you to scale each independently. This means that you use resources more efficiently and ultimately can save costs.

Presto has several other advantages:

Supports ANSI SQL
: This includes complex interactive analytics, window functions, large aggregations, joins, and more, which you can use against structured and unstructured data.

Separates storage from compute
: Presto is built such that each command is passed through a master coordinator that dictates which nodes will run the job through a scheduler.

Supports federated queries to other data sources
: Presto supports querying of other database systems like MySQL, Postgres, Cassandra, MongoDB, and more, so you can bring together disparate data sources.

Performs fast distributed SQL query processing
: The in-memory engine enables massive amounts of data to be processed quickly.

And as we've said, Presto is open source. As such, it has accepted contributions from third parties that enhance it, such as Uber, AWS, Netflix, Qubole, FINRA, Starburst Data, Teradata, and Lyft. Facebook has kept a close eye on the project as well, and continues to contribute its own improvements to Presto to the open source community. Using an open source engine like Presto means that you get advantages from others' work while remaining in control of your own technology.

Because Presto doesn't typically care what storage you use, you can quickly join or aggregate datasets and can have a unified view of your data to query against. The engine is also built to handle data processing in memory. Why does this matter? If you can read data more swiftly, the performance of your queries improves correspondingly—always a good thing when you have business analyst, executive, and customer reports that need to be made available regularly.

How Companies Adopt Data Lakes: The Maturity Model

A TDWI report released in March surveyed how companies are using data lakes and what benefits or drawbacks they're seeing. In the survey, conducted in late 2017, 23% of respondents said their organizations were already using data lakes, whereas another 24% expected to have one in production in the next 12 months.

Given this, how do companies move from traditional on-premises data warehouses to data lakes in the cloud?

Qubole's five-step Big Data Cloud Maturity Model closely correlates with a company's migration to a data lake, as demonstrated in Figure 3-3.

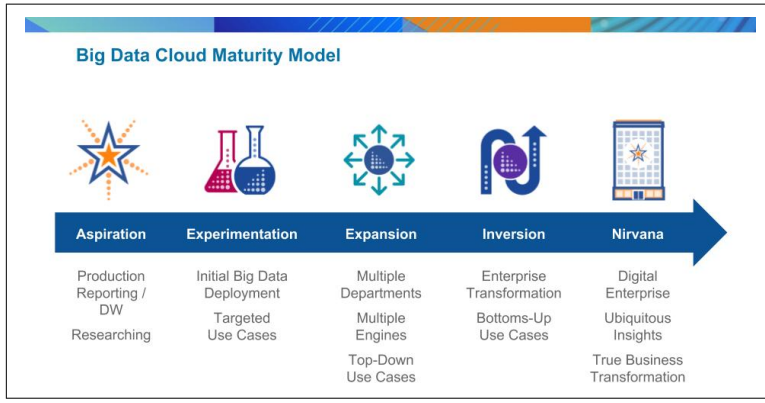

Figure 3-3. The Qubole Big Data Cloud Maturity Model

Stage 1: Aspiration—Thinking About Moving Away from the Data Warehouse

In the first stage, the company is typically using a traditional data warehouse with production reporting and ad hoc analyses.

Signs of a Stage 1 company include having a large number of apps collecting growing volumes of data, researching big data but not investing in it yet, and beginning to hire big data engineers. The company likely also has a data warehouse or a variety of databases instead of a data lake.

The classic sign of a Stage 1 company is that the data team acts as a conduit to the data, so all employees must go through that team to access data.

The key to getting from Stage 1 to Stage 2 is to not think too big. A company should begin by focusing on one problem it has that might be solved by a big data initiative, starting with something small and concrete that will provide measurable ROI. For ecommerce companies, that could mean building a straightforward A/B testing algorithm, or if your company has an on-premises setup, it could be moving some customer reports so that you can deliver them under a faster SLA.

It's important to understand that every new project build is a proof of concept (PoC), whether you are new to data lakes or a pro. Companies should use the power of elasticity in the cloud to be able to test and fail fast to iterate on what technology works best for each use case.

Stage 2: Experimentation—Moving from a Data Warehouse to a Data Lake

In this stage, you deploy your first big data initiative in a data lake. This is typically small and targeted at one specific problem that you hope to solve.

You know you're in Stage 2 if you have successfully identified a big data initiative. The project should have a name, a business objective, and an executive sponsor. You probably haven't yet decided on a platform, and you don't have a clear strategy for going forward. That comes in Stage 3. You still need to circumvent numerous challenges in this stage.

Some typical characteristics of a Stage 2 company:

- Company personnel don't know the potential pitfalls ahead. Because of that, they are confused about how to proceed.
- The company lacks the resources and skills to manage a big data project. This is extremely common in a labor market in which people with big data skills are snapped up at exorbitant salaries.
- The company cannot expand beyond its initial success, usually because the initial project was not designed to scale, and expanding it proves too complex.
- The company doesn't have a clearly defined support plan.
- The company lacks cross-group collaboration.
- The company has not defined the budget.
- The company is unclear about the security requirements.

Whereas the impetus to reach the aspirational stage typically comes from senior executives—where perhaps the CEO or CIO has read about data lakes or heard about them at a conference—the experimental stage must be driven by hands-on technologists who need to determine how to fulfill the CEO's or CTO's vision. It's still very much a PoC.

Although you can certainly build your initial data lake on-premises, it makes sense to start in the cloud. Why? Because you want to fail fast, and you want to separate compute from storage so that you don't have racks upon racks of servers and storage arrays sitting there on the chance that you might find workloads to fill them up one day.

Starting with the cloud means that you have access to elastic resources to do your PoCs and begin training your workforce. They will fail at first—this is inevitable—but they can fail fast and cheaply in the cloud, and they can realize success sooner than if they were on-premises.

Analyzing data from different formats

During Stage 2, you're getting initial datasets into the data lake to meet initial use cases and preparing the foundations of your data lake by doing database dumps in either JSON or CSV format. You're also beginning to analyze log files or clickstream data.

Remember, every build is a PoC. Whether you are new to data lakes or a pro, you want to be able to test and fail fast to iterate your way to success.

> **NOTE**
>
> The JSON library in Python can parse JSON from strings or files. The library parses JSON into a Python dictionary or list. There are various circumstances in which you receive data in JSON format and you need to send or store it in CSV format. Of course, JSON files can have much more complex structures than CSV files, so a direct conversion is not always possible.

Then there's clickstream data. Many platforms such as Facebook and Google Ads rely on the data generated by user clicks. To analyze clickstream data, you must follow a user's click-by-click activity on a web page or application. This is extraordinarily valuable because having a 360-degree view of what customers are and are not clicking on can dramatically improve both your products and your customers' experiences.

Important considerations for the data lake

Here are some common considerations during the exploratory phase of data lake maturity:

1. *Onboard and ingest data quickly*
   One innovation of the data lake is rapid—and early—ingestion coupled with late processing. This allows you to make integrated data immediately available for reporting and analytics.

2. *Put governance in place to control who loads which data into the lake*
   Without these types of controls, a data lake can easily turn into a data swamp, which is a disorganized and undocumented dataset from which it's difficult to derive value. Establish control via policy-based data governance and, above all, enforce so-called "antidumping" policies. You should also document data as it enters the data lake by using metadata, an information catalog, business glossary, or other semantics so that users can find the data they need and optimize queries.

3. *Keep data in a raw state to preserve its original details and schema*
   Detailed source data is preserved in storage so that it can be used over and over again as new business requirements emerge.

Stage 3: Expansion—Moving the Data Lake to the Cloud

In this stage, multiple projects are using big data, so you have the foundation for a big data infrastructure. You have created a roadmap for building out teams to support the environment.

You also face a plethora of possible projects. These typically are "top-down" projects; that is, they come from high up in the organization, from executives or directors. You are focused on scalability and automation, but you're not yet evaluating new technologies to see whether they can help you. However, you do have the capacity and resources to meet future needs and have won management buy-in for the project on your existing infrastructure.

If you're not in the cloud by this time, you should be moving there during this stage.

However, you're still pretty rudimentary at this point. You probably don't have governance in place, and you probably don't have well-structured teams in place, either. You're really just a step beyond performing PoCs and you probably haven't started hiring additional team members. You haven't started reaching out for people who are visionaries or have a lot of experience in this area. Your teams are

still learning. And you probably haven't figured out what storage formats you should have, how well they're optimized, how to convert one to another, or how to order them. This is all perfectly natural.

As far as challenges go, here's what Stage 3 companies often encounter:

- A skills gap: needing access to more specialized talent
- Difficulty prioritizing possible projects
- No budget or roadmap to keep TCO within reasonable limits
- Difficulty keeping up with the speed of innovation

Getting from Stage 3 to Stage 4 is the most difficult transition. At Stage 3, people throughout the organization are clamoring for data, and you realize that having a centralized team as the conduit to the data and infrastructure puts a tremendous amount of pressure on that team. To avoid this bottleneck in the company's big data initiatives, you need to find a way to invert your current model and open up infrastructure resources to everyone, to fully operationalize the data lake and expand its use cases.

Case Study: SolidFire Moves to the Cloud

Flash storage system vendor SolidFire (which was eventually acquired by NetApp) was first running its data lake on-premises in its datacenter using Hortonworks. The first iteration of the data lake took almost a year to stand up due to the burden of having to provision hardware and find space and power in an already-crowded datacenter. After the cluster was up and running and data was flowing in, users were able to start accessing the data through Hue and Hive.

However, the DataOps team at SolidFire was small. As the users grew, keeping up with the numbers and scaling the system became problematic. There were also many usability issues with the first iteration of the data lake. The table schemas for the data were extremely complex, and Hive was the only engine provided to users to interact with the data. This created a less-than-optimal experience for users. At the same time, team members were busy with operational challenges on a daily basis. It wasn't uncommon for the cluster to be 100% used by a few users and members of the DataOps

teams to be up in the middle of the night changing out disks, RAM, or CPUs multiple times a week. Burnout became a problem.

After the acquisition by NetApp and the development of an aggressive growth plan for the SolidFire product, the DataOps team knew the on-premises architecture wasn't going to scale with the business. The team needed to stay small, yet needed to focus more on providing a great data experience. Users wanted more ways to interact with the data beyond Hive, and they were very interested in Spark and notebooks. It was time for a massive change to the architecture, and the team had a short list of requirements for the new platform:

- Multiple engines to accommodate many ways to interact with the data
- Better OpEx and CapEx control through a platform that can autoscale
- A full-featured user interface so people can access the system through a web app or programmatically via an API
- A cloud-native approach that can respond to changing business needs

Ultimately, the team decided on Qubole as the platform going forward. The multiple engines could easily drop in the existing Hive and MapReduce ETL workloads already in production, and users could now choose the engine that best suited their desired way to access the data. Moving to a cloud-native platform also brought several CapEx and OpEx advantages. The DataOps team didn't need to manage hardware any longer, and Qubole's ephemeral clusters coupled with workload-aware autoscaling provided the desired CapEx and OpEx controls. Finally, the full-featured user interface meant that users could use a wide variety of tools to find value in the data.

The changeover from the on-premises cluster to Qubole on AWS was very quick;: around one month. After synchronizing the data from HDFS to S3 and synchronizing the Hive metastore with Qubole, users were given access to the tool and training if they needed it.

The bottom line is that Stage 3 naturally pushes you out of your comfort zone and to the point where you will need to invest in new technologies and to shift your corporate mindset and culture. At this

time, you absolutely begin thinking of self-service infrastructure, and looking at the data team as a data *platform* team. You're ready to move to Stage 4.

Stage 4: Inversion

It is at this stage that your company achieves an enterprise transformation and begins seeing "bottom-up" use cases—meaning that employees are identifying projects for big data themselves rather than depending on executives to commission them. Though this is a huge shift for the company, there is a new challenge now, which is maintaining organization and looking at where to spend smarter or more efficiently across teams' workloads.

You know you are in Stage 4 if you have spent many months building a cluster and have invested a considerable amount of money, but you no longer feel in control. Your users used to be happy with the big data infrastructure, but now they complain. You're also simultaneously seeing high growth in your business—which means more customers and more data—and you're finding it difficult, if not impossible, to scale quickly. This results in massive queuing for data. You're not able to serve your "customers"—employees and lines of business are not getting the insight they need to make decisions.

Stage 4 companies worry about the following:

- Not meeting SLAs
- Not being able to grow the data products and users
- Not being able to control rising costs from growing or new projects
- Not seeing collaboration across teams or data

Still, at this point the company is pretty mature. You know what you're doing. You're now a data-driven organization and your data lake is returning measurable ROI and has become the heartbeat of the business.

You are now concerned about governance: controlling this very valuable asset you now have in place. Governance structures need to be mandated from the top down and strictly adhered to, and you must publish action items that are followed up on every month or every quarter to ensure that your governance activities live up to these standards. You need to update your governance plans and add on

security and reporting procedures. That's why governance must be a living, breathing thing.

You actually need three governance plans:

- Data governance
- Financial governance
- Security governance

We discuss governance more thoroughly in Chapter 5.

Stage 5: Nirvana

If you've reached this stage, you're on par with the Facebooks and Googles of the world. You are a truly data-driven enterprise that is gaining invaluable insights from its data. Your business has been successfully transformed.

Case Study: Large Travel Conglomerate Moves Its Data Lake to the Cloud

One of Qubole's customers is among the largest travel brands in the world—one that has grown tremendously through acquisitions over the past few years. An early data lake proponent, this firm (which requested anonymity) adopted Hadoop and Spark early on. The data team was centralized and worked as a "service provider" to all the departments and business units. Says a senior data scientist for the firm, "We basically had a centralized data science department, but the individual brands and business units each had their own data science teams as well that focused on their specific needs."

The centralized data team owned and operated a massive on-premises 600-node Hadoop cluster with a Teradata system for data warehousing. It was using Hortonworks. The data scientists and the data analysts from all the various divisions and departments would tap into this data lake for their own needs.

But because the company's architecture under Hadoop had the compute and storage coupled tightly, it began to run into limits with its model. As demand grew from the departmental data scientists—who wanted to run queries on the data to solve their own business challenges—compute became scarce. Requests for compute resources were being either pushed down the pipeline or

refused. There were questions of who should bear the cost of purchasing more computer and storage resources.

One department realized that it had to do something. It decided to jump to the cloud.

As part of a PoC, Qubole came in and moved some data to the cloud for one of the business units. After this was set up, this particular team was completely autonomous. It had turnkey access to the computing framework, and because the framework was automated and self-managed, the team no longer had to depend on someone who had the technical skills to run the cluster, because Qubole takes care of that. "At first, it was just the one team using Qubole," recalls the senior data scientist. "And we had full run of the show and unlimited capabilities."

Slowly but surely, other data teams became involved, and eventually the entire company moved to the cloud. "And with that came the bean counters," says the senior scientist. These were the professionals from the finance department who were concerned about financial governance. "And the way our company is set up now is that we've got rules, and we've got tags associated with specific teams and products," he adds. This created further change:

> Each tag gets assigned a budget and someone reports on that budget. If you go over that budget, an email gets sent to a person in charge. Each team also now has a project manager who is usually an experienced data scientist himself or herself. Their job, among other things, is to make sure that projects stay within budget. Because it's too easy to commit cost overruns on the cloud.

Soon the company knew precisely which departments were using what cloud resources and for how long. This was essential given the company's structure, which had to do complicated chargebacks based on which division was using shared resources. On top of everything else, financial governance was achieved, so teams could reinvest their budgets in other projects valuable to the organization.

As news of this new capability spread through the company, soon the team, through Qubole, was providing access to data in the cloud to a broad range of internal "customers" within the larger company. Effectively, it was taking the centralized data team's customers away by meeting their needs better. There was still some administration involved, but the individual teams handled that themselves.

They learned that they didn't need to rip out and replace their entire ETL processes; they simply needed to replicate the data com-

ing into the data lake to the cloud. They internally developed a tool that would help replicate this system from an on-premises HDFS system into Amazon S3.

This enabled the data scientists and analysts and other users to dip into the cloud-native data lake and run their models and queries without any hassles.

CHAPTER 4

Setting the Foundation for Your Data Lake

In Chapter 3, we examined the maturation stages that you will go through as you begin to actualize the value of your data. Now, it's time to look more closely at what it takes to build the data lake.

Setting Up the Storage for the Data Lake

One of your first considerations in building your data lake will be storage. There are three basic types of data storage: *immutable raw storage*, *optimized storage*, and *scratch databases*. The type of data and how you use it will determine which data goes where.

Immutable Raw Storage Bucket

Data kept in immutable storage cannot, and should not, be changed after it has been written. In an *immutable raw* storage area in your data lake, you store data that hasn't been scrubbed. You might never have even looked at it. But it should have sufficient self-descriptive language, or metadata, around it—such as table names and column names—so that you can determine where the data came from. You might store it in a text format such as JavaScript Object Notation (JSON) or comma-separated values (CSV), or perhaps even Apache Avro. Most people choose to store it in either JSON or CSV files.

Immutable raw storage fills many data storage needs. Three of the most important are:

Disaster recovery
: If anything ever happens to the original data stores, you have an exact replica.

Forensic analysis
: Immutable raw storage records can be used to trace problems, such as when bugs were introduced into a program.

Ability to re-create and update optimized formats
: Immutable raw storage ensures that the data is always there in its original state, to be used again if needed.

For example, if your transactional tables are dumped every morning into the immutable raw storage area, you would have snapshots of data, which is very important for the three aforementioned reasons. Financial companies may need these snapshots to review data and see how it has changed over time. You might need these tables if you're ever audited, or if transformed files become corrupted.

After you have an audit table, transactional systems become unwieldy and difficult to manage—that's why you want a raw data bucket. You shouldn't change the data in it; after all, it's raw, static, slow, and pretty damn dirty. But you can query and optimize it.

