Chapter 1

History of Analytics and Big Data

There are various definitions of analytics, but in my personal opinion *analytics* is a science and an art at the same time, where the science part is the ability to convert raw data into refined ready-to-use actionable information using the right tools for the right job, while the art part is interpreting the information and the KPIs created to manage and grow your business effectively. Analytics is thus the bridge between data and effective decision making that enables organizations and business leaders to move from making decisions based on gut feel to making decisions based on supporting data.

The question arises as to whether analytics is a new phenomenon or has been around for a long time. While we started to hear about big data and analytics around a decade ago, data has always exceeded computation capacity in one way or the other, and by definition the moment the data exceeds the computational capacity of a system, it can be considered big data. From the earliest records, in 1663 John Graunt recorded the mortality rates in London in order to build an early warning system for the bubonic plague, which led to better accounting systems and systems of record.

In 1865, the term *business intelligence* was first used by Richard Millar Devens in his *Cyclopædia of Commercial and Business Anecdotes* (D. Appleton and company, 1865). Devens coined the term to explain how Sir Henry Furnese gained superior edge over his competitors by collecting information about the environment in which he operated.

In 1880, the US Census Bureau came across as the first known big data problem, where according to the estimates it would have taken them 8 years to process the data calculated in the 1890 census, and the census would take an additional 10 years, meaning that the data growth was outpacing the computational capacity at hand. In 1881, Herman Hollerith, who was working for the bureau as an engineer, invented a tabulating machine that would reduce the work from 10 years to 3 months and came to be known as the father of automated computation. The company he founded came to be known as IBM.

While many other important inventions and discoveries happened throughout the twentieth century, it was in 1989 when the term *big data* was first used by Erik Larkson, who penned an article in *Harper's Magazine* discussing the origin of junk email he received. He famously wrote, "The keepers of big data say they are doing it for the consumer's benefit. But data have a way of being used for purposes other than originally intended."

The scale of data available in the world was first theorized in a paper by Michel Lesk ("How much information there is in the world?"), estimating it to be around 12,000 petabytes in late 1990s (`www.lesk.com/mlesk/ksg97/ksg.html`). Google search debuted in the same year. While there was noise about growing amounts of information, overall the landscape was relatively simple, with the primary mechanism of storing information to be an online transaction process (OLTP) database, and online analytical processing (OLAP) was restricted to fewer large-scale companies.

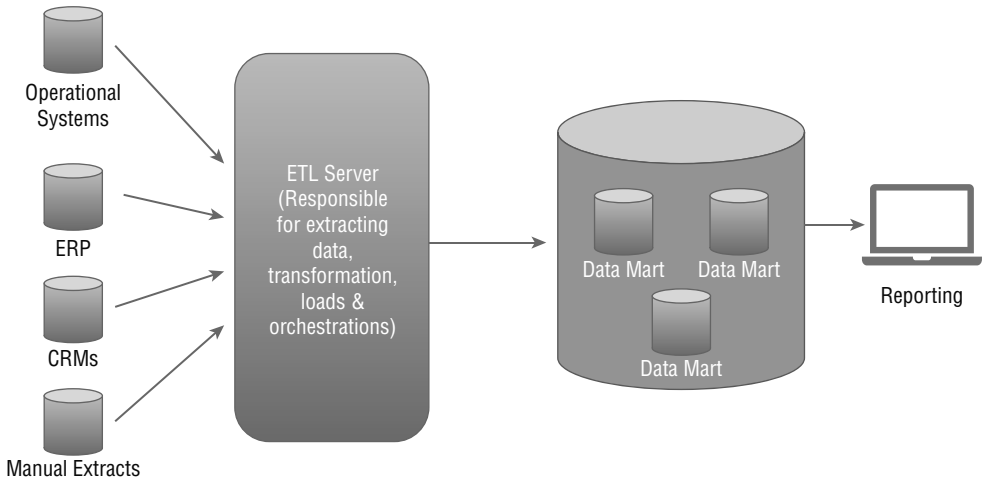
The key takeaway is that big data and analytics have been around for a long time. Even today companies face challenges when dealing with big data if they are using incorrect tools for solving the problems at hand. While the key metrics to define big data include volume, variety, and velocity, the fact is that big data can be any data that limits your ability to analyze it with the tools at your disposal. Hence, for a few organizations, an exabyte scale data would be a big data problem, whereas for others, it can be challenging to extract information from a few terabytes.