Types of Data Formats in Your Immutable Raw Storage Bucket

JSON, a lightweight data-interchange format based on a subset of JavaScript, is easy for humans to read and write, and also easy for machines to parse and generate. The following is a quick sample:

```
{
 "title":"Programming Hive"
 "authors":[“Edward Capriolo","Dean Wampler","Jason
 Rutherglen"],
 "isbn-10":"1449319335"
}
```

CSV format files are delimited plain-text files in which each record consists of one or more fields that are separated by commas. The commas separate values of tabular data:

```
title,authors,isbn-10
"Programming Hive","Edward Capriolo,Dean Wampler,Jason
 Rutherglen","1449319335"
```

When you create this raw storage area, remove all "destructive" privileges from the data to ensure that no one can alter or delete it. You also need to do careful access control if the data contains any personally identifiable information (PII).

Optimized Storage Bucket

As your raw data grows, your queries into it become slower. No one likes waiting hours to see whether their query succeeds, only to find that it failed. Data scientists and analysts need their questions answered to turn data into insights faster than that. To gain this speed, transforming your data by storing it using one of the many optimized formats available. Three open source choices are Parquet, optimized row column (ORC), and Avro.

Apache Parquet

Apache Parquet (Figure 4-1) is an open source, column-oriented storage format for Hadoop. Parquet is optimized to work with complex data in bulk and includes methods for efficient data compression and encoding types.

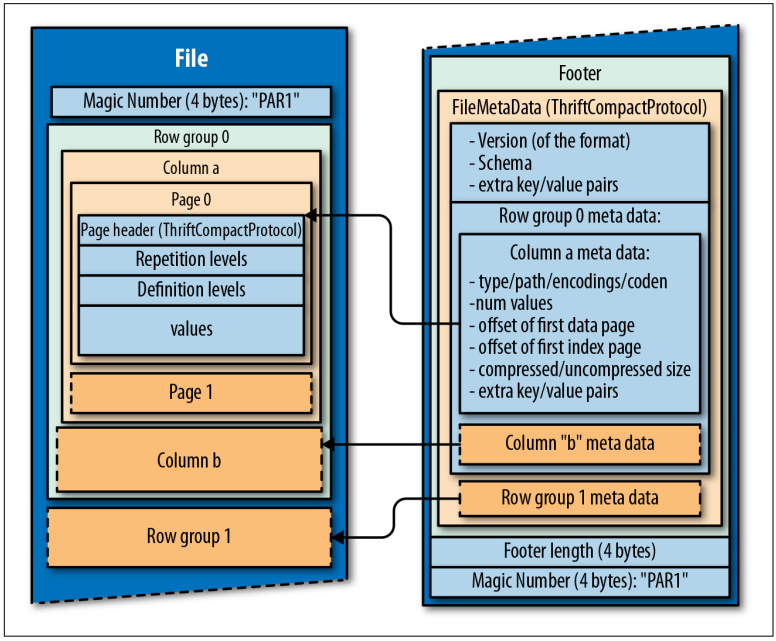

Figure 4-1. The Parquet file structure

ORC

ORC (Figure 4-2) stores collections of rows in one file, with the row data stored in a columnar format. This allows parallel processing of row collections across a cluster. Each file with the columnar layout is optimized for compression. Skipping data and columns reduces both read and decompression loads.

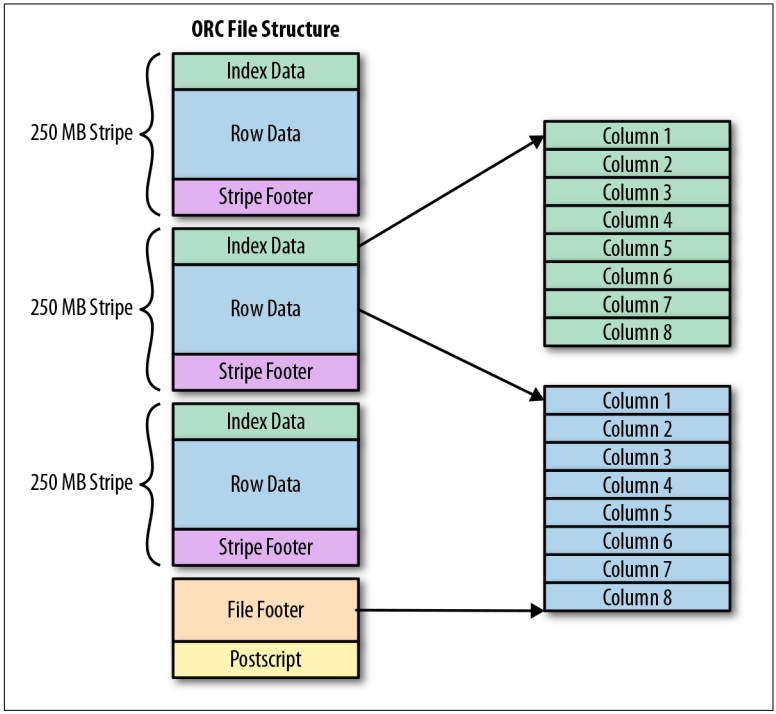

Figure 4-2. The ORC file structure

Avro

Avro (Figure 4-3) is a remote procedure call and data serialization framework. Developed within Apache's Hadoop project, it uses JSON to define data types and protocols, and serializes data in a compact binary format.

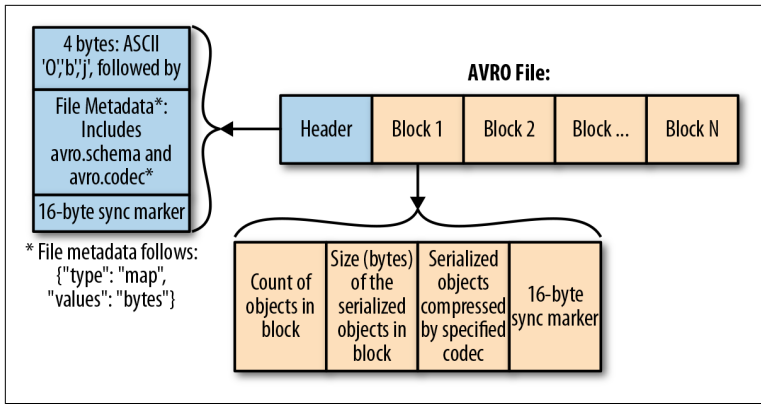

Figure 4-3. The Avro file format

You can split files stored in Parquet, ORC, and Avro formats across multiple disks, which enables scalability and parallel processing. JSON and XML files cannot be split, which seriously limits their usefulness.

All three formats carry the data schema within the files themselves, which is to say they're self-described. You can take an ORC, Parquet, or Avro file from one cluster and load it on a completely different machine, and the machine will know what the data is and be able to process it.

In addition to being file formats, Parquet, ORC, and Avro are also *on-the-wire formats*, which means you can use them to pass data between nodes in your Hadoop cluster. Table 4-1 compares the characteristics of the formats.

Table 4-1. Qualities of ORC, Parquet, and Avro

	ORC	Parquet	Avro
Row or column	Column	Column	Row
Compression	Great	Great	Good
Speedup (compared to text file)	10–100x	10–100x	10x
Schema evolution	Good	Better	Best
Platforms	Hive, Spark, Presto	Hive, Spark, Presto	Hive, Spark
Splittability	Best	Best	Better
File statistics	Yes	Yes	No
Indexes	Yes	Yes	No
Bloom filters	Yes	No	No

Scratch Database

Finally, you will usually create what are called *user scratch databases*. These are necessary because data scientists and analysts will want to take data out of the optimized schema and build test tables for their own purposes. But because you don't want anyone to inadvertently mess up your data lake—or turn it into a data swamp—you need a place where users can have their own little sandboxes to play in that won't mess up the clean, well-defined, and well-structured data in the optimal data space.

For this, too, you need governance. How big can these databases be? How do you monitor them? Do you need an automated report fired off each week to remind people to clean up their data? There's a lot of housekeeping to perform when you have scratch databases in your data lake.

Here are some of the benefits of a scratch database:

- Users can do their work without fear of overwriting sources of truth.
- DataOps can manage resources by user, team, or product.
- The business has the ability to place governance controls at the user level.

The Sources of Data

For many businesses, the primary source of data in their data lake is transactional systems, MySQL, PostgreSQL, and Oracle, among others. These are the frontend databases that interact with your customers. We recommend pulling data out of these databases, converting it to JSON or CSV, and then storing it in your immutable raw storage area where it can be made easily available to your users.

You can also have data feeds that come from internal applications or third-party data services.

The other kind of data is the *clickstream data* coming in from applications, social media, the IoT, or sensors. Those can have any number of formats. Although most applications use JSON for this kind of data, remember that it's coming from your edge, and that it's

probably had very little scrubbing. Again, we recommend putting that into the raw data bucket.

Getting Data into the Data Lake

Because data can be moved directly into the raw storage area of the data lake from a broad variety of sources in a wide range of formats, the data lake becomes a very compelling business tool. Data scientists and analysts now have the "one source of truth" we talked about in Chapter 2, and they can immediately begin querying the data for insights without having to worry about navigating silos or waiting for the data to be modeled and put through ETL processes.

Any refining, structuring, filtering, or preparation of data can happen whenever the data is needed by the business.

You can move data into this raw storage area using a broad range of tools, including ETL tools, bulk file-loading facilities, data integration tools, and even data movement tools that were specifically created for big data technologies. The result is that over time all the raw data—structured and unstructured—from all over the organization can be made available in one place.

Automating Metadata Capture

If you build it correctly, a data lake can add significant value to your data architecture. By providing you with large data storage, processing, and analytics capabilities,it enables you to be very agile when filling it or accessing it. When used together with a data warehouse, it can free up costly resources and more efficiently process large volumes of data.

To work most efficiently, a data lake needs to take advantage of automated metadata capturing and management tools. You need to capture and maintain attributes like data lineage, data quality, and usage history to make the data actually usable, yet doing this requires a highly automated metadata extraction, capture, and tracking facility. Manual metadata management processes cause the metadata to quickly fall out of synchronization with the data itself, turning the data lake, again, into a data swamp.

As discussed in Chapter 3, a well-designed data lake will integrate closely with the enterprise data warehouse and its ETL, data cleans-

ing, and data profiling tools. Your users should be able to run reports and perform analyses using the tools they've always used.

Data Types

You'll have many, many different types of data to work with as you fill up your data lake. These can be divided into three groups: structured (which includes the data you can put into ORC and Parquet); semi-structured (text, CSV, and JSON); and unstructured (images and binary files). The three data types exist on a continuum, with unstructured data being the least formatted and structured data being the most formatted.

The more structured the data, the easier it is to work with, whereas semi-structured and unstructured data create more challenges. Yet all types of data play roles in effective data analysis.

Let's now take a closer look at each of them.

Structured Data

Structured data is data that has been organized into a formatted repository, typically a database, so that its elements can be made addressable for more effective processing and analysis. Some examples of structured data include HR data on employees—for example, names, Social Security numbers, addresses—that is captured and put in separate fields so that it can be easily searched and retrieved. The same is true for sales data; when a customer buys a product, you have a structured record of what that person bought, the date it was purchased, and the price.

Semi-Structured Data

Semi-structured data has some structure to it; for example, it maintains internal tags and markings that identify separate data elements, enabling you to create information groupings and hierarchies. Both documents and databases can be semi-structured. Although this type of data is at most 10% of the data pie, it is used in many business-critical applications. CSV, XML, and JSON are all examples of semi-structured data. Some less common examples include email and fixed-length electronic data interchange formats.

Unstructured Data

Unstructured data encompasses everything that isn't structured or semi-structured data. It has an internal structure but is not structured via predefined data models or schema, and can be textual or nontextual, human- or machine-generated. Text documents and the different kinds of multimedia files (audio, video, photo) are all types of unstructured data file formats.

The reason all of this matters is because a cloud data lake allows you to quickly throw structured, semi-structured, and unstructured datasets into it and to analyze them using the specific technologies that make sense for each particular workload or use case. Table 4-2 compares the three data types.

Table 4-2. Qualities of structured, semi-structured, and unstructured data

	Structured data	Semi-structured data	Unstructured data
Example	RDMS tables, columnar stores	XML, JSON, CSV	Images, audio, binary, text, PDF files
Uses	Transactional or analytical stores	Clickstream, logging	Photos, songs, PDF files, binary storage formats
Transaction management	Mature transactions and concurrency	Maturing transactions and concurrency	No transaction management or concurrency
Version management	Versioned over tuples, rows, tables	Not very common; possible over tuples and graphs	Versioned as a whole
Flexibility	Rigorous schema	Flexible, tolerant schema	Flexible due to no schema

Storage Management in the Cloud

In Chapter 5, we look at how data life cycle management is a policy-based approach to managing the flow of a system's data throughout its life cycle—from creation and initial storage to the time when it becomes obsolete and is deleted. You will need to make decisions when it comes to where different datasets are stored and when to move them to a different storage type or delete them.

A *multitemperature data management solution* refers to one system with different types of data storage and access. In such a system there might be data that is frequently accessed on fast storage (hot data), as well as less frequently accessed data stored on slightly

slower storage (warm data), and rarely accessed data stored on the slowest storage an organization has (cold data).

AWS, for example, has a cold storage format called, appropriately enough, Glacier. Although it's much cheaper than the typical, immediately retrievable storage in S3, you don't have immediate access to data in Glacier. It could take up to 24 hours to get your data out of Glacier and back into your hot storage area.

In fact, Amazon gives you tiers in S3 so that you can have frequent storage access, infrequent access, and probably-never-accessed data (archival data in case you are ever audited).

Data Governance

As your data journey continues, you will need to think of data governance. For example, how do you handle National Provider Identifiers (NPI) such as customers' financial information and PII?

Questions like the following need to be answered:

- Where did this data come from?
- What is the data lineage?
- How has it been transformed?
- What did it look like originally?
- What is the shape of my data?
- How will the schema evolve?

Though many startups don't use credit cards at first for fear of fraud or because of cost, eventually they start storing credit card numbers and customers' PII, and then they're under a different set of guidelines: payment card industry (PCI) regulations.

At this point there is a shift in data governance needs. These startups need to begin adding more change controls with respect to who can access the data and how they access it to comply with regulations like the European Union's General Data Protection Regulation (GDPR) or Health Insurance Portability and Accountability Act (HIPAA) for medical data. The next chapter discusses the topic of data governance in more detail.

CHAPTER 5

Governing Your Data Lake

Now that you've built the "house" for your data lake, you need to consider governance. And you need to do it before you open the data lake up to users, because the kind of governance you put into place will directly affect data lake security, user productivity, and overall operational costs. As described in Chapter 3, you need to create three governance plans:

- Data governance
- Financial governance
- Security governance

Data Governance

When formulating a data governance policy, you'll inevitably encounter these questions about the data life cycle:

- How long is this data good for?
- How long will it be valuable?
- Should I keep it forever or eventually throw it away?
- Do I need to store it because of government regulations?
- Should I put it into "colder" storage to lower costs?

Many enterprises have data that doesn't need to be accessed frequently. In fact, your data has a natural life cycle, and an important

data governance task is to manage data as it moves between various storage resources over the course of that life cycle. Storage life-cycle management is thus becoming an increasingly important aspect of data storage decisions. The cloud offers a variety of storage options based on volume, cost, and performance that you can choose from depending on where in its life cycle your data currently resides.

Public cloud providers like AWS and Azure offer storage life-cycle management services. These allow you to move data to and from various storage services. In most cases, life-cycle management options allow you to set rules that automatically move your data—or schedule deletion of unneeded data—after specified amounts of time.

Although details vary, data-management experts often identify seven stages, more or less, in the data life cycle (see also Figure 5-1).

1. *Create*
   Data enters or is created by the business. This can be from a transaction, from the purchase of third-party data sources, or, increasingly, from sources like sensors on the IoT.

2. *Store*
   If the data is stored in a data lake, it is kept in its raw state, and governance makes sure it doesn't become corrupted, changed, or lost.

3. *Optimize*
   The data is scrubbed, organized, optimized, and democratized to be accessible to the rest of the organization.

4. *Consume*
   This is where analysts, scientists, and business users access the data and perform analyses and transformations on it.

5. *Share*
   Analysts, scientists, and users can share the data analyses and transformations they've created with others.

6. *Archive*
   Data is removed from the data lake. It is no longer processed, used, or published but is stored in case it is needed again in the future.

7. Destroy

In this phase, data that was previously available in hot storage is deleted. This is usually done after the data has been archived.

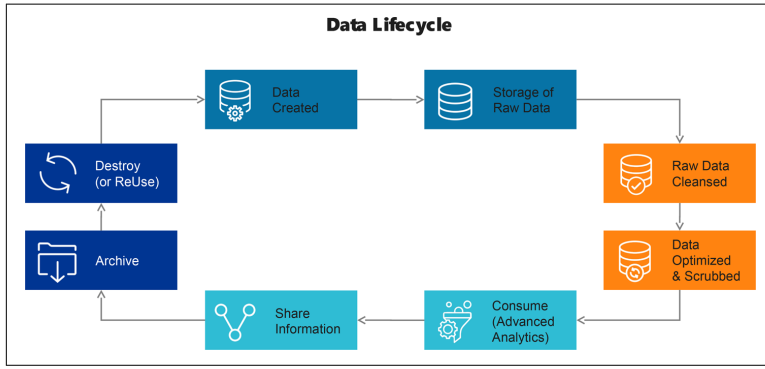

Figure 5-1. The seven stages in the data life cycle

Privacy and Security in the Cloud

Data privacy has become a center of focus because of the ever-increasing requirements of regulations in the industry, including government regulations such as GDPR and HIPAA.

This is especially important when you move to the cloud because you are conceding some control of your environment to a third-party provider. Traditional forms of security that are perimeter-centric are no longer sufficient, because data has become the perimeter in the cloud data lake.

Traditionally, all data operations were kept in-house in your on-premises datacenter, but now that you're using the cloud, the cloud provider controls where the data resides, and it's up to you to manage the rights of the data subjects. For example, under GDPR, you are responsible for protecting the privacy rights of customers, whether it is through anonymization or deletion of their data. You need to keep in mind that your cloud provider might be working with other third-party vendors. In such cases, it becomes essential to look at who controls which people can access what data. You need to ensure that you have the appropriate controls in your application infrastructure, but you also need to take a more data-centric approach to security that offers granular data security using technologies like encryption, masking, filtering, and data redaction.

Depending on your industry, you'll need to think carefully about conforming to HIPAA or PCI regulations. And you need auditability so that you can demonstrate who has access to data, how the data is being accessed or is proliferating, and how you ensure that appropriate controls are in place to demonstrate compliance.

Ensuring that your data lake is secure also requires you to think about data retention. You have two different kinds of responsibilities when it comes to data retention. First, there are the general data retention policies that enterprises must follow as defined by regulations. The best guidelines for these are from the National Institute of Standards and Technology and the International Organization for Standardization. Second, you need to ensure that you put provisions in place so you can delete, purge, or archive data that you've been collecting about individuals or businesses—especially given the EU's GDPR and "right to be forgotten" rules.

Governing your data for privacy and security also requires that you take certain actions. First, you need to be aware of your data. Technologies and tools exist to help you automate this process, so you can scan your data lake to identify exactly what kind of data you have and what's appropriate for your business.

After data discovery, you need to confirm that the intelligence you've gathered can be continuously and automatically augmented. You also need to be able to identify things like data lineage, which gives you the ability to keep up to date on when data moves across the enterprise, when data is transformed, and when data is deleted.

The third aspect is to ensure that this catalog that you created and are continually augmenting is available for users to consume. These could be business users, data analysts, or data scientists, but they will all want the ability to see the data lineage, where the data has come from, and who has done what to it.

Security Governance

One of the biggest concerns businesses have is whether the cloud is secure. One way of reassuring yourself is to first count how many security engineers you have working in your center. One, two, or maybe three?

Then, ask yourself how many security engineers Amazon, Google, or Microsoft have working for them. Probably thousands, if not tens

of thousands. Security is possibly the biggest concern for their existing business model. If companies with data in the cloud can't provide adequate security, no one will trust them.

Most security breaches come from internal vulnerabilities—for example, people who don't use strong passwords, don't change their passwords often enough, tape their passwords under the keyboard, or share their passwords with colleagues. One way of dealing with this type of threat is to segregate your work into different systems. You might have one system for ad hoc reporting, one for canned reports, and one for dashboards. Users are looking at the same information, but in separate systems.

Ultimately, this is where object storage comes in. Remember, object storage is ubiquitous across the entire computer infrastructure. Now you can have that system of record or source of truth, and it can sit on Amazon S3 and it can sit on Blob storage, and everyone is looking at the same data, even though they're not running on the same clusters or the same hardware. This provides different levels of security access to the data.

Financial Governance

Financial governance for data lakes is actually easier when you're using the cloud model than when you're using the on-premises model. The reason is very simple: you have visibility and control.

First, let's talk about financial governance over compute. Compute in the cloud is a logical, not physical, entity. It's not like a CPU in your datacenter. In the cloud, you can actually know at very fine granularity exactly how many compute hours you use. And because of that ability, you can be very specific about what you need—and know how much it will cost. You can also put limits on compute hours so that you can create a policy-based financial governance model that says one group has a certain amount of compute hours, whereas another group has more or less.

In the on-premises world, the notion of the compute hour never existed. There was no need for it. After you bought a particular compute unit or CPU, you had no incentive to actually break it down and see exactly how many hours each particular workload took. This lack of visibility led to a great deal of "server sprawl."

Many businesses find costs growing out of their control in the on-premises world for a number of other reasons. First, scaling compute resources also means buying more storage nodes, regardless of whether you need them, because compute and storage are intertwined (unlike in the cloud). Next, as your investments in hardware and software grow, you need additional engineering resources to manage them. Migrating or upgrading on-premises hardware and software is also costly, and raises the risk of creating siloed data operations as it becomes more difficult for a single team to manage a growing infrastructure and data platform. You also need to invest in disaster recovery measures, which can mean having to buy duplicate systems to locate in a safe remote location.

In fact, the overall costs of running a physical datacenter are high whether you rent or buy. Think of the expenses that accrue for power, cooling, uninterruptible power supplies, and the space itself. It all adds up.

On the other hand, financial governance becomes much easier to achieve in the cloud. Reporting and monitoring is simpler, as is enforcement. This can lead to significant cost savings overall.

Why is that? You now have a metric (compute hours) and you can use that metric to pinpoint where, if at all, cost overruns or abuse is occurring. Combine this elastic infrastructure with a billing model that allows businesses to quickly scale as both data and users increase, and you have an OpEx model that your CFO will love. No more CapEx that can encourage overspending on unneeded capacity. Or, the reverse frequently happens: your data teams run out of resources and must wait for new ones to be procured and set up. It's no wonder that the OpEx model is being enthusiastically embraced by traditional and cloud-native businesses alike.

You will need a plan for governing the costs of data storage in the cloud. Otherwise, you could find your storage bills begin to rise dramatically as data begets data: the more users get into the data, the more they're creating other derived datasets. So, make sure that you keep an eye on your storage bills as well as compute hours. Happily, there are plenty of tools that give you the visibility and control you need to do this.

A Deeper Dive into Why the Cloud Makes Solid Financial Sense

To understand why so many businesses save money in the cloud, let's look more closely at some of the fundamental components of the cloud that enable financial governance. In particular, we define *big data clusters*, *cloud servers*, and *cloud virtual machine clusters* and explain how they contribute to the financial story of the cloud.

What is a big data cluster?

When you request compute resources in the cloud, you are getting a section of a *cluster*. A big data cluster is a collection of machines, called *nodes*, that provide the compute resources. The entire collection of nodes is referred to as the cluster, as illustrated in Figure 5-2. You can easily manage clusters by using one of several available frameworks. Qubole uses YARN's framework for processing and resource allocation with Apache Hadoop, Hive, and Spark engines (see the upcoming sidebar), whereas Presto has its own internal resource manager.

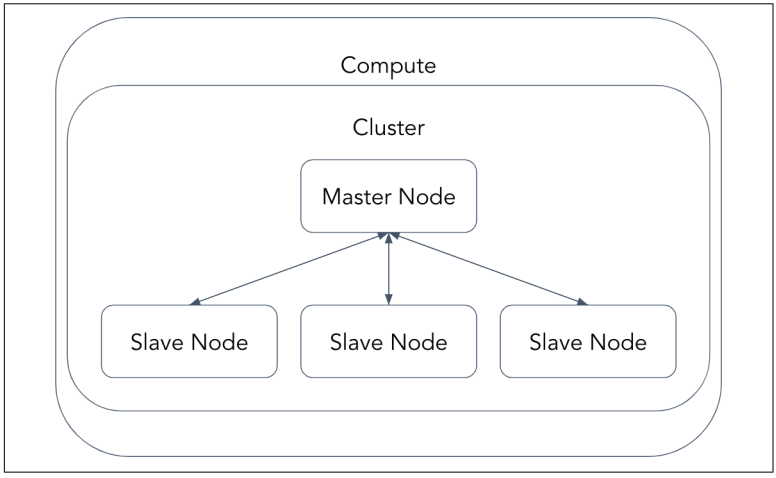

Figure 5-2. A big data cluster

YARN

YARN (Yet Another Resource Negotiator) is a large-scale, distributed operating system designed for cluster management and is one of the key features in the second generation of Hadoop. Basi-

cally, it works like this: within the cluster, YARN understands everything about the CPU and the RAM. It understands how much RAM you need and how much CPU you need, and it monitors that for all jobs coming into the cluster. In effect, YARN divides up CPU and RAM into "containers," and there's only so many that live on a particular cluster. With any given job, YARN will tell you how much of your CPU and RAM have been consumed. The statistic called "containers pending" lets you know what's being processed versus what's in the queue.

Cloud virtual machine cluster

In the cloud, clusters are composed of VMs that reside together within the compute space and are paid for when needed, providing an elastic infrastructure to meet the demands of a business, as shown in Figure 5-3. By using VMs, you get the following:

- Decreased spending and higher utilization
- Capacity to automate infrastructure management
- The right environment and the right tools for each workload and team

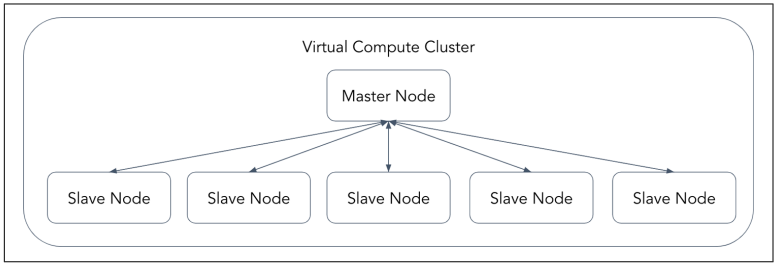

Figure 5-3. How a virtual compute cluster works

Ultimately, this model of having ephemeral servers available to scale up or down dynamically according to the workload demand of various big data clusters provides the foundational model of ensuring financial governance for each of your workloads in the cloud.

Cloud servers

A cloud server is primarily an Infrastructure-as-a-Service–based cloud service model. There are two types of cloud servers: *logical* and *physical*. A cloud server is considered to be logical when it is

delivered through virtualization. In this delivery model, the physical server is logically distributed into two or more logical servers, each of which has a separate operating system, user interface, and apps, although they share physical components from the underlying physical server. A physical cloud server, on the other hand, is also accessed through the internet remotely; however, it isn't shared or distributed. This is commonly known as a *dedicated cloud server*.

How to Mitigate Cloud Costs: Autoscaling

There are several ways of imposing financial governance on your cloud-based data lake. The most efficient way is autoscaling. Autoscaling is a way to automatically scale up or scale down the number of compute resources that are being allocated to your application based on its needs at any given time.

Figure 5-4 depicts this scaling, which is a visual of an autoscaling Apache Spark cluster on Qubole. The x-axis represents a one-month span of cloud servers (nodes) used by hour, and the y-axis is the number of nodes used.

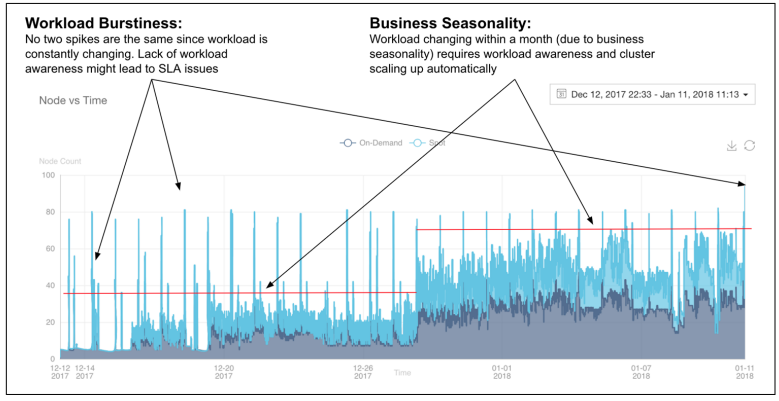

Figure 5-4. Autoscaling on an Apache cluster

Look at the first spikes of blue—that is, the cluster scaling up to around 80 nodes to meet the demand of the workload. When that workload is complete, autoscaling kicks back in and brings the nodes back down to the amount needed to process that data.

Another interesting implication of autoscaling for this workload is the jump in volume that indicates a spike in the seasonality of the

business and thus the need to process more data to meet customer demand.

These servers can be quickly provisioned and turned off, which means that you can have different workloads with different server needs. For instance, in an ad hoc environment, you might want nodes that have a lot of memory to handle the concurrency of your teams' queries, whereas if you are running a massive ETL job, you will likely need a lot of disk space to be able to handle larger volumes of data. The nature of these workloads will also affect the way autoscaling works.

Figure 5-5 depicts the cluster workloads over a one-week period. Zooming in on a 24-hour period shows how autoscaling can significantly reduce the costs of running clusters due to the fact that clusters are upscaled for only a small percentage of the day.

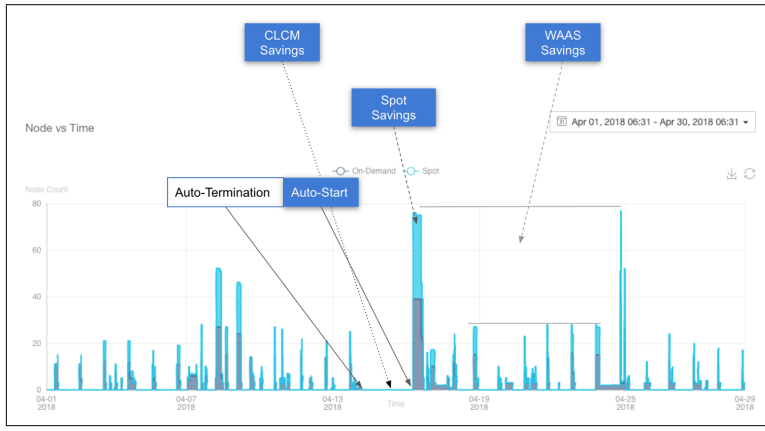

Figure 5-5. Cluster workloads over a one-month period

Spot Instances

Another way to save money in the cloud is to use *spot instances*. Spot instances arise when AWS has any EC2 instance types sitting idle in one of its datacenters. It releases these "spot" instances out to a marketplace where any AWS customer can bid to use the extra capacity. You can usually get away with bidding a fraction of the full value. This is a great way to save money for various ephemeral big data and analytics workloads because you can easily find EC2 instances at up to 90% off the on-demand cost.

In Figure 5-6, the lighter blue shading indicates spot nodes that are running with on-demand nodes in a Qubole autoscaling cluster that has been changed to run Spot on 90% to 100% of the nodes. This allows for this 200- to 1,000-node cluster to run at more than an 80% discount, which is far beyond any vendor commitment discounts from AWS.

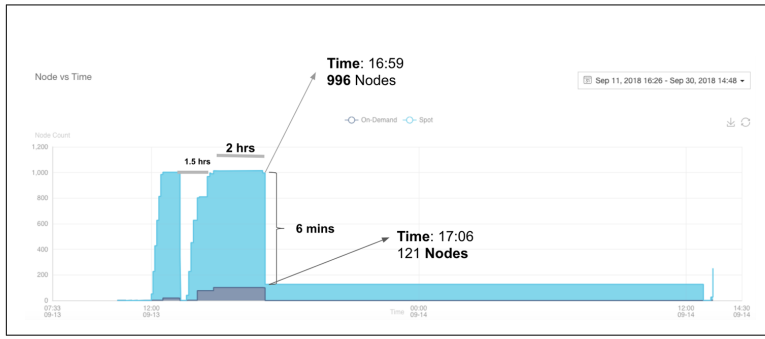

Figure 5-6. Using spot instances to achieve significant cost savings

Using AWS spot instances does come at a risk, though, as someone can easily come in and outbid the spot instances you want if they're willing to pay a bit more. Finding the right configuration based on different workloads requires a bit of skill, art, and occasional luck—although we have noticed that stability and alerting for spot instances has improved significantly over the past few years.

Here' are a few good considerations when using spot instances:

- Spot instances can be taken away at any time. Resiliency must be built in to the pipeline if you're going to use them.
- Availability of spot instances can be lower at certain times of the year. Examples of this are holiday seasons and large sporting events such as the Super Bowl.
- Providing fault tolerance (for example, a high number of spot nodes) introduces volatility into the entire cluster.

Measuring Financial Impact

Ultimately, the cloud data lake—and, more specifically, autoscaling for cloud computing—allows you to segment your spending by workload as needed. This is a huge shift for your teams' budgets, as

you might find spending fluctuates when your teams are investigating or developing new projects, rather than focusing on system optimizations. Here are a few questions to ask yourself regularly when managing analytics in a cloud data lake:

- Are your projects delivering ROIs that push key performance indicators in a positive direction?
- Are you spending well? Are you getting your resources for the best price possible?
- Are you keeping in mind that sometimes it's not about spending less, it's about spending smarter?

Qubole's Approach to Autoscaling

When Qubole started building autoscaling technologies, it evaluated existing approaches to autoscaling and rejected them as being insufficient for building a truly cloud-native big data solution. At the time, and today still, most autoscaling is built on the server level, which makes decision making very reactive and so causes latencies and other inaccurate decisions (such as the estimated size of data volume used in a query, or impact on memory when more queries come in).

To avoid this, Qubole built autoscaling into Hadoop and Spark, enabling it to access the details of big data applications and the detailed states of the cluster nodes. Being workload-aware makes a big difference when your business is trying to orchestrate Hadoop and Spark in the cloud. Figure 5-7 shows how Qubole uses the following features to enable autoscaling:

World-aware autoscaling
: This is automation that allows autoscaling to be malleable in different use cases and clusters based on the workload demands, whether they are bursty or more constant.

Cluster life cycle management
: This is when a cluster automatically starts and terminates (upon idleness). This is a big difference between on-premises and prebuying cloud instances that remain available 24/7.

On-demand instances (nodes)
: These are common cloud servers that are available to any customer immediately.

Spot nodes
: These cloud servers are excess capacity for the infrastructure/datacenter. They're sold at discount prices, but are unstable given that they can be taken back at any moment.