Evolution of Analytics Architecture Over the Years

I have been working in the data domain for over 20 years now, and I have seen various stages of evolution of the analytics pipeline.

Around 20 years ago, when you talked about analytics, the audience would consider this some sort of wizardry. In fact, *statistics* was a more common term than *data science*, *machine learning*, *artificial intelligence*, or *advanced analytics*. Having a data warehouse was considered a luxury, and most systems had a standard three-tiered stack, which consisted of multiple sources of data that were typically accessed using either an extraction, transformation, and Loading (ETL) tool (which was considered a luxury) or handwritten ETL scripts using a scripting language of choice, which was, more often than not, traditional shell scripts. Figure 1.1 shows a traditional data warehousing setup that was quite common during early 2000s.

FIGURE 1.1 Traditional data warehousing setup in early to mid-2000s



Data extraction transformation was done on the landing server running ETL scripts and in some cases an ETL tool like Informatica. If you ask me what caused the most pains in this environment, it was this ETL server, which would often be the cause of concern for enterprises due to a variety of reasons, including limited disk space, memory, compute capacity, and the overall orchestration. As you can see, this piece was the "glue" between the sources of data and the potential information and insights that were expected to be extracted from these sources.

The key challenges in such a setup included the rigid process where everything operated in a waterfall model and businesses were asked to provide the requirements really early on in the cycle. It was these requirements which were the basis of creation of an information model which normally took anywhere between 6 to 12 months for large projects. Without any prototyping capabilities, limited scalability, and available tools, it was quite common for business requirements and the eventual delivered product to be vastly different, and quite often by the time the IT delivered the requirements, the business had moved on from the requirements, leading to discontent between the two teams.

The data world was expanding with the dot-com bubble, and ever-increasing system logs were being generated, only to be shoved away in some archival system as the storage was just too expensive, and computation on such large amounts of data was becoming impossible. The world was more content with getting CSVs and TSVs and just not ready for multi-structured data like XML and unstructured data like text. While the CIOs knew there was value in these datasets, the fact that you had to provide a business case for any investment of this scale meant such datasets were often archived or discarded.

The business intelligence (BI) tools were often very rigid, where instead of the users having access to data and self-service analytics, they would have to rely on BI developers who would have to build universes based around an information schema, which would eventually service the user requirements. If you add to this the inability to elastically scale your hardware based on the business needs, you would often come across solution architects working in September of this year to understand and forecast what capacity requirements would be in December of next year and make sure budgets were in place to procure the necessary hardware and software by January, something that would not be required until 12 months later.

Adding to this the strict licensing costs often based on the number of cores your software was running or the amount of data you would store meant the IT budgets were ever increasing, often under increasing scrutiny and IT departments failing to meet the expectations.

The story paints a gloomy picture of what the analytics world looked like 15 to 20 years ago, but things weren't that bad. Enterprises that have been struggling with the challenges of the traditional BI infrastructure had started to look at options, and necessity is the mother of invention. There were a few key requirements that the enterprises realized needed to be met:

1. A distributed storage
2. A distributed compute

3. Scalability of hardware
4. Pay-as-you-go pricing model

Remember, that it was not the fact that distributed storage and compute were unavailable, it was more the realization that the price point at which these technologies were available would make them unsuitable for a number of use cases coming out from the dot-com world. The scalability of hardware was virtually nonexistent, and the strict pricing models meant failure was costly.