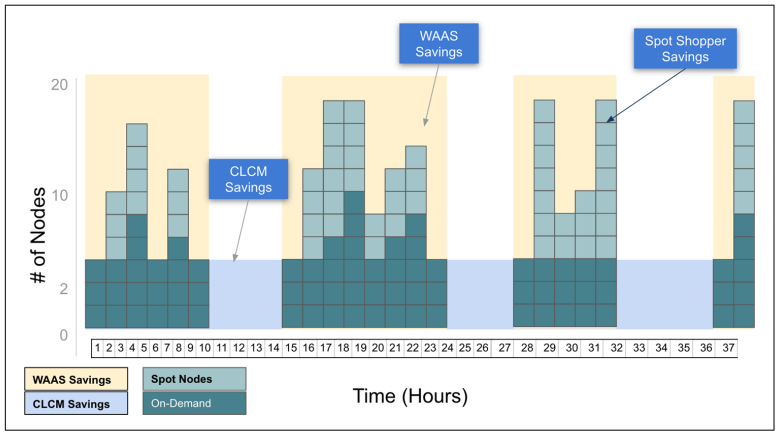

Figure 5-7. How Qubole autoscaling works for cost management

CHAPTER 6

Tools for Making the Data Lake Platform

Now that you've got the frame of a data lake house, it's time to think about the tools that you need to build up the data lake platform in the cloud. These tools—in various combinations—can be used with your cloud-based data lake. And certain platforms, like Qubole, allow you to use these tools at will depending on the skills of your team and your particular use cases.

The Six-Step Model for Operationalizing a Cloud-Native Data Lake

Figure 6-1 illustrates a step-by-step model for operationalizing the data in your cloud-native data lake platform. In the subsections that follow, we discuss each step and the tools involved.

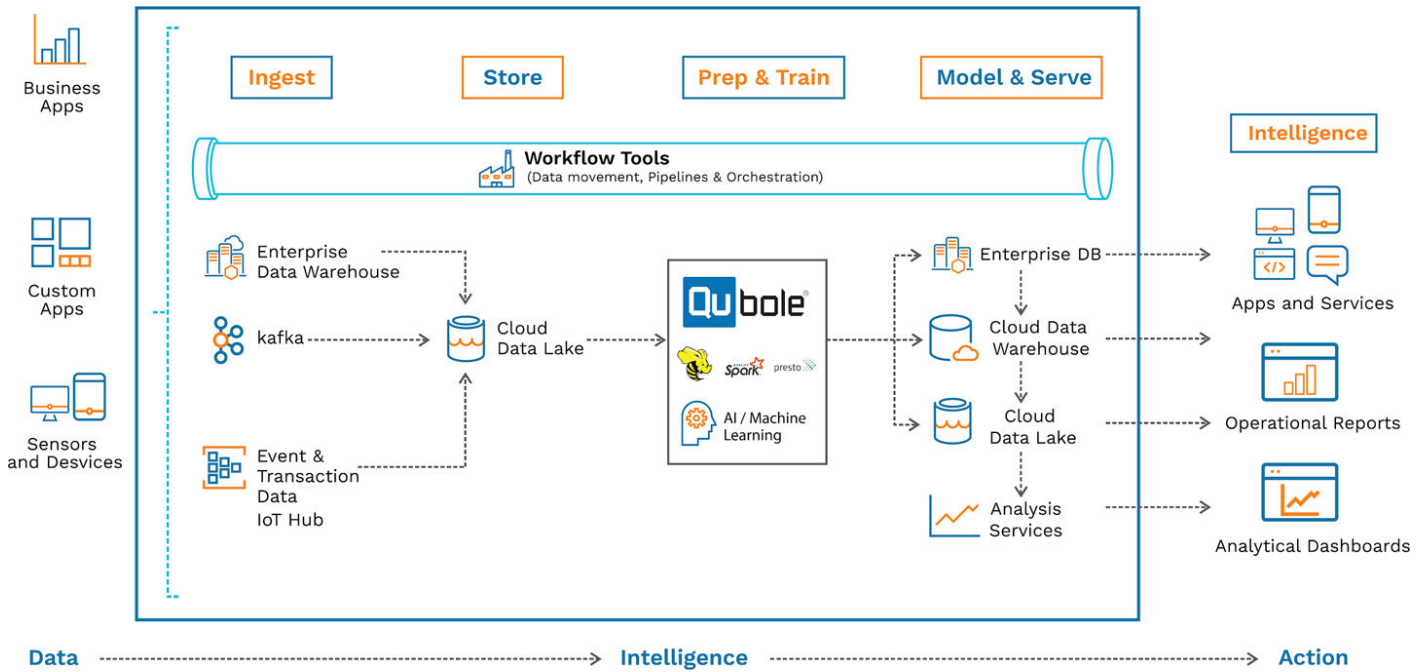

Figure 6-1. The six-step model for operationalizing data

Step 1: Ingest Data

As Chapter 5 discusses, you have various sources of data: applications, transactional systems, and emails, as well as a plethora of streaming data from sensors and the IoT, logs and clickstream data, and third-party information. Let's focus on streaming data. Why is it so important?

First, it provides the data for the services that depend on the data lake. Data must of course arrive on time and go where it needs to go for these services to work. Second, it can be used to quickly get value from data, whereas batch processing would take too long. Recommendation engines that improve the user experience, network anomaly detection, and other real-time applications all require streaming data.

You first need tools that ensure *syncs* or *end points* so we can automatically group the data into logical buckets. In other words, you need tools to help you organize the streaming data in an automated fashion, so you don't have one "big blob" of data; you can separate it into logical groupings. Relevant tools include Kinesis Streams for capturing streaming data from edge devices, and Kinesis Firehose to sync or move that data to Amazon S3 or Redshift. Then there's Kafka plus KStreams and Spark Structured Streaming. Although aggressively promoted by Hortonworks, Apache Flume and Apache Storm are less popular today because they lack the performance and scalability of Kafka or Spark Structured Streaming.

When you first set up your streaming pipelines, it's usually considered a best practice to refresh the data every 24 hours, depending on your business. For example, a financial company probably wants to know what's been happening in the markets within the last five minutes. A shipping company has different needs. It wouldn't make any sense to update its data every five minutes, so relaxing and letting some latency into the pipeline for a few hours would probably be all right for that particular use case.

The "Cold Start" Problem

Consumer-facing applications created by business-to-consumer (B2C) companies, such as ecommerce retailing or banking, often experience something called the *cold start problem*. When a new user signs up for an application, that application has no informa-

tion about the user. But the B2C company wants to keep that user engaged, so it asks for basic information such as name, age, gender, and probably location. This is because the company wants to provide the best user experience possible, even if it knows nothing about that user.

Suppose you are that user. The company must figure out, in a very short time, what is the best content to show you, even though it has minimal information about you. In other words, it needs to categorize you as some segment. As a millennial? As a Californian? As a man or woman? As the application gradually collects more data, the cold start problem dissipates. For example, the first time you log on to and browse through Amazon, the retailing giant knows very little about you. But as you begin to browse, it's recording your keystrokes and purchases, and even where you pause on the website, all data to be used in building your profile. That's streaming data at its finest.

At this point, many businesses are already moving away from the raw batch data derived from transactional systems, applications, or IoT sensors. As you might recall, this raw data is neither cleansed nor formatted. This is just an initial stage for the data.

After the data is moved into an optimal data store in the data lake, it is cleaned and transformed. At this point, operators perform necessary governance, such as stripping out names or Social Security numbers, depending on whether your business is responsible for meeting compliance mandates such as HIPAA or PCI regulations.

You can transform and clean data inline at the same time. There is no need for a separate batch process. And then your users can get into real-time predictions and learning.

At this state, you need to be sure that the data lineage—the data's origins—has been tracked. You need to know not only where it came from, but where it moved over time and when and if anything was done to it.

Apache Atlas (discussed in more detail later in this section) can help you track data lineage. Remember when we talked about all the tribal knowledge held within these data silos? Atlas helps you to concentrate it into one system and expose it to people searching for data. Your users then can search for a particular column, data type, or particular expression of data.

One way to make this search process easier is to assign metadata in plain English, free of acronyms or terms that only a few technical users might understand. Think of the business user, not a member of the technical team, who will be reading the metadata and making decisions and discoveries based on it.

Querying the data stream

First, what is a data stream? In computer science, a stream is a sequence of unbounded data elements made available over a span of time. You can think of a stream as items on a conveyor belt being processed one at a time, in a continuous flow, rather than in large batches, or, to continue the warehouse analogy, a delivery truck periodically dropping off a large load of items all at once. Streams are processed differently than batch data—most normal system functions can't operate on streams, because they have potentially unlimited data.

You can run queries on the data that's in the stream. It wouldn't be the same as querying all of the data in the data lake. You would also need a shorter time interval to query, such as 1 hour or 24 hours. You could ask questions like, how many users signed up in the past hour? How many financial transactions occurred in the past 2 hours?

Streams are not meant to hold massive amounts of data; they're designed to hold data only for a short time, typically from 5 minutes to 24 hours. A streaming platform has three key capabilities. First, it can publish and subscribe to streams of data, like traditional message queues or enterprise messaging systems can. Second, it stores data streams in a reliable, fault-tolerant way. Finally, it can process streams as they arrive.

Apache Kafka. Kafka is an open source stream-processing software platform that started as a distributed message queue for stream data ingestion. Over time, it developed into a full-fledged streaming platform by including processing capabilities as well. Written in Scala and Java, Kafka offers a unified, high-throughput, low-latency platform for handling real-time data feeds.

Apache Kafka is used for two types of applications:

- Real-time streaming data pipelines that reliably get data between systems or applications

- Real-time streaming applications that transform or react to the streams of data

Tools to use for stream processing

The following tools can be used to process streaming data:

Spark Streaming
: Although Kafka is often referred to as the "message bus," Spark can provide a streaming engine in conjunction with the Kafka operation. The same data is going through Kafka; however, you are using the Spark engine to enable processing of live data streams. This is a streaming module of the Apache Spark ecosystem (Figure 6-2) known for scalable, high-throughput, and fault-tolerant stream data processing.

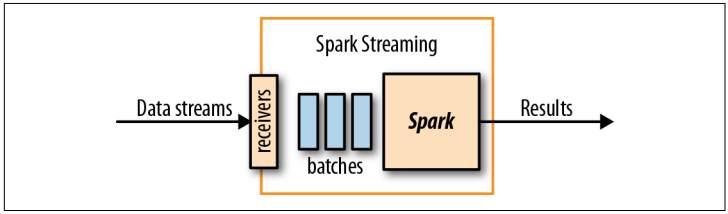

Figure 6-2. How Spark Streaming works

With Spark 2.0, Spark Streaming (now known as Structured Streaming) has evolved significantly in terms of capabilities and simplicity, enabling you to write code for streaming applications the same way you write batch jobs. Internally, it uses the same core APIs as Spark Batch with all of the optimizations intact. It supports Java, Scala, and Python. Structured Streaming can read data from HDFS, Flume, Kafka, Twitter, and ZeroMQ. You can also define your own custom data sources, which could be storage such as the Amazon S3 object store or a streaming database like Druid.

Apache Flink
: Apache Flink is a scalable, high-throughput, and fault-tolerant stream-processing framework popular for its very low-latency processing capabilities. Flink was built by developers from the Apache Software Foundation, most of whom are employed by data Artisans (recently acquired by Alibaba).

The core of Apache Flink is a distributed streaming dataflow engine written in Java and Scala. Flink executes operators in a continuous flow, allowing multiple jobs to be processed in parallel as new data arrives.

There are several key differences between Spark Streaming and Flink. Spark is a microbatch technology that allows latency to be counted in seconds. Alternatively, Flink offers event-by-event stream processing, so latency can be measured in milliseconds. Flink is usually used for business scenarios where very low end-to-end latency is important, such as real-time fraud detection. Spark is usually used for streaming ingestion and streaming processing, for which low latency is unimportant.

A point in favor of using Spark most of the time is its popularity. Data engineers are familiar with Spark for batch and ETL use cases, so they end up using Spark for streaming as well for familiarity and ease of use—unless there is a strong need for very low-latency stream processing.

Apache Druid

This is an open source, high-performance, and column-oriented distributed database built for event-driven data that was designed for real-time, subsecond OLAP queries on large datasets. Druid is currently in *incubation* (see the following note) at the Apache Foundation. Druid has two query languages: a SQL dialect and a JSON-over-HTTP API. Druid is extremely powerful when it comes to running fast interactive analytics on real-time and historical information.

NOTE

What Is "Incubation" for Open Source Projects?

After a project has been created within the Apache Foundation, the incubation phase is used to establish a fully functioning open source project. In this context, incubation is about developing the process, the community, and the technology. Incubation is a phase rather than a place: new projects can be incubated under any existing Apache top-level project.

Step 2: Store, Monitor, and Manage Your Data

As you begin to develop your data lake, the structures in it become well defined.

You now have some governance around what the structures of your data look like. You might now even have a DevOps team that is monitoring the data using tools like LogicMonitor or Datadog (more on these later).

At this point, you're monitoring for operations that might be running out of bounds or not working the way you expect. For data engineers, in particular, dataflow governance is really important. As data becomes more critical to the organization, the data engineering team is in effect the headwaters of the data river. This is the team to which everyone looks if something goes wrong. And if the data doesn't flow because the ETLs don't happen, or if the data isn't processed correctly, increasingly serious problems can occur downstream—potentially all the way to the senior executives, or even the CEO.

Monitoring your data lake

Monitoring is a critical function for any successful data lake. Monitoring is a broad-ranging subject. Different types of users—DataOps professionals, data engineers, data scientists, and data analysts—all have different requirements for monitoring. For example, DataOps professionals might be looking at resource usage, data ingest rates, and overall CapEx and OpEx efficiency as well as cloud provider health. Alternatively, data engineers might be looking at SLA objectives for ETL pipelines and data-quality reports. Data analysts and data scientists might be interested in the data-quality reports so that they can have confidence in the data provided to them.

Modern monitoring systems provide a rich set of services such as dashboarding, anomaly detection, alerting, and messaging.

Reports and anomaly detection. Monitoring services for cloud-scale applications are extremely important to build into your systems at this stage. They provide dashboarding capabilities to visualize the health of your servers, databases, tools, and services through a SaaS-based data analytics platform. Management tools with cutting-edge capabilities such as collaboration, workflow, and dashboarding include LogicMonitor, Datadog, and VictorOps.

Alerting. Alerting is one potential output of a monitoring system, drawing attention to an issue based on predetermined metrics so you can take action to resolve the problem. Modern alerting plat-

forms let you program on-call rotations as well as escalation logic. Leaders in this space include PagerDuty, Opsgenie, and VictorOps.

Messaging/communications. Monitoring systems typically have messaging and communication components that allow your team to collaborate. Whereas the alerting system can alert a specified group or user if a specified system threshold is passed, the messaging and communication capabilities help your team resolve issues quickly and effectively. Another benefit of a centralized communication system is that it gives you the ability to search historical discussions for solutions if issues recur. Leaders in this space include Slack, Google Groups, and Discord.

Tools for managing data services

Data services play an important role in managing an organization's critical data. These tools are responsible for handling data ingestion, preparation, federation, and propagation in the data lake.

Many times, these systems can directly impact the overall success of a big data project by making it easier for users to discover and have confidence in the data early in a project life cycle. Data services also allow administrators to enforce governance by obfuscating sensitive data or controlling which users can access which data.

Apache Ranger: Security

Ranger gives businesses the ability to enforce granular data access controls. It's a framework that lets you define policies to specify who can access what data and in what context. For example, you want HR staff to see payroll data, but not marketing. Ranger also enables you to limit and enforce when data is accessed.

There are several benefits to using a tool like Ranger for managing data in your data lake. First, you have a place to centrally define policies and enforce them. Second, Ranger offers a level of granularity not previously attainable. Historically, you had to apply granularity at the table level, not the row or column level. Ranger offers a mechanism to apply data access controls at the row or column level as well as data protection using techniques like *masking*. It can also integrate with third-party solutions to offer a more granular encryption solution. Ranger is a good tool for auditability as well: you can easily demonstrate compliance with the security rules that you have in place.

There are drawbacks, however. Critics of Ranger point out that it is a solution in an evolving space. The technology of big data changes rapidly, with a variety of acts and mechanisms. Unfortunately, Ranger currently restricts access or applies controls to only a subset of access mechanisms. For example, it is suited for SQL-like query access to data, but this can be limiting when users use other access methods such as Java, R, or Python scripts. As granular access controls become more critical, we are seeing new solution offerings in the market to address these gaps.

Best practices

Here are some of the most effective ways to use Ranger:

- It's important to look at data protection as a variety of layers. Don't expect Ranger to be the one solution that offers complete protection; it should be just one of the protection controls that you enforce.
- Because the Ranger policies you set affect your business users, always begin by trying simple policies for auditing, then follow with more data access controls like row and column filtering, and finally look at a much stronger mechanism to enforce protection like masking or encryption. Proceed gradually.

Apache Atlas

Apache Atlas is a low-level service in the Hadoop stack that provides core metadata services.

Atlas supports data lineage by helping people understand the metadata for the data. It's intended to provide a discoverability mechanism for people to better parse the data.

Atlas concentrates all the tribal knowledge held within all these data silos and exposes it to the people searching for data. Users can search for a column, a data type, or even a particular expression of data.

Step 3: Prepare and Train Data

Data preparation is important at the data lake level during the ingestion process. If data is not transformed and cleaned here, you will need to do it downstream in other data pipelines, which can cause inconsistency or duplicate workloads. You need to focus on prepar-

ing data to put into a data lake that's accessible for one source of truth; in effect, this allows your data teams to speak the same language about data and easily consume it. We can break down data preparation into four main areas:

- Data discoverability
- Data preparation
- Data fusion (augmentation)
- Data filtering

Let's take a closer look at each of these areas.

Data discoverability is important during data preparation because as your organization grows, it can be very time-consuming to try to identify what data is available. You need to make it easy for users to log into a data portal and see what data assets they can access. It is also critical that your users have a high degree of confidence in the data that you are putting into the data lake.

Data preparation is not one step, but a combination of many steps. For example, ingestion basically means consuming data from sources. That's the first step in data preparation, which is ensuring that the data is coming in. It might be coming from multiple sources, and you need to connect to all of them and get the data, which is essentially raw, into a new system (data lake).

This data is not yet ready for direct consumption, which is where sanitizing comes in. This is also known as data "cleansing," in which you remove redundancies and incorrect records and values.

Data fusion, also referred to as *data augmentation*, is when you fuse data from one source with data from other sources. Implementing good master-data management and data-preparation practices enables users to more easily fuse disparate data sources. Then, having a single shared repository of data and breaking down data silos means that users can extract value from more types of data in a much shorter timeframe.

Data filtering comes in during data preparation, when the data team helps you filter out everything but the data that's useful to you. Some groups want only a subset of the data. For instance, if you're working in a big bank and you're in the equities group, you might not want to see bond data.

This is especially important to companies that must comply with a lot of data privacy or security mandates, like health care organizations or large banks. For example, the latter has tightly controlled structures, so data could be coming from the mortgage capital division, the equities division, the bonds division, or wealth management. Users might need different segments of data that come out from the general data lake in which this data has been stored. You typically need to filter based on your requirements. When you filter, the datasets are refined into simply what you need, without including other data that can be repetitive, irrelevant, or sensitive.

The Importance of Data Confidence

Data preparation is all about sanitizing, filtering, and cataloging the data to capture its lineage. Why do you need to do this? Because you'll always have issues related to data quality. How confident are you in the data? Do you have a clean set of the data? Is the data complete and was it processed correctly? These are probably the most important questions you'll hear from users within your company.

Data preparation is all about ensuring the quality and consistent timely arrival of data. The absolute last thing you want is the CEO coming to you and asking, "Where's my data?" or worse, "Where did these numbers come from? The CFO thinks they're wrong."

That's not fun, because then you must scramble and go through all the different channels and groups that might have touched that data before it got to the CEO.

Tools for Data Preparation

Hive Metastore is a service that stores the metadata for Hive tables and partitions in a relational database. It then provides clients (including Hive) access to this information using the metastore service API that will be used later, during SQL query processing.

Apache Atlas, as we've discussed, is a scalable and extensible framework for data governance including metadata management and potentially also data usage and lineage. It allows enterprises to define and classify data and build a unified catalog of governable data assets. This unified catalog can be used by data analysts and scientists for their data analysis as well as by engineers and IT adminis-

trators in defining data pipelines. Security personnel can use it for audit and compliance monitoring through data usage and lineage.

Apache Hive

Hive was originally designed to be a data warehouse on top of Hadoop/MapReduce. It combined SQL-based language that allowed people to query unstructured data with a metadata system that allowed definition of tables and schemas on the data. Hive was invented at Facebook, where these capacities were needed, as analysts were having trouble accessing meaningful business questions on ever-growing datasets. Two lead engineers at Facebook, Joydeep Sen Sarma and Ashish Thusoo, who together eventually founded Qubole, saw the potential in Hadoop to enable this capability and created Hive as a result. They decided to build a SQL interface to Hadoop, because SQL was the language most popular with their data teams. Over time, SQL has become easy to combine with business intelligence (BI) software, and is a common gateway to even simpler dashboard interfaces for analyzing data.

The main building blocks are the SQL language built on Hadoop and the Hive Metastore, which is becoming the de facto schema for the data lake. Today, Hive is used widely to query against cloud storage to perform large-scale ETL processing against data.

The Hive Metastore has become a foundational component of many enterprise data platforms today. The Hive Metastore provides a layer of metadata for tables (such as schema and location) that is used to query against different sources in the data lake. Even though the number of SQL implementations has exploded, all of them—Presto, SparkSQL, Impala, Drill, and so on—continue to use the Hive Metastore as the central schema in the data lake.

How Hive Works

When Hive was initially built, MapReduce was already a very popular processing engine. Here's how MapReduce works: when you give it a very large amount of data, MapReduce cuts it into small pieces and then runs small amounts of computation on every one of the pieces. Then it collects all the data and does what you've instructed it to do. In essence, the "map" in MapReduce breaks up the data, running some small amounts of compute on it, and the "reduce" part collects all of that data and performs whatever task you instructed on the collected data.

MapReduce was not implemented by Hive, nor did it come with Hive, but was something Hive could use to generate code. Still, Hive itself is MapReduce agnostic. It was always possible to generate code for other sorts of processing paradigms. And that's what ended up happening. Today, if you look at Hive, MapReduce is, to a large extent, legacy and has been replaced by newer engines like Spark and Tez.

A key benefit of Hive is scale. Hive can analyze very large amounts of data at very high processing speeds using the Hadoop system underneath.

A second, very distinctive benefit of Hive is enabling *schema-on-read*. This allows you to store data in any format, allocate a schema on the fly, and read it without having to store it.

One important difference between the data lake and traditional data warehouses is that there's no data loading and no data conversion. You store data in whatever format it comes in, and then you define how it should be processed and how it should be represented.

Summing up, here are some of the advantages of Hive:

SQL-like interface for distributed computing
: Powerful SQL support for Hadoop/MapReduce workloads.

Dynamic user capabilities
: Ability for users to define business logic in code with custom user-defined functions.

Mature project and community support
: Hive's engine has a robust ecosystem built for data governance, monitoring, and security.

Built for petabyte scale
: Scalable engine that is built on top of the familiar Hadoop framework.

Step 4: Model and Serve Data

The more you can empower users by lowering the barriers to access data and the right engines to process said data, the faster you will derive value from your investment in big data technologies.

In particular, moving to a self-service culture—as described in Chapter 2—will free up time for your data scientists and engineers

to begin using advanced analytics and AI tools such as machine learning instead of worrying about provisioning infrastructure resources. This shift is essential if you want to reap the competitive benefits of big data in the data lake.

First, let's examine some definitions and talk about some ways in which you can use machine learning in the data lake.

Machine learning defined

Machine learning is a specific type of AI technology that enables systems to automatically learn and improve from accessing data and having "experiences." Many experts today consider machine learning to be a "productized statistical implementation."

By creating a machine learning algorithm, data scientists are basically teaching a machine how to respond to inputs that it has not seen before, while still producing accurate outputs. Machine learning is often used to learn from big data and forecast future events based on that data.

Machine learning is becoming quite common, and many of our day-to-day activities are affected and informed by it. Take Netflix, which predicts what other movies or television shows you might like depending on your past viewing history. Or Amazon's famous recommendation engine, which makes shopping suggestions based on your purchasing patterns. These both utilize machine learning.

Apache Spark

As we mentioned earlier, Apache Spark is an open source distributed general-purpose, cluster-computing framework.

Two of the biggest benefits of Spark are its scalability and speed of processing. Until Spark came along, machine learning was not practically scalable and took too long. Spark also accommodates multiple languages. Developers can write in Scala or Java, and data scientists can program in Python or R. There is ease of use in terms of computational development for all the different data personas.

The open source community supporting Spark is large, which means innovations are continuously happening and being shared with the community. New features are constantly being added, and the stability is increasing as well.

The API layer of Spark, which helps users write applications, is very intuitive. Spark offers those APIs in various languages, such as Python and Scala; analysts can use SQL, and statisticians can use R, and distributed R in the form of Spark R. The ecosystem is quite rich.

Although the Spark core aims to analyze the data in distributed memory, there is a separate module in Apache Spark called Spark MLlib for enabling machine learning workloads and associated tasks on massive datasets. With MLlib, fitting a machine learning model to a billion observations can take a couple lines of code and take advantage of hundreds of machines. MLlib greatly simplifies the model development process.

Spark has several other advantages, among them:

Is designed for fast distributed processing
: Spark's engine enables massive amounts of data to be processed quickly, in-memory or with batch.

Supports multiple languages
: Spark allows users to code with SQL, Python, R, and Scala.

Offers a robust data science ecosystem
: Spark allows you to easily use custom or prebuilt packages for machine learning and advanced analytics use cases.

Handles a wide variety of workloads
: Spark enables you to use streaming for near-real-time data processing, batch processing, and running ad hoc queries across various data sources.

Use cases for machine learning

Here are some of the ways in which machine learning can be used in a real-world business setting:

Deploy AI-powered robotic process automation (RPA)
: Cognitive RPA combines process automation with machine learning. RPA by itself is good for basic, repetitive rules-based tasks such as streamlining HR onboarding procedures or processing standard purchase orders. When machine learning is added to it, RPA can do more sophisticated tasks like automating insurance risk assessments. By augmenting traditional rules-

based automation with machine learning, RPA software robots ("bots") can even make simple judgments and decisions.

Improve sales and marketing
: Sales and marketing operations generate huge amounts of unstructured data that previously went untapped. For example, companies are using machine learning to do customer sentiment analysis based on remarks made in social media or on sales calls as well as forecasting sales and customer churn based on detecting complex patterns of customer behavior that would otherwise go unnoticed.

Streamline customer service
: When coupled with other AI techniques such as natural language processing, machine learning and big data have the opportunity to transform customer service. Already, customers interact with companies using chatbots and virtual digital assistants, and the huge quantities of data that these interactions produce that are ready for analysis boggles the imagination. Such "virtual agents" today use machine learning algorithms to parse customer questions or statements about problems and provide a speedy resolution—problem-solving that gets better as more time passes and more data is available from which the system can learn.

Bolster security
: Machine learning can also help companies enhance threat analyses and prevent potential security breaches. Predictive analytics can detect threats early, and machine learning enables you to monitor the millions of data logs from IoT devices and learn from each incident how best to thwart attacks.

This isn't to say that machine learning is without challenges. According to the 2018 Qubole big data survey, many obstacles impede the effective use of machine learning. The number one obstacle is learning how to analyze extremely large datasets (40% of respondents), followed by being able to secure adequate resources—including expert staff (see Figure 6-3).

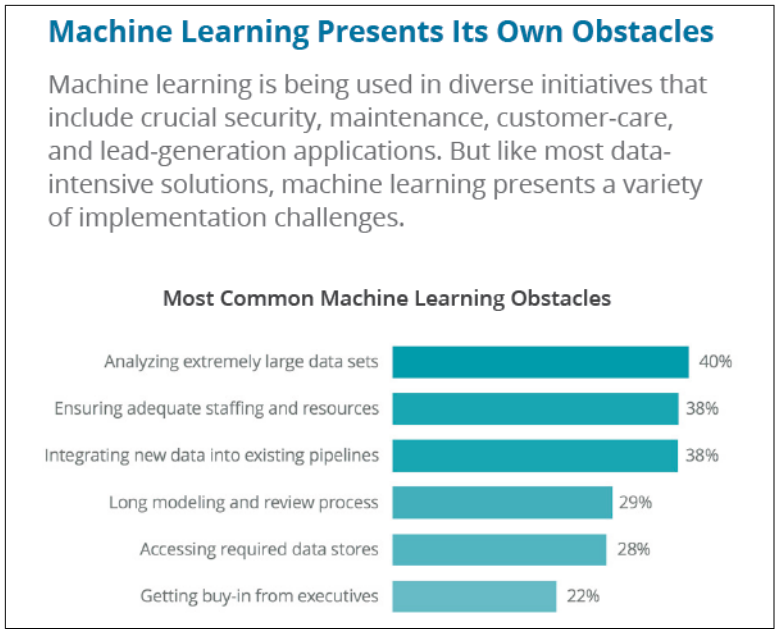

Figure 6-3. Common challenges with machine learning

The Four Phases of a Developing a Machine-Learning Model

According to Piero Cinquegrana, senior data science product manager at Qubole, there are four phases of building a machine learning application:

Phase 1: Data prepping

The first phase is the responsibility of your data scientists. It includes exploring the data, becoming acquainted with the data, visualizing it, validating its quality (QA), and ensuring that the data is the correct dataset to use for the particular problem you are trying to solve. "And even prior to this, of course, there's a whole prioritization exercise and business validation," says Cinquegrana. "That goes without saying."

Phase 2: Model building

A machine learning model is a mathematical representation of a real-world process that you then can use to make predictions. The task of model building also is the responsibility of the data scientist,

and involves selecting different "estimators" that can be applied to the problem and then training the model.

Phase 3: Model validating

The next step is to validate the output of the model, typically with a partner from the business side—the stakeholder who is seeking a prediction. Does this prediction make sense? Does it defy expectations? These are the questions you ask at the validation stage. This is generally a very experimental, iterative phase during which you go back and forth, changing queries, adding features, and trying different models.

Phase 4: Model deploying and monitoring

This final exercise becomes one for the machine learning engineer, who takes the validated model and moves it into production. Monitoring the model is also very important. Suppose that you've put a model for recommending ebooks on an ecommerce website into production. How's that recommendation model doing? How many of the users are clicking the recommended products? Is it performing as expected? Is the performance degrading or improving over time? Those are the questions that a data engineer would be most concerned about. Then, if the model is performing well, the questions focus on how to improve it by providing more features—and the cycle starts over in Phase 1 with the data scientist.

Tools for Deploying Machine Learning in the Cloud

When it comes to deploying machine learning models, a number of technologies have recently emerged to help provide data scientists with more self-service capabilities that enable them to take their models into a production pipeline.

Open Source Machine Learning Tools

In the open source world, a few of these tools that have gained popularity recently are MLflow, Kubeflow, and MLeap, which focus on cataloging and managing models that are deployed into production for large-scale datasets.

Managed Machine Learning Services

DataRobot, Dataiku, and H2O are managed services to help data scientists build and train machine learning models for predictive applications. These companies are also helping provide the new wave of "Algorithms-as-a-Service," which aim to provide point-and-click solutions for machine learning across a wide variety of verticals and data. These companies also provide capabilities to manage models to deploy.

Cloud Machine Learning Services

Lastly, cloud providers are also beginning to offer solutions that manage the end-to-end workflow for data scientists. Recently, AWS released SageMaker, a tool that was designed to make machine learning easier for novices to develop models. SageMaker does this by providing common, built-in machine learning algorithms along with easy-to-use tools for building machine learning models.

SageMaker supports Jupyter Notebooks and includes drivers and libraries for common machine learning platforms and frameworks. Data scientists can use SageMaker to launch prebuilt notebooks from AWS, customizing them based on the dataset and schema they want to train. Additionally, data scientists can take advantage of custom-built algorithms written in one of the supported machine learning frameworks or any code that has been packaged as a Docker image.

Traditionally, data engineers have had to rework models built by data scientists before embedding them in production-quality applications. SageMaker hosting services enable data scientists to deploy their models independently by decoupling them from application code. SageMaker can pull virtually unlimited data from Amazon S3 to train.

Step 5: Extract Intelligence

What helps people understand data best is pretty pictures and charts. That's because we're human and can take in only so much information. It really is true that a picture is worth a thousand words.

Databases are the cornerstone of how we store and interact with data on a daily basis. These systems have been evolving and matur-

ing since the 1970s and support a wide range of business tools that users deploy to digest and report on data. These databases work well for structured data such as OLTP and OLAP workloads. At the same time, the increase in volume and variety of data now requires database engines to run in more distributed ways.

Examples of these types of engines are Google BigQuery, Snowflake, Redshift, Druid, and Presto. Because most of these tools are built with SQL capabilities, you can plug and play your BI solution into most big data technologies that support ANSI-SQL. Having these tools connected to your data lake eliminates the huge wait time it previously took to query data warehouses, reducing the time to get access to information from hours or days to near real time or minutes.

Visualization tools such as Looker, Power BI, and Tableau will always be important. Even Excel on top of the data lake can be valuable because now you can connect your BI tools directly to your data lake—not just a traditional data warehouse—so that you can perform advanced analytics that require immediate access to petabytes of data, data discovery, or the many other use cases we see today.

Tools for Extracting Intelligence

The following are some tools that you can use to get actionable intelligence from your data lake.

Looker
: Looker is a BI tool that provides visualization capabilities and comes with real-time analysis. Users can select different types of visualizations from the Looker library or create a custom visualization, including bubble charts, word clouds, chord diagrams, spider-web charts, and heat maps. Looker offers analytics code blocks (Looker Blocks) with SQL patterns, data models, and visualizations included. Although the blocks are prebuilt, they are customizable and can be adjusted to the needs of the user. LookML employs some features of SQL, but with improved functionality.

Power BI
: Power BI is a business analytics service by Microsoft. It offers interactive visualizations with self-service BI capabilities, allow-

ing users to create reports and dashboards without having to ask IT staff or database administrators for assistance.

Tableau
: Tableau is a BI and data visualization tool with an intuitive user interface. Users don't need to know how to code, and they can drill down into data and create reports and visualizations without intervention from the data team or IT.

Getting Data Out of Your Data Lake

When you begin thinking of extracting data from your data lake to drive business insight, you generally have two options: canned reports and ad hoc queries.

Canned reports are typically executive reports that are regularly generated. Whether they're created every morning or even every hour, a pipeline runs a set of predefined data operations that usually results in a PDF or dashboard. Canned reports tend not to change very often, because they are really what runs the day-to-day business.

Ad hoc reports involve users asking general questions of the data, such as "I have a customer on the phone, what's happening with his order? When will this product arrive in inventory?" and so on. These reports are usefuly when you're trying to solve a problem at the moment, but don't necessarily need to be persistent like a daily or weekly report.

Presto for Ad Hoc Analytics

One of the key personas in a large company's data team is the data analyst. Typically, the job of a data analyst is to explore data, generate reports, and interpret how the business is doing. Usually, a lot of data comes in. It might be in real time, it might be on an hour's delay, or it might be on a delay of a day or more. It doesn't matter. The job of the analyst is to analyze the data to reveal various metrics about the business.