The New World Order

The analytics world was undergoing some massive transformation. Google had released its Google FileSystem[3] and MapReduce[4] papers in 2003 and 2004, thus paving the way for open-source projects like Apache Hadoop (discussed in more detail in our data processing chapter), whereas Amazon launched Amazon Web Services[5] in 2006, initially launching just three services—Amazon S3 cloud storage, Amazon SQS, and Amazon EC2—paving the way for a world where the pay-as-you-go model for software and services was now becoming a reality. Hadoop was becoming popular in the world, with enterprises realizing the benefits that some of the larger Internet companies had achieved using their large and highly focused engineering teams. The challenge had been to replicate those successes when the focus of the enterprise was to serve their customers better and now actually build the technology.

Because Hadoop was becoming a popular open-source project, more and more projects started to crop up in the open-source space, like Hive, Pig, Mahout, Zookeeper, and Sqoop. The objective was to enable Hadoop to be used by a larger user base, by not limiting it to the hands of the engineers who could write MapReduce code, but also to enable the business users who are more comfortable with SQL and other well-known languages. Enterprises wanted to get on the Hadoop bandwagon but realized that they not only needed a core distributed storage and compute but also other ecosystem projects with the software, which led to creation of a number of Hadoop distributions, with the most popular ones from Cloudera, Hortonworks, and MapR.

Of course, existing data warehouse vendors wanted to be a part of this change, and some created their own distributions, while others preferred a hardware-based approach by hosting a popular distribution within their appliance and making Hadoop part of the analytics stack, where Hadoop would now start to be a *data lake* for the analytics platform and the existing data warehouse systems would still host the cleanest data. The Hadoop distribution companies were not entirely happy with the relegation of Hadoop to a mere landing zone, which resulted in more innovation up the stack in terms of ease of use of analytics.

While the continuing and high speed of innovation was very beneficial to the end users and they had moved from a point where they had limited choice to a point where they were now spoiled for choice with almost half a dozen Hadoop distributions, some suggested

that Hadoop was a magic bullet, whereas others suggested that Hadoop had its own place in analytics but was not yet ready to replace the "real stuff." Mind you, the open-source world was still coming up with some really cool and important projects aimed at replacing the shortcomings from existing projects. For example, Map Reduce was originally built to solve a search problem, but due to its success it was now being used as an analytics engine, fronted by the likes of Hive and Pig, which resulted in Hadoop (originally a batch processing engine) being used for iterative analytics and thus failing to meet the high expectations of users who were used to getting high-speed access to the data from the enterprise data warehouse systems.

This led to the creation of Apache Spark, a project that would find a new way to solve the data engineering and iterative analytics problem, and then other engines were added to it in the form of Spark SQL (originally called Shark), Spark Streaming, Spark ML, and GraphX. The idea was that this was a one-stop shop for all things analytics.

Since cloud computing was getting popular, more and more customers wanted to use the elastic nature of the cloud to run data engineering workloads (elastic in nature) and asked Amazon Web Services to provide support for such workloads. Customers were running Hadoop workloads on Amazon EC2 instances already; for example, the *New York Times*[6] used 100 EC2 instances to run a Hadoop application to process 4 TB of raw TIFF data (in S3) into 11 million finished PDFs in the space of 24 hours at the computation cost of $240. With the popularity of such workloads and the great demand, Amazon started working on a new project called Amazon Elastic MapReduce (EMR), released in April 2009. The initial objective of EMR was a managed Hadoop environment, but the service later became a catalog of other open-source projects in the big data domain like HBase, Spark, Presto, and Livy.

As the environment looked quite impressive on the data engineering side, the business intelligence (BI) side was also undergoing massive change with business managers increasingly requesting direct access to the data to be able to respond to business challenges more effectively, thus paving the way for tools that allowed fast and ad hoc access to data and demand for more self-service BI tools. This led to popular tools like Tableau, Spotfire, and Qlik, despite being around for a long time, becoming more popular in the market and widely accepted.