Sometimes, an analyst needs data about something that is happening now. For example, a shared-ride aggregator like Lyft might want to know how one of its ride categories is doing in Seattle today because it did a promo there. Or, it may want to do a little bit more

historical analysis that investigates which category of service different customer demographics like. For example, are college students going for the cheap cars, are they going for the midsized cars, or are they going for the SUVs?

This is where Presto shines compared to Spark and Hive. Recall in Chapter 3 that Presto is built for running fast, large-scale analytics workloads distributed across multiple servers. Presto is built for SQL, which is the common skill among analysts. In fact, it is an imperative language that helps analysts transition to Presto from other databases. They do not need to worry about the underlying technology or the scale of the system to handle big data analysis.

When it comes to query runtime performance, Presto will not have the millisecond response offered by some of the traditional databases out there (e.g., Oracle or Greenplum with DB2), but it is generally considered faster than Spark and Hive. Presto also shines for common analytics ETL operations, such as large joins and table aggregations. All in all, there is a good chance that Presto will work better with more business tools than the other engines. For these reasons, if you must move away from one of the more traditional mainstream databases to something that is built for the data lake, Presto is the easiest option.

Because of its support for federated queries, Presto fits well into any ecosystem, as depicted in Figure 6-4. It works well with Hive Metastore, HDFS, relational database service, and cloud storage. It also integrates with other data stores like MySQL, PostgreSQL, MongoDB, and Oracle. This helps analysts quickly bring together ad hoc data sources for impromptu analysis without the need for an ETL phase.

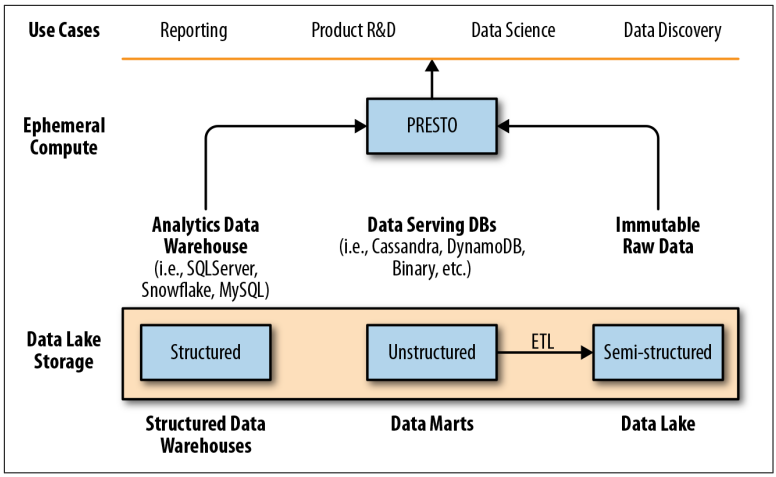

Figure 6-4. Operational dataflow using Presto in the cloud

Because you have a lot of data, speed is definitely important. You probably don't need subsecond speeds, but you definitely need subminute speeds. And because you have hundreds of terabytes of data and you want subminute speeds, there's an assumption that you would want to use a cluster of machines to do this job. And Presto can do the capacity planning to make it work well.

On top of that, Presto has also been adopted by large companies like Uber, Lyft, and Netflix. This means that it is also battle hardened, not just at Facebook, but at some of the other top technology companies in the world. There is therefore less of a burden on other companies to do quality and stress testing. The technology world is doing that for us.

Ad Hoc Versus Canned Reporting

As we've discussed, the data lake is meant to work for both structured reporting—often called prepackaged, prebuilt, or canned reporting—and ad hoc reporting, in which you're searching for data in an unexpected or unique way. Data lakes are especially good for ad hoc reporting—the spontaneous SQL or search queries to which you wouldn't know how to tune your relational database to respond.

The type of report depends on the type of user

Different users will prefer one type of report over the others. Users without particular technical expertise or who lack SQL experience generally are given the prebuilt, canned reports and given access to drill-down reports you get from popular visualization tools. Typically, the higher you go in an organization, the more users will want these simple canned reports that show only high-level views of data.

As you go down the organizational hierarchy, you will want to deliver more elaborate reports—still canned—that are consumed by mid-level management and some data analysts. These kinds of reports go into more depth, into "what if" varieties of ad hoc searching on a limited dataset.

Finally, some data analysts will want to do their own ad hoc querying in SQL for ad hoc reporting.

The challenges of dealing with ad hoc queries

There are two particular challenges when it comes to ad hoc queries: ensuring quality and balancing speed of reporting with speed of searching.

The first challenge is data quality. Organizationally, you need a formal program for evaluating data quality. After all, the reports will only be helpful if the data is accurate, up to date, and meaningful. To set up a program like this, you would need to inject quality checks into your ETL processes. You might do random checks of certain datasets or metadata.

The other challenge is balancing speed of reporting and speed of searching. Typically, the data analytics team that receives requests from users monitors the various types of requests. If a number of requests for the same type of query grows, you have an obvious need to create a standardized report based around that query. It could even become a separate dataset prebuilt for that query that is faster to access and therefore can satisfy those requests quickly.

This will reduce your ability to do additional ad hoc searching on that dataset, but it increases speed and helps you avoid having to do additional programming to deliver similar reports to users over and over again. And, ultimately, you want to shift users as far as possible from having to ask for ad hoc reports to receiving prepared reports.

The need for competence in analytics reporting

In analytics reporting, you also need what we'll call *competence*. The three main areas of competence when it comes to analytics are: *data-enablement competence*, *decision-support competence*, and *data-excellence competence*. Let's look a bit more closely at each of these:

Data-enablement competence
: This means providing platform information, integrity, and data and enterprise integration in a way that ensures the data will always be available and reliable. This is critical to a smooth analytics reporting function.

Decision-support competence
: For your team to be competent at decision support, you first must acknowledge that there are two sides to successfully applying data science. One is that the data scientist is expected to be knowledgeable about implementing various mathematical models to detect patterns in the data. The other side of it is that data scientists also need to understand the business, or they won't be of any real help to the bottom line.

Data-excellence competence
: What happens with data in the end is very much a function of how well and how tight the data-excellence processes are. These processes include what the data team—data engineering in particular—is doing to ensure that the data is of the highest quality so that business decisions can be made using it.

Structurally, this means that you can organize your data teams in two different ways. The first way is to have data scientists work closely and directly with users within your different business units or with executives in the C-suite. To do this, the data scientists must both understand the business's needs as well as use their experience and expertise to deliver insights to the business based on big data.

Alternatively, you could put the data scientists into two different groups by function. One group would primarily interact with business users to gather their needs. Then, they would take those needs to the other group, who would do the actual data science. We think the former is a much healthier way to go, and thus we recommend that businesses go that route.

Step 6: Productionize and Automate

At this stage, you have your data lake built. You are focused on making it into a production-ready resource and improving its operations through automation.

Here's how you'll know that your data lake is production ready:

- Your users have clearly defined expectations and specific goals that they've prioritized for using the big data in your data lake.
- Your data lake possesses the security and governance required of any enterprise-class infrastructure or resource.
- You can scale and add new storage, compute, and network capacity quickly that exactly match your needs at the moment, with no waste.
- Your data team has the required skills to support the data lake throughout the data life cycle.
- You can efficiently perform incident response, manage trouble tickets, provide training, and in general deliver all the support functions required of an enterprise-class operation.
- You have automated as much as possible to reduce errors and the stress on the data team.

Tools for Moving to Production and Automating

Now that you are ready to put your data lake into production and automate it, you will need tools. Happily, a number of great tools have emerged so that you don't need to build them yourself. Specifically, you can deploy workflow schedulers and ETL managed services from the open source world to help you with this final stage of building your data lake.

Open Source Workflow Schedulers

Apache Airflow is an open source tool for orchestrating complex computational workflows and data-processing pipelines. An Airflow workflow is designed as a directed acyclic graph (DAG). When using Airflow to author workflows, you divide it into tasks that can be executed independently.

Oozie is a workflow scheduler system to manage Apache Hadoop jobs. Oozie Workflow jobs are DAGs of actions. Oozie Coordinator jobs are recurrent Oozie Workflow jobs triggered by time (frequency) and data availability.

Azkaban is an open source workflow scheduler created by LinkedIn that aims to reduce Hadoop job dependencies. It is a batch job scheduler that allows developers to control job execution inside Java and especially inside Hadoop projects.

Pinball is workflow management software developed to manage big data pipelines. It is available as open source.

ETL Managed Services

ETL is a process for preparing data for analysis. It is most commonly associated with data warehouses, but it can also apply to data lakes. *Extract* involves extracting data from homogeneous or heterogeneous sources. *Transform* processes data by transforming it into the chosen format for analysis. Finally, *load* describes placing the data into the final repository—in our case, the data lake. Following are three tools that you can use to manage your ETL tasks:

Informatica
: A widely used ETL tool, Informatica's PowerCenter helps you to extract data from multiple sources, transform it to the right format for your data lake, and load it into the data lake.

Pentaho
: This is BI software that provides data integration, OLAP services, reporting, information dashboards, data mining, and ETL. Pentaho enables businesses to access, prepare, and analyze data from multiple sources.

Talend
: This is an open source data integration platform that provides various software and services for data integration, data management, enterprise application integration, data quality, and cloud storage.

Apache Airflow

As noted, Apache Airflow is an open source tool for orchestrating complex computational workflows and data-processing pipelines.

Airflow workflows are designed as DAGs. When authoring a workflow, you need to consider how it could be divided into tasks that can be executed independently.

Apache Airflow was built because traditional workflows either relied on Cron or a fixed series of serialized commands. Airbnb needed to manage the tasks associated with multiple big data operations. It wanted to provide its users with a better system for managing task workflows, and Jenkins, the existing tool for doing this, was difficult to manage.

Airflow ultimately provides a much easier way of chaining complex interdependent tasks, and also has the advantage of using Python.

As both a scheduling tool and task workflow automation system, Airflow's lightweight, Python-based service uses workflows made of DAGs for tasks. It has an intuitive user interface and is very easy to scale. All the heavy lifting is done by engines, and it can manage complex workflows such as these:

- Building data ingest and monitoring pipelines
- Deploying machine learning models
- Building data-reporting pipelines

Qubole found that more than a third of its customers use Airflow, typically to more easily build and manage data workflows and orchestrate extremely complex data pipelines. Airflow has a strong and growing open source community.

Best practices for using Airflow include the following:

- Use multiple Airflow instances for dev, test, and production.
- Run Airflow in Coordinated Universal Time.
- Control who can deploy DAGs. Isolate pipelines by team by using multiple Airflow instances.
- Ensure that heavy ETLs are done on clusters, not on Airflow's instances.
- Don't immediately assume that you need an Airflow cluster. A single instance is generally easier to maintain than a distributed cluster.
- Understand the difference between the web server, scheduler, and workers.

CHAPTER 7
Securing Your Data Lake

Security architectures in the cloud are very different from those on-premises. Today, the cloud is reasonably secure. But it has taken some time to get here.

When the public cloud began, it lacked security functionality. For example, AWS EC2-Classic instances received public IP addresses. A few years later, Amazon introduced its virtual private cloud (Amazon VPC), which included private subnetting and boundaries. Since then, the cloud has matured from typical compute in which security is a minor consideration to an environment with the extensibility and functionality that allow a security professional to more reasonably protect infrastructure.

We've seen three generations of cloud security thus far. In the first generation there was little—typically just ad hoc—security. Then, the second generation introduced virtual private clouds and third-party services to enhance security, such as application firewalls. The third generation has included logging approaches like setting up Lambdas to trigger on certain events and AWS Gatekeeper. There is room for a fourth generation to improve cloud security even more.

It's very important to look at the security primitives differently than those of on-premises setups, and to adopt those security primitives to create a secure data infrastructure.

When securing a data lake in the cloud for the first time, security engineers need to:

- Understand the different parties involved in cloud security.

- Expect a lot of noise from your security tools as well as the big data frameworks and solutions that your team needs to use.
- Carefully establish privileges and policies on your data based on what you're trying to protect and from whom.
- Use big data analytics internally to inform and improve your organization's security capabilities and decision making.

We will explore these considerations in the rest of the chapter.

Consideration 1: Understand the Three "Distinct Parties" Involved in Cloud Security

Ensuring security in a cloud-based data lake is a relatively new science; we're still figuring it all out. But when your data lake resides in the cloud, the first thing you need to realize is that you still must think about security. Yes, your cloud vendor has broad security responsibilities, which we examine later in the chapter, but you do, too. And if you use a cloud platform for big data like Qubole or another vendor with which to build your data lake, you have three organizations that must work together to ensure that your data and systems are protected.

There's you, the company that is building a data lake. There's the cloud platform owner. And then there's the cloud provider, which not only supports the platform, but also provides the tools and resources that allow the customer to store the data that is relevant, and do the processing required to analyze that data.

You can't just depend on the cloud provider to protect you. You must rigorously practice safe access control and safe security throughout your organization. And you need to question whether the cloud platform provider has sufficient security in place as well. You need to ask the tough questions. How do I know my data is secure? How can I ensure that the compute resources are safe?

Here are your responsibilities for security in the cloud:

- Customer data
- Data encryption
- User management
- Infrastructure identity and access management

- Definitions of users' roles and responsibilities (perhaps using personas)

Here are the cloud platform vendor's responsibilities for security in the cloud:

- Secure access to the data platform
- Secure transport of commands
- Third-party attestations or validations: service organization controls, HIPAA, PCI
- Operating system firewall configuration
- Metadata security

And, finally, here are the cloud provider's responsibilities for security in the cloud:

- Storage
- Availability and redundancy
- Compute
- Networking

Security Functionality That Your Cloud Platform Should Deliver

Ensure that your cloud platform provides the following security functionality:

- Strong compliance and audit logging
- Assurances that all data resides in user cloud accounts
- Strong data encryption throughout (at rest and in transit)
- Flexible and granular role-based access control
- Transparent availability and performance reporting
- Secure software development life cycle

Consideration 2: Expect a Lot of Noise from Your Security Tools

Security tools are notoriously "noisy." They generate a lot of false positives or alerts that are really about nothing.

A 2018 survey by Advanced Threat Analytics found that nearly half (44%) of respondents reported a 50% or higher false-positive rate. Of that 44%, 22% reported a 50 to 75% false-positive rate, and 22% reported that from 75 to 99% of their alerts were false positives.

Security mechanisms such as intrusion detection systems, file integrity tools, and firewalls tend to generate a lot of logs. Take the firewall, for example. It has rules programmed in to either deny or allow traffic. It creates logs for every decision. Intrusion detection systems are also rules based, and they're looking for traffic anomalies, explains Drew Daniels, VP and CISO of Qubole. They're saying, ""These are the things that I care about, and if you see any of these logins, take action."" But, he says, they still generate "volumes and volumes of data." And there are other security mechanisms that generate logs and have the potential to alert you when something unwanted or unknown occurs.

Today, 80% of security tools are rules driven. But we are beginning to see tools that are AI and machine learning aware. It's now possible for companies to take these rules-based solutions and learn from what they find. More important, they can correlate events and alerts to ease incident response.

Additionally, many security professionals dream of a "single pane" that gives them insight into their entire infrastructure instead of having to log in to multiple security tools to figure out what is going on. Having a tool that links or uses data from multiple sources would satisfy this desire.

The reason that rules-based engines are not ideal is because an incident responder must collect all the relevant log data and then put it in the correct order. Only then can security professionals determine how severe an incident was and how to best address it.

It can take up to an hour to analyze whether something is really an incident. And if you're getting 20 to 30 alerts per day, you're going to have two or three people doing nothing but chasing incidents.

Because most of those reports end up being nothing, that's quite a lot of noise, and a real problem.

Consideration 3: Protect Critical Data

Security professionals need to first consider what they are trying to protect, and from what. They don't think in terms of preventing users from accessing data; instead, they view security in terms of what's sensitive, and how they can best keep the company's data safe and secure. After all, data is increasingly a business's most valuable asset.

A best practice in the cloud (as when on-premises) is to apply the principle of *least privilege*: you don't want to deny access to data if someone needs it to do their job, but you do want to restrict that person's access to *only* the data they need to carry out that job.

How best to allow access? That depends on the organization. For example, Qubole has a policy that if you're in the office, you must use multifactor authentication (MFA) and you must change your password every 90 days. When you've used MFA to log in, it's valid for 24 hours. Then you need to sign in again.

However, when you're out of the office (remote), your MFA login will expire after 8 hours. Some companies are even more restrictive and don't allow personnel to access resources remotely without the use of a virtual private network (VPN) connection to ensure that the data transmitted and received is properly encrypted.

"Some of our most successful customers set up roles and groups. Some also use SAML [Security Assertion Markup Language] for extra protection," says Daniels.

A good first step is to create roles, decide what access rights belong to each role, and then assign users to the correct roles. Sometimes, you can have an entire group of users with the same access rights; at other times, you need to allow access to certain data or systems or tools on a case-by-case basis. But using the principle of least privilege is critical throughout.

Because certain users need to work with multiple systems and use cases, there can be difficulties. You want to make sure that you understand what access each person needs to be able to do their job. Then you want to provide that access, but not overprovide it. Many

companies dictate access rights by role—for example, data scientists have certain rights, and data analysts have others. Or you can assign rights by functional team. You can tailor the specifics to your context, as long as you're operating on the basis of the principle of least privilege.

"For example, you don't necessarily want employees from marketing or sales accessing corporate financial data," says Daniels. "So, you can limit their access based on their function." Following from this, you might want to go quite granular on what you allow users to do within a system or with data. You could allow your data analysts to use existing clusters, but not to configure or create new ones. In such a case, says Daniels, "data scientists may want to start certain clusters and tune them to do something different."

Many of today's cloud-native tools allow for remarkably granular controls. Fundamentally, it is about protecting the business's data. After you get past that, you should be making sure that the right configurations exist for the right users. Tools like Apache Ranger (discussed in Chapter 6) make this possible.

In a typical business environment, the data team will sit down with whoever needs data—perhaps the finance team—and they'll capture requirements from that team. What do they need to see? What do they need to do? And then the data team creates a role that grants certain rights to financial employees. They might be able to look at the last two quarters of financial results, but they can't go in and do a custom query or ask questions that haven't already been structured in a report or a notebook. Then, if users need more access than that, they can come to the data team, open a support ticket, and make their requests, which can be decided on a case-by-case basis.

Consideration 4: Use Big Data to Enhance Security

The fourth and final issue is this: instead of simply serving the needs of the other functions in the organizations, why not take advantage of all of these tools—big data, advanced analytics, AI, machine learning—to improve security operations?

"Much like other areas in technology, security generates a lot of data, and one of my passions is trying to figure out how I can take a look at that data, combine it with other datasets, and be able to find

things that I might not otherwise be able to discover," says Daniels. "A lot of these tools generate terabytes of log data every day, maybe even more depending on how large the infrastructure is and how widely the tool is deployed."

Companies need to find ways to leverage all this security data, and use data science and machine learning to learn how to make their organizations more secure. This is the dream of incident response leaders, who, as mentioned earlier, currently spend far too much time per incident, which is both inefficient and costly.

Very few people, if any, are doing this today. Managed services security operations centers (MSSOCs) are trying, however. With an MSSOC, you have a third party adjusting all of the logs and doing the analysis for you. Often this raises concerns. Does this MSSOC support your tools and data? How do you send them your data? What are they doing with the data? Who owns and controls the data?

From a security perspective, this is something that many CIOs or CISOs worry about. "I don't like when somebody is saying, 'Give me all your data. Oh, here's your incidents,'" says Daniels. What are they potentially missing? Because it's completely outsourced, you don't know what that is, and you don't know how bad it might be.

"I think the second thing is security professionals are often paranoid," says Daniels. "They want to know how the lasagna's being made."

Right now, the most common way that companies manage security is by sifting through all the data that their security tools generate. But the more tools they use, the more difficult that becomes.

"We at Qubole have started taking the approach of mandating that our security vendors dump all that data in a common format and location so that we can take it and figure out what issues might need our attention," says Daniels.

When you get to this advanced stage of security, you're not just focusing on data governance, but are also proactively seeking out use cases around fraud detection, or hacking attacks, or easier user administration and management.

"Typically, security teams are understaffed and underfunded," says Daniels. "So I get a lot of people coming to me asking, 'How did you

do it?' They're extremely thrilled about the possibility of doing this themselves."

Qubole is close, he says. It has been working with its vendors to get security data outboarded to Amazon S3 so that it can ingest it, "and we now have three or four of those data sources ready to be ingested," he says. "I'm excited at the prospect."

CHAPTER 8
Considerations for the Data Engineer

The data engineer is the person on the data team responsible for gathering and collecting the data, storing it, performing batch or real-time processing on it, and serving it via an API to data scientists so that the data scientists can easily delve into it to create their experiments and models. Data engineers are often the first team called on when things break, and the last to get credit when things go well, yet they are arguably the most critical component to operationalizing the data lake.

Data engineers are primarily responsible for managing the volume, variety, velocity, and veracity of the data in the data lake. By properly managing these "four Vs," data scientists and data analysts can more easily find value in the data—the fifth V.

With organizations wanting to embrace the data-driven model, data engineers are constantly under pressure to learn more technologies and apply them to their development processes. Today they must understand distributed computing, good data warehousing technologies, and the difference between transactional modeling and analytic modeling, such as in OLAP versus OLTP systems.

To add to these many hats, data engineers also must be quality engineers. They need to monitor the quality of their data, the shape of their data, and their metadata statistics so that they can quickly spot a problem and correct it.

The upshot: today, data engineers need to master almost an entire development methodology to do their jobs well.

Top Considerations for Data Engineers Using a Data Lake in the Cloud

Most organizations decide to build a data lake because their business and data needs are growing beyond reporting off their transactional databases—the ones that run their businesses—and that simply isn't working for them anymore. Although it's fairly easy to run canned reports off such systems, ad hoc reporting is trickier. As your data expands and your users become more sophisticated, delving into complex analytics and machine learning makes clear that the next evolution of the data architecture is to build a data lake.

Protect Your Users

Indeed, the first sign that you need to build a data lake is that your data consumers begin complaining. This can manifest in different ways. Perhaps analytics queries are causing issues with the transactional databases, or data scientists want to begin building models, but they can't get the data out of the transactional stores.

What organizations typically do at this point is take dumps of their data out of their transactional databases and put them into the raw storage of a data lake. This is where data engineers become critical.

Data engineers often write jobs to pull tables out of transactional databases and break them down into JSON, which is then stored in the raw partitions of the data lake. From there, data scientists could use Spark, or analysts can use Presto and Hive, to analyze the data.

Although this works for a while, using a text format is a magnitude slower than using an optimized format. As your data consumers request that their data be served faster and faster, data engineers are tasked with building data pipelines that convert data from raw to optimized. The two dominant formats data engineers use currently are ORC and Parquet. (Although others are emerging, they have yet to gain widespread adoption.)

However, depending on how your tables are structured, your data consumers can become confused if you're constantly changing for-

mats. Data engineers thus need to somehow isolate their users from these optimizations and schema evolutions.

Ensure That Data Governance Is in Place

Now, let's talk about data veracity in a cloud-based data lake. After all, data quality assurance is one of the chief duties of a data engineer. When working with a data lake in the cloud, data engineers need to understand the shape of the data in their pipelines. They need to understand, based on columnar data, what the minimum, maximum, and averages of the data are. This will help them catch outliers in the data, or whether something has gone awry during data conversion. There are a lot of tools for this functionality.

Many DataOps teams will run statistical analyses on the data and emit the results to a monitoring tool, such as Datadog or LogicMonitor, that alerts the team when an error or outlier is detected. There are several great open source data profilers available.

Data engineers are also responsible for managing the data life cycle as part of data governance. This means that they work with business analysts, users, and data scientists to create a data life cycle strategy, determine each dataset's time to live, and find the right storage media for it (hot, warm, or cold) for as long as it needs to be stored. Public cloud platforms offer a range of data storage options to meet businesses' storage needs.

Typically, as data ages, it becomes less and less valuable, although there are regulations that require keeping certain types of data for a specified number of years. For example, the HIPAA health care data mandate requires that affected businesses retain patient data for at least six years from creation date or last effective date, whichever happens to be later. Financial services firms need to keep data for a minimum of seven years in case of an audit. Data engineers need to stay on top of these and other regulations.

Although most businesses hate deleting data, most are able to articulate when their data isn't useful, whether that's after 30 days, 90 days, or even a year, and do away with it. All that is part of the business's data governance policy, as overseen by the data engineer.

Although security is everyone's responsibility, data engineers are part of the security process and should help to find a solution that

minimizes risk to the organization yet doesn't critically impede the needs of the data-driven organization.

Designate Areas for Raw and Optimal Data Storage

In the raw storage area, the data should remain as is, without any transformations. This area is critical to the overall data lake architecture because it provides the ability to rebuild downstream tables if needed due to some kind of catastrophic failure. It also enables forensics to determine whether and when data changed or an error occurred. The optimized data area is where you want to put your data that has been cleaned, scrubbed, transformed, and democratized for the end consumers. You need to create both areas in your data lake if you want to successfully mine the data for value while keeping lineage in order and quality assured.

Considerations for Data Engineers in the Cloud

Data engineering is also responsible for properly democratizing data in the organization. This is best done in the cloud.

Democratize the data
: Data democratization can be done several ways. Many organizations will probably have a data dictionary for data exploration, whereas more advanced organizations will create automated metadata management and data lineage tools, which provide more context around the data.

Promote ease of use
: It's imperative that the data lake be easy to use for downstream consumers. Data engineers are responsible for understanding the needs of the consumers and designing the data lake in a way that maximizes a downstream consumer's productivity. An example of this is implementing views on tables so that downstream users are insulated from upstream schema changes.

Enable automation
: A good automation tool is worth its weight in gold for a data engineer. It's important that the tool be easy to use, scalable, extensible, and flexible. It's also important that the tool have a notification system in the event of a missed SLA or pipeline failure. Our personal favorite is Apache Airflow.

Summary

In short, the goal of advanced data engineering is to democratize data in the organization so that nontechnical users can consume and use data efficiently without having to delve into the technical parts. You could say that as data engineering becomes more advanced, the focus becomes less technical and more user oriented. This is counterintuitive and seems backward, but it's true.

The data engineers' basic responsibility is to enable the rest of the organization to consume data in an easy, efficient, and confident manner. This could involve taking data scientists' machine learning models and developing a smartphone app that allows marketing professionals to perform ad hoc querying of real-time sales data. Or, enabling data analysts to develop an interactive dashboard to identify areas of business opportunities or inefficiencies. Or, perhaps creating an application that allows product managers to do A/B testing against historical data to determine the best functionality to put in a new product design. A data engineer could build a portal through which external customers could view an account, run scenarios, and identify key data points in a safe and secure manner.

CHAPTER 9
Considerations for the Data Scientist

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from data in structured, semi-structured, and unstructured forms.

Data scientists live at the intersection of science and statistics. They care about the fifth "V" of the "five Vs" of data science: *value*. The results of their labor are usually used to drive business decisions and perform predictive problem solving. For example, a data scientist might analyze a broad range of customer data from multiple sources —structured and unstructured—to attempt to predict when a customer is at risk of churning.

Pradeep Reddy, a solutions architect at Qubole, says, "If I'm a telecom provider, if I can actually predict a customer who will churn three months from now, I could take some corrective actions." He continues: "I could then send him a flier or offer a promotion in an attempt to retain him or her."

It's important in such cases for data scientists to identify the "white spaces" in their data. For example, if you depend on geolocation data to power an application that tracks consumer behavior, you are at the mercy of whether a consumer has opted out of geolocation. Suppose you are collecting this data from customers' smartphones—you're fine until they power down. Then, you don't know where they went. What they bought. Where they ate lunch. There's a hole in

your data. That's when you turn to third-party data providers, to find the right data and fill those holes. Otherwise, you make the wrong assumptions—and poor decisions result.

Data scientists are frequently skilled at a lot of things. They can often put on the hats of data engineers and perform their duties. They are also experts at statistics and can design a statistical model that extracts value out of data.

Of course, there is no perfect data scientist. Some are very skilled at statistics, others are proficient at engineering, and still others excel at execution. It takes a collaborative approach to extract maximum value from the data. "In the Qubole customer base, the ones who have been most successful at executing data science are the ones who have embraced a collaborative culture. It is a team effort," says Reddy.

In fact, your data team should include data scientists, data engineers, web developers, and product people as well. "You've got to leverage all your functions to inform your data science program," says Reddy.

Data Scientists Versus Machine Learning Engineers: What's the Difference?

It's a truism that technology is easy; it's the humans that are difficult. It's also true—as the notion of DataOps brings home—that data teams do their best work while collaborating. Having said that, there are certain cultural issues around the two main personas, data scientists and machine learning engineers, that you need to know about when attempting to operationalize machine learning in the data lake.

First, be aware that data science is an evolution of the analytics function. Companies have historically tasked data scientists with building advanced models and statistical models to understand data, but they also have been tasked with building products out of data that were used in production. Data science traditionally, then took advantage of the skills of analysts to gain advanced insights from data but also of data engineers to build products that gave the business a competitive advantage.

In the mid-2000s, big data technologies emerged, which led businesses to define a new data role: the machine learning engineer.

As data scientists increasingly focused on experimentation, businesses needed a new role that was specifically tasked with building models in production. In some cases, data engineers can do that, but often they don't have the statistical knowledge to understand what's going on with the models. Thus, today many businesses hire machine learning engineers, which are software engineers who specialize in deploying machine learning applications that use Hive, Hadoop, and Spark technologies.

Today, the data scientist is primarily tasked with doing experiments with data. Because of the way models work, it's a trial-and-error process. They try one model, they try one feature, they try one algorithm, until they hopefully arrive at an optimal combination. Data scientists are experts at that process. They also are expert at storytelling. They require strong communication skills to work with product managers and business users to formulate hypotheses and test them using data and the models. Often, you can recognize data scientists by the tools they use: R and R Studio or Python and Jupyter Notebooks.

After creating models, today data scientists typically hand off the code—or, as a best practice, collaborate with the machine learning engineers—to *productionize* the model.

What does productionizing a model mean? The model the scientist data created is not optimized for production. That is, it might not scale as the business requires, it might not have logging or fault-tolerance capabilities, and it might not be modularized so that it can be embedded into an application.

Machine learning engineers come in and restructure the code. Sometimes, they rewrite it into a different, more complex language that performs better, such as Scala, Java, or C++. The tools data engineers prefer to use for doing this are typically integrated development environments (IDEs) such as IntelliJ IDEA or Eclipse, which are very different from the notebooks that data scientists use. An IDE's output generally is a library or module that is deployed into an application.

In businesses that are less advanced along their big data and machine learning journeys, data scientists tend to wear many hats. Indeed, even data scientists in large organizations, according to interviews and expert estimates, spend from 50% to 80% of their time on the tedious aspects of collecting and preparing data—what

data scientists call *data wrangling*—before they can mine it for useful insight. Although tedious, collecting data and extracting features determines the final quality of the model built. "Garbage in, garbage out" is a common adage to describe a model constructed with bad data.

In fact, most organizations are still struggling with how to organize their data teams—specifically, with dividing the tasks between data engineers and data scientists. "Should there be two distinct functional areas? Or should they be virtual, shifting teams, in which there is a sharing of best practices among them? Should you teach data scientists how to do basic data engineering and ad hoc analytics? Or is that something they shouldn't waste their time on?" asks Mohit Bhatnagar, senior vice president of products at Qubole. "This is not fluff. These are very real problems that people are grappling with," he says.

Data Scientist Use Cases

Use cases for data scientists can be categorized as either batch or real time. A batch job means that you don't have a person waiting for the model to be scored, and it doesn't have strict SLAs. There's nobody waiting on a website or a mobile app who needs to be served a prediction in real time. This means that you're running a scheduled job periodically: once per hour, per day, or per week. In many such cases, the data scientist can do it themselves if they know Spark or some other production-level engine.

In a batch scenario, data scientists will create data transformations using a database or perhaps Python, but if the dataset is not that large, they don't really need Spark, so they can do it all themselves easily.

If, however, your dataset is large, Spark is optimal. Writing a model in Python could end up causing bottlenecks because Python doesn't scale in production. A machine learning engineer is going to need to be involved at that point.

Alternatively, if data scientists know how to write in PySpark or even Scala, you're much better off. They can write a SQL CREATE model. They can do more advanced data processing and schedule it via notebook or a job and using a cloud platform like Qubole or Databricks or EMR to get it to production.

Under a real-time scenario, you're focused on delivering a model that serves a prediction to a person—a customer or employee—who is waiting for it, in real time. Suppose that you're Amazon. You have strict requirements for your model to serve search predictions very fast; otherwise, the customer is not going to return to the website. The pipeline for your model has an entire set of requirements that are very different from a batch job that doesn't have a person waiting on the other side. In that case, most likely you will need an engineer to specialize in squeezing any sort of inefficiencies in the code because the data scientist doesn't specialize in that.

In an ideal world, you would want a platform administrator or a platform engineer who understands all of these intricacies so that you can plan ahead for all the possible use cases. Depending on whether you want to do batch deployment or real-time machine learning applications, the infrastructure is going to be very different, as are the skillsets required. If you have planned ahead of time, you can choose the appropriate platform and also recruit people with the necessary skillsets from the market.

How a Data Scientist Begins a Project

It all starts with business understanding. "There's a lot of hype associated with data science, but unless you actually tie business value to your data science programs, they won't be successful," says Reddy.

And it's a truism, but it's valid, that you need to begin with the problem, not the data, says Ashish Dubey, vice president of solutions architecture at Qubole. This means following a four-step process:

1. Identify a problem that needs solving.
2. Hypothesize what a possible solution would look like.
3. Evaluate your data resources.
4. Create a model and iterate, iterate, iterate.

This means that a data scientist should first think about deep business problems, and also hypothesize about what a feasible solution might be, before going to the data.

"The reason to emphasize what kind of solution you are looking for is that there's always a chance that you could overengineer the problem, whereas the answer might be quite simple," says Dubey. "Some-

times you might find your solution with a much simpler traditional approach such as a series of SQL workflows."

The next step, which again is very important, is to understand the datasets to which you have access. You might have a problem and a viable hypothesis for a solution, but if you don't have the data to test your hypothesis, you're going nowhere.

Then, it's time to build your model. And you won't be building just one. You'll be constantly tweaking, changing, and iterating to keep testing until you get the results you are looking for.

Top Considerations for Data Scientists Using a Data Lake in the Cloud

Here are some best practices for data scientists, according to Dubey.

Focus on data quality before production
: As we mentioned earlier, it's very important to understand the problem, the hypothetical solution, the feasibility of that solution, and your access to quality data, and then to assess how much time a particular data project might take and how much it would cost.

Divide larger training workloads into smaller ones that can scale
: This is especially important with distributed systems, for which you are processing terabytes or petabytes of data and need to apply similar algorithms in a much more distributed way. To make your model production quality and scalable, you always need to divide your solution using smaller building blocks. Your model might be working flawlessly with 10 terabytes, but can it scale to a petabyte without encountering any bottlenecks? You can ensure this by having a robust pipeline where you've conquered one issue at a time.

Embrace the data lake
: A central repository improves data accuracy and model auditability. The predictions from a machine learning model are only as accurate and representative as the data used to train them. Every modern manifestation of artificial intelligence and machine learning is made possible by the availability of high-quality data. For instance, apps that provide turn-by-turn directions have been around for decades, but they're much more

accurate today thanks to the larger volume and variety of data that can be housed in cloud object stores.