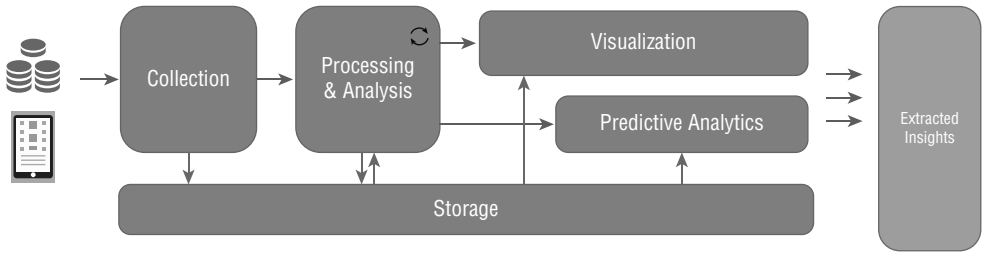
The world of BI and analytics looks very different to what we had 20 years ago, where ETL tools are much more intelligent and BI tools allow you to create a deep learning model with a few clicks. This book talks about the world of BI and analytics that you see today and beyond, and it specifically talks about the AWS platform, which offers the most depth and breadth of solutions around big data and analytics.

Analytics Pipeline

Since this is an advanced-level book, we'll jump directly to the analytics pipeline, which is a critical component that needs to be understood before you undertake any big data project. Figure 1.2 shows a typical big data pipeline that includes various phases of data

management. While there would be cases and situations where you would need to alter the pipeline based on your need, in most cases this is the sort of pipeline the customers would typically build to solve their analytical problems. The technologies might change, but they key idea remains the same.

FIGURE 1.2 Overview of an analytics pipeline



The general flow of a big data pipeline is as follows:

1. Data is collected by a collection tool.
2. Data is stored in a persistent way.
3. Data is processed and analyzed. Often this processing involves multiple steps, and there is typically movement of data between the processing system and the storage system. The result of an analysis can warrant another analysis and processing.
4. Once the processing is done, the data is made available to the business users who can then use the BI tools to visualize and generate further insights from the data.
5. The data is also often used for predictive analytics. Examples are churn prediction, customer segmentation, and product recommendation.

Let's look at each one of these phases in more detail.

Data Sources

In any analytics pipeline, you will start your work with data being produced from producer systems. The system producing data may be an OLTP system like your customer relationship management (CRM) system or your retail point of sale system running on MySQL, Postgres, Aurora, or another commercial database, or it may be your existing data warehouse, like Teradata, Oracle, or Netezza. The producers are typically not limited to relational data sources as they can often include sensors and devices on shops and factory floors producing event data like temperature, humidity, noise, and heat levels or even social media feeds that can be captured from Twitter and Facebook. The data may therefore be in various formats and need to be captured at a varying velocity, variety, and volume. These data sources are what produce the data, and quite often the analytics pipeline's key objective is to transform this raw data into information that can be consumed by business users.

This leads us to the collection part of the big data pipeline.

Collection

Data collection is a first major step in the big data pipeline, and data can originate from a number of sources in a variety of formats and at varying velocity. Having the right tool for the right job is pretty important to building a highly efficient data architecture that includes data collection, and hence the AWS platform provides a number of tools that can be used to collect the data from those sources. One of the major sources of big data include the billions of devices in homes, factories, oil wells, cars, and so on. There is an increasing demand to connect to the devices and collect, store, and analyze that data. This increases the demands on the collection technologies to not only collect the data often in near real time but also to have the ability to analyze at such huge volumes.

High velocity data capture is quite common however Analytics pipelines also encounter data arriving in mini-batches or micro-batches at relatively higher volumes, often needing an entirely different way to capture this data. There are technologies available with the AWS ecosystem that allow you to capture the data in mini-batches and micro-batches thus allowing you to do processing in a serverless fashion, on a pay-as-you-go model. We'll look at these technologies later in this chapter and in more detail during the remainder of the book.

Storage

Once the collection technologies are able to collect the data from the source systems, there is a need to retain this data for downstream processing to ensure it is consumable from other tools that can use this to produce actionable information for the business.

Typically, the need to store the data is as follows:

1. To retain within the collection services
2. To retain in a storage technology of choice

Data collection services like Kinesis (and/or other streaming technologies like Kafka) allow you to store the data in the stream for some period for in-stream analytics. However, you would often need to store data into permanent storage, and that is where permanent storage systems come into play. Often the data is initially landed onto a landing zone, with the core characteristics of the landing zone being the ability to store enormous amounts of data in a variety of file formats, including popular open-source formats like Parquet and ORC, stored at various price points depending on the importance, use, and access of the data.

While the data is stored in a landing zone or a data lake for further processing and analytics, often this is complemented by other storage technologies that are at the extreme end of the spectrum, from sub-second access to long-term archival storage. Since the objective of storage technologies is to store the data for downstream processing and analytics, there are often purpose-built file systems that allow the applications to achieve higher I/O

throughput (100+ Mb/second/core), thus allowing complex applications such as machine learning, deep learning, and high-performance computing (HPC) to consume same datasets. We'll look at the storage technologies in Chapter 3, "Data Storage."

Processing and Analysis

Processing and analysis are often done hand in hand, as the processing or preprocessing of datasets is often required to do meaningful analysis, which may often lead to additional processing needs. The processing can involve converting data from one format to another (e.g., CSV to Parquet), or it can be more complicated processing that involves application of certain business rules to data coming from a variety of sources. This could often involve deduplication processes, data cleansing like replacement of missing values, and so on.

There are a variety of tools available in the market that can be used for data processing, including the most common ones like Hadoop, Spark, and SQL environments like Amazon Redshift. The choice of tool depends on the particular use case, which could be one of the use cases discussed earlier or perhaps building a data mart. We will look into this in more detail in Chapter 4, "Data Processing and Analysis."

Analysis is often an iterative process and ranges from ad hoc analysis to more recurrent query patterns based on standard access paths. Given the variety of choices available at our disposal, like Apache Presto, Amazon Redshift, Apache Hive, Apache Pig, and Apache Spark SQL, we will discuss the selection criteria for the tool based on the specific requirement and use cases at hand in Chapter 4, "Data Processing and Analysis."

Visualization, Predictive and Prescriptive Analytics

Visualization, often referred to as the BI, comprises tools, technologies, and processes that are being used by enterprises to analyze their data and provide information to key stakeholders. Analytics is often categorized by going from hindsight to foresight, and visualization typically looks at the descriptive and diagnostic analytics side of things.

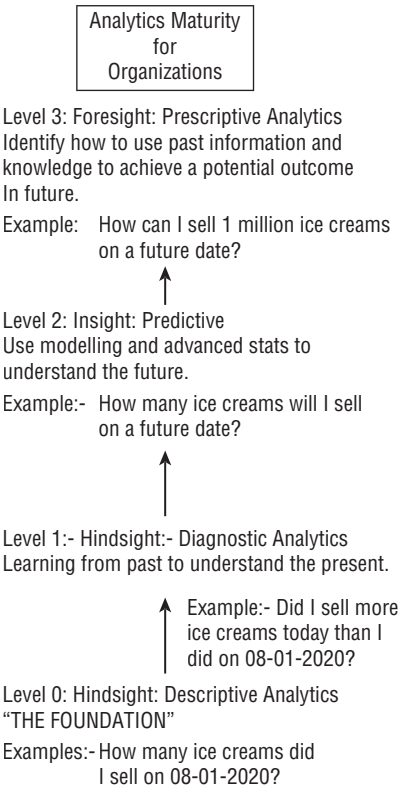
The objective of visualization technologies is to understand what happened, and this is often achieved by a variety of visualization techniques. Enterprises rely on a variety of tools to understand the information available in the data and then move toward drilling down to understand the root cause of the positive or negative impacts. Gartner framed the analytic progression from information to optimization in their analytics ascendancy model published in 2012. Figure 1.3 shows the spectrum of business analytics from analyzing historical data to predicting the future state of affairs for a business and the questions you can expect during each phase of business ascendancy.

As you are aware, past is generally (not always) a good predictive of the future; however, machine learning and AI can be used to predict the future and help businesses plan for future potential outcomes. For example, if a business based on the predictive analytics

realizes that a particular customer is going to churn (`https://tinyurl.com/y9ho4kbu`), they may devise policies to retain the customer using various marketing campaigns. We'll cover descriptive, diagnostic, and predictive analytics later in this book.