Enable faster access to data science tools
: All of the aforementioned considerations contribute to delayed time-to-value with laptop-based data science. In a typical workflow for a data scientist working on a laptop or on-premises server, the first step is to sample the data and download datasets manually onto the local system or to connect via Open Database Connectivity (ODBC) driver to a database. The second step is to install all of the required software tools and packages, such as RStudio, Jupyter Notebooks, Anaconda distributions and machine learning libraries, and language versions such as R, Python, and Java. In a data lake, you just need to access your tools and you're ready to query.

CHAPTER 10
Considerations for the Data Analyst

The mission of the data analyst within the data team is to apply analytics techniques to solve relevant business problems and to gain business insights that the company can use to make decisions.

Data analysts share this mission with the data scientists, but analysts are closer to the business. In fact, typically data analysts are located within the lines of business. You could even call them line-of-business users. They report up to an intelligence director for a specific line of business as well as to a general manager. This means that they have a very strong operational understanding of the business.

They also tend to know the bottlenecks or choke points of the business, by virtue of where they sit and what they do on a daily basis.

They are not statisticians, and they are not data scientists. But what they bring is a business-oriented analytical mindset. They can do early hypothesis testing, for instance. They can approach a data scientist and say, "Hey, here is a sample of data that is representative of my business. If I were to take this data sample and run some experiments, I could make a difference in how my department operates." Their job is primarily testing, identifying opportunities, and then passing that information on to people who can take action on that insight.

Data analysts always need a clear business goal toward which to work. Rather than doing "exploratory" work on data, it is essential

that they have a solid business case that they're trying to solve. That way, the project is much less risky—not just in terms of working with executives, but also when working at lower organizational levels. If you have a clear goal, you can identify a set of steps to help you get to that goal.

As a data analyst, it helps to catalog problems and do a value analysis at the outset to determine what you would gain by solving them. If you were in the operations division of your company, you would pick an operations problem that is important for the company. If you worked in finance, you'd pick a problem that would help streamline financial processes, for example. Value to the business should be clear from the get-go, if the problem is clearly defined. Risks are better managed at the ground level than at the executive level. And you want to reduce your potential losses by being ready to change directions if you run into a roadblock.

A Typical Experience for a Data Analyst

An analyst for a home food delivery service is trying to write up a new query to understand why order cancellations increased in New York within the past two days. The analyst knew this because she regularly checks a key performance indicator (KPI) dashboard that the data science team has created for her. She refers to it hourly, because it is an important metric for the success of the business.

An analyst from Chicago had had a similar issue two months ago and had run an ad hoc analysis to understand spikes in cancellations. He'd found it was caused by the Chicago office announcing a later ETA of meal delivery of up to 25 minutes during peak dinner hours. The New York analyst knows nothing of the Chicago analyst. But she vaguely remembers that something similar had happened in the past. She goes through her emails and realizes that the central team had sent out an update.

The analyst writes to the product manager of the central team but does not get an immediate response. She decides to go look for what was done in the past to deep-dive into the issue. She doesn't have a clear strategy in mind at this point. She begins searching for queries. She gets some matches on a few of the aforementioned Chicago searches from two months ago, so she opens these to see if they could be helpful.

The searches seem to be an explorative approach to audit marketplace configuration changes by time to determine if KPIs saw a steep change. The analyst realizes she can do a similar examination New York. She writes her query based on the Chicago analyst's work and receives a near-immediate response that what happened in Chicago had happened in New York because a large convention was in town with a lot of attendees who had ordered restaurant meals from their hotels using the delivery service. Problem solved.

Top Considerations for Data Analysts Using a Data Lake in the Cloud

Here are five best practices for data analysts working on a data lake in the cloud:

Make absolutely sure that you're working on the right problem and that it's an important problem
: There are many ways to do this. For example, you can collect input from the rest of the company, catalog the different problems you hear about as well as the possible solutions available, and start to substantiate why this specific project should warrant additional investment. You need in effect to do two things: estimate the value and calculate the risks of trying to achieve that value. It's that old business proposition: risk versus reward. Quantifying or qualifying that early on is important.

Establish and meet milestones
: Whatever project methodology you're using and whatever time frame you've set, establish milestones to ensure that you're on your way to finding value in the data. Demonstrate the value to your organization if you're successful.

Understand the business problem and translate it to the functional data requirements
: For instance, if you need to solve a specific problem, where do you go to find this data? How do you make that data available? What kind of tooling do you need? If you're servicing hundreds or thousands of reports, you also need to consider the costs of adding new workloads to support the reports and users that receive them.

Map functional requirements to workloads
: Identify the workloads that need to run and the right tools to run those workloads. What kind of processing do you want to do against that tooling? What is the optimum server to process your data? What are the storage requirements? What are the availability and security requirements of the data?

Work across teams
: More often than not, there could always be more information that isn't immediately available for analysis in the data lake. Staying involved with product, data science, and the platform engineering team allows you to know what other information could be available or enhanced with data mining.

CHAPTER 11

Case Study: Ibotta Builds a Cost-Efficient, Self-Service Data Lake

Ibotta is a mobile technology company, founded in 2011, that is transforming the traditional rebates industry. They provide in-app cashback rewards on receipts and online purchases for groceries, electronics, clothing, gifts, supplies, restaurant dining, and more for anyone with a smartphone.

Today, Ibotta is one of the most used shopping apps in the United States, driving more than $7 billion in purchases per year to companies like Target, Costco, and Walmart. Ibotta has more than 27 million total downloads and has paid out more than $500 million to users since its founding in 2012. Maintaining a competitive edge in the ecommerce and retail industry is extremely difficult because it requires creating engaging and unique shopping experiences for consumers.

Prior to moving to a big data platform with Qubole, Ibotta's data and analytics infrastructure was based on a cloud data warehouse that was static and rigid. This worked as long as the datasets were well structured and in tabular format. However, as the business grew, new and more complex data formats were being developed and ingested.

At the same time, Ibotta was heavily investing in new data analytics teams such as data engineering, decision science, and machine

learning. The teams needed access to the same data, but each team needed a different way of interacting with the data. Data engineering needed a set of tools that allowed it to perform ETL in many differents ways, such as using MapReduce, Hive, Spark, and Presto. The machine learning team wanted to use Spark for feature engineering and to train and deploy its models. Decision science wanted to use SQL, R, and Python to extract insights and business recommendations from the data.

Ibotta needed to grow beyond the descriptive analytics, which was complementary to its products, into a pure data-driven company. This meant that the organization needed to be segmented so that it could adequately staff the appropriate teams in order to help accomplish the following goals:

For the data engineering team
: Design the data lake, manage technologies, provide data services, and create automated pipelines that feed into various data marts

For the machine learning team
: Create new product features and move to predictive and prescriptive analytics with use cases ranging from personalization to optimization

For the decision science team
: Develop and deliver a self-service insights platform for internal stakeholders and external client partners.

To address the various goals of its data teams, Ibotta built a cost-efficient, self-service data lake using a cloud-native platform.

Ibotta needed a way for every user to have self-service access to the data and to be able to use the right tools for their use cases with big data engines like Spark, Hive, and Presto. At the same time, the data engineering team needed to be able to prepare data for easy consumption. Qubole provided an answer to the demands of both teams, those perfecting operations as well as those analyzing the data. Ibotta realized the first step to building a self-service platform was to define what data was critical to enable the analytics teams to meet critical business milestones. At the time, users were employing a combination of data (from the transactional system and the data warehouse) to run their models.

After the value of each dataset was defined, the data engineering team could begin building pipelines that extracted data from the data warehouse and Aurora and converted it to JSON format, which was then stored in the raw storage area.

From there, additional pipelines converted the JSON format into ORC and Parquet columnar format and stored the resulting data in the optimized storage area. Thanks to Airflow and its ability to monitor new partitions in the metastore, downstream pipelines could then start running as soon as the new data locations were exposed to the Hive metastore.

To mitigate the legacy data warehouse constraints, Ibotta now has ETL jobs loading data from Hive into Snowflake for consumption by its BI tool, Looker. Ibotta utilizes Hive and Spark jobs for processing raw data into production-ready tables used by the decision science team. This is all orchestrated using Airflow's hooks into Qubole to ease automating jobs via the API. Airflow gives more control over orchestration than cron and AWS Data Pipeline. It also provides performance benefits, including parallelization and the flexibility of scheduling jobs as a DAG instead of assuming linear dependency.

Ibotta uses Qubole to provision and automate its big data clusters. Specifically, it uses Spark for machine learning and other complicated data processing tasks; Hive and Spark for ETL processes; and Presto for ad hoc queries like exploratory analytics.

Utilizing this platform, Ibotta has empowered the decision science team to use BI tools to produce real-time dashboards for hundreds of users. Since instituting their new data platform, Ibotta has increased the volume of processed data by more than three times within four months, and it is passing more than 30,000 queries per week through Qubole.

Ibotta's decision science team was immediately empowered after Qubole was in place. It achieved the goal of self-service access to the data and efficient scale of compute resources in Amazon EC2 for big data workloads. Within a month, the machine learning team was launching new prescriptive analytics features in the product that included a recommendation engine, A/B testing framework, and an item-text classification process.

By using Qubole on AWS, teams at Ibotta are able to provision resources themselves without having to engage a central administration group. Big data clusters are using a 60% to 90% mix of spot instances with on-demand nodes, which, combined with the use of Qubole's heterogeneous cluster capability, makes it really easy and reliable to achieve the lowest running cost for big data workloads. Additionally, autoscaling and cluster life-cycle management provide significant savings to Ibotta's cloud infrastructure costs. This means managing budget and ROI is much easier, and Ibotta can forecast how to scale different features and projects accordingly.

Ibotta is focusing on delivering next-generation ecommerce features and products that help drive both a better user experience and partner monetization. Qubole allows Ibotta to spend time developing and productionizing scalable data products. More important, Ibotta can concentrate on bringing value back to users and customers. Specifically, a cloud-native data platform has allowed Ibotta to achieve success in the following areas:

Data-driven culture
: Continuing to ensure that technology, analytics, and company culture work together seamlessly

Product innovations
: Using Qubole to drive even greater actionable insights for Ibotta's client partners

Performance
: Query tuning, code reviews, and optimizing cluster and data structure to improve operational efficiencies and performance across queries and workloads

CHAPTER 12
Conclusion

Although there are numerous ways to build a data lake, we believe that adopting a cloud-native platform capable of handling complex, varying workloads as well as delivering deep analytics and machine learning for your big data is the way to go. Even if you are not using all of the technologies mentioned, we hope that you were able to see why we argue this is the case.

In this book, we've provided a brief history of big data tools as they evolved, first in the open source world and then later as commercial or Software-as-a-Service distributions, and the subsequent development of the public cloud market. We've discussed how the trends have converged to give today's enterprises powerful choices on how to extract value from their structured and unstructured data.

We also showed you why you need a data lake to most effectively take advantage of your data. We made the case for a data-driven culture and showed you how to get there. Then, we walked you through how to build a data lake and stressed the benefits of building your data lake in the cloud. After you're in the cloud, you'll need tools to manage your growing data lake, so we provided a roundup of those. Security is just as important in the cloud as it is for on-premises setups, and we explained how to do it right. Then, we went deeper into the roles and responsibilities of three key members of the data team—data scientists, data engineers, and data analysts—to help you establish your own team with a structure that works, given your size, budget, and culture.

Today's leading platforms provide easy-to-use tools such as SQL query tools, notebooks, and dashboards that use powerful open source engines. Optimally, such platforms also provide a single, shared infrastructure that enables your users to more efficiently conduct data preparation, analytics, and machine learning workloads across best-of-breed open source engines.

This approach gives your data team the five key capabilities they need to operationalize the data lake and make it possible for your business to extract value from it:

Scalability
: With the cloud, the sky's the limit. You can analyze, process, and store unlimited amounts of data. By scaling up resources when you need them, you can minimize costs by paying only for the compute and storage you need. This helps you with financial governance to ensure that you stay within assigned budgets.

Elasticity
: What goes up can come down again. The cloud allows you to scale down as easily as you scale up. You can even change the capacity and power of machines on the fly, making your business much more agile and flexible when dealing with today's often-volatile markets.

Self-service and collaboration
: Because everything in the cloud is driven by APIs, your data consumers—your data scientists and data analysts—can choose the resources they need without requiring that someone else provision these for them. This eliminates the bottleneck of waiting for someone to set up appropriate infrastructure for your models or queries.

Cost efficiencies
: With the cloud, you reap cost efficiencies on two levels. First, you save because, unlike with an enterprise software license, you pay only for what you use. Second, your operational costs are much lower because the cloud boosts the productivity of your IT personnel. They don't spend time managing hardware or software for infrastructure, which includes never again having to perform an upgrade. All of that is done by the cloud vendor.

Monitoring and usage tracking
: Finally, the cloud provides monitoring tools that allow organizations to tie usage costs to business outcomes and therefore gain visibility into their ROI. Having the tight financial governance that this enables is a huge deal.

Best Practices for Operationalizing the Data Lake

To wrap up, here are some tips on how to choose the right cloud-native big data platform:

Don't get locked into a single open source engine
: Some cloud platforms allow you to use only certain engines. Don't go there. As we explained in this book, your data team is made up of a diverse mix of operators and consumers, and you're going to have multiple personas with multiple needs working on your big data initiatives. Each persona on your data team probably has their own preferences with respect to big data tools.

  Also, data science is evolving very quickly. A good best practice is to be open to using all the engines that are out there. What was cool yesterday is okay today, and will be passé tomorrow. Even if a tool is hot today, you know something new is around the corner. Giving your data team choices from a broad range of tools is a vital best practice. This means choosing a cloud platform that supports all the engines available, to keep your options open as your team works together to productionize advanced analytics models for both batch and streaming jobs.

Make sure the platform can do autoscaling to help you with financial governance
: Cloud-native platforms like Qubole intelligently optimize resources through autoscaling. They automatically assign more capacity when needed and release resources when workloads require less capacity by doing intelligent workload-aware autoscaling. This represents a huge game changer for organizations that pay only for what they use rather than preemptively ordering capacity and hiring teams to provision and maintain that technology.

Require self-service capabilities for all the different personas on your data team

Self-service infrastructure is critical for several reasons. First, it is efficient economically. Second, it eliminates bottlenecks that could happen at the data-engineer or data-scientist levels. The ability of these data professionals to do their jobs should not be dependent on somebody else provisioning the suitable kind of Spark cluster for them.

However, this self-service capability should not equate to any of the personas having to do all the work. For example, data scientists need access to data, infrastructure, and the appropriate kinds of libraries—but these resources should be made easily available to them. Likewise, data engineers should be able to get the resources they need very quickly without going through an intermediary.

Demand a cloud-native architecture for everything that will live in the cloud

The cloud provides a set of capabilities that the data scientists should be able to use. It's important to understand, however, that having a cloud-native architecture doesn't mean that you cannot do things on-premises. It means that when you move to cloud, you do not carry the baggage of on-premises architectures and tools with you.

Ensure that it is designed to scale

Big data analytics and machine learning models tend to expand very quickly. The amount of data that you are trying to digest can suddenly explode. You need to be able to build a scalable architecture. You need a cloud-native platform that can deal with very large-scale nodes, lots of different types of users, a range of use cases, and, as we've said before, big data engines. Keep in mind that big data initiatives rarely shrink when operationalized. Most commonly, after users get a taste of what can be done in a data-driven world, they want more, and they will put pressure on your data team until they get it.

Verify its automation capabilities

It's critical that you automate, automate, automate. Many of the functions of these big data analytical models are extraordinarily complex. It's simply not possible for anybody to do all the required maintenance manually. Whether you need to scale a

cluster up or down, or access the newest version of an open source engine, you must automate tasks as much as possible.

General Best Practices

There are also three platform-independent best practices that you should consider when planning your data strategy in the cloud:

Give your data team the freedom to fail
: Analytics is all about *learning*. Even on the human side. That's why modeling and querying both require lots of iterations. You want your data scientists to think out of the box, to try wild new things—and to continually fail. Then, they try again. You want your data analysts to be able to hone their queries and iterate on them until they get the answers they need. Don't expect immediate success, and don't criticize ideas that don't pan out. The next one might be the winner.

Break down data silos
: This is essential—and not just a philosophical argument. This is grounded in the reality that to work, your big data initiatives require clean, unified data pipelines. If you've ever worked for a fairly large multinational company, you know that each business channel typically operates in its own silo. In a financial institution, you could have a mortgage capital system, an equity system, and a credit card approval system, each having its own data and its own architecture. It could literally take one or two years before you could create models or run queries across these silos. This, of course, is why creating a data lake for one "source of the truth" is essential.

Create an end-to-end data pipeline
: Finally, when you think of operationalizing the data lake, you also need to think about end-to-end pipelines. Where is the data coming from? How is it coming? How is streaming data handled? What kind of data transformations are being done? How are you masking certain information? How is metadata being extracted? How is data labeled? After the data scientists have built these models, how is the data reused by other people? These are all essential considerations.

Now that you've laid the foundation, it's time to find the right cloud-native platform and get busy applying these precepts to use data to solve your business's toughest challenges!

About the Authors

Holden Ackerman is part of the big data community. He worked at Qubole for over three years, helping dozens of companies build successful big data platforms in AWS cloud. He stays very active in open source, and is currently a member in the Presto community, among others.

Jon King is a solutions architect at Qubole. For the past 10 years, he has been working in Big Data and building scalable solutions to derive information and value from data. His biggest accomplishment over the last year is building Ibotta's Data Engineering team and implementing a data lake solution using AWS and Qubole. He was able to do this in only seven months and now has over 90% of Ibotta's petabyte of data available for analysis. His specialties include Hadoop, MapReduce, Hive, Flume, Airflow, Elasticsearch, Kibana, and Presto.