FIGURE 1.3 Business analytics spectrum

Source: Gartner Analytics Maturity Model, https://www.zdnet.com/article/a-guide-for-prescriptive-analytics-the-art-and-science-of-choosing-and-applying-the-right-techniques/

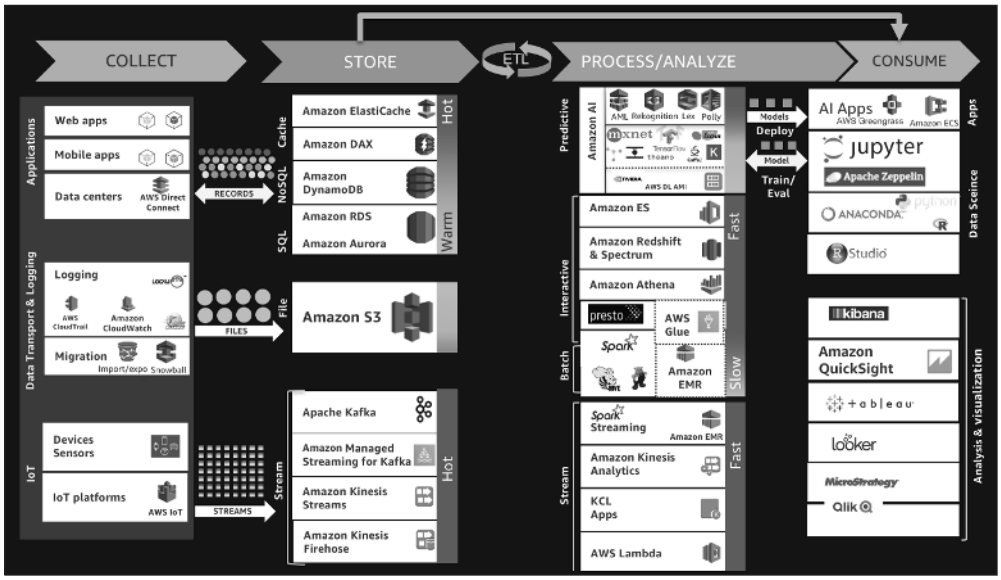


While prescriptive analytics provides recommendations on what needs to be done, the details are out of scope for this book.

The Big Data Reference Architecture

We had a look at the big data pipeline in an earlier section. It is time to now start looking at the Big Data Reference architecture and the AWS and partner components that fit the analytics ecosystem. AWS re:Invent is often a source of some of the most amazing content you will come across on Analytics in general and AWS Analytics ecosystem in particular. Figure 1.4 has been adapted from Siva Raghupathy's talk at re:Invent 2017 showing the various stages of the analytics pipeline and also maps the AWS services to the various stages of the pipeline.

FIGURE 1.4 AWS analytics architecture



Data Characteristics: Hot, Warm, and Cold

Choosing the right technology for processing depends on the characteristics of data, which can vary from cold to hot. You may have heard of the term *multitemperature data management*, where the temperature of the data is determined by the frequency with which the data is accessed, the frequency at which the data is updated and maintained, the levels of durability needed, and the depths of history you would like to maintain.

Let us look at an example of a financial institution that has data for all its customers: their online profile, the transactions that they have done over the years, and so forth. Consider the following set of transactions: The customer is going on `Amazon.com` to purchase some essential items for its daily use. Any financial institution would carry fraud detection on individual transactions; hence, your use case here not only requires you to capture the data in real time, but also to identify if the transaction is fraudulent or not, based on the customer and the transaction profile. The fact that you need to not only capture the transaction in real time but also perform an analytics operation on it indicates that your data collection technology will be defined by the use case, which in this case requires you to do stream processing, and your processing technology will also be defined by the use case, which defines what temperature of data you are dealing with.

As your time to action is relatively lower, this is the case for hot data, and hence the technological choice will be defined where the selected technologies meet your demands of low latency and high-speed access.

Let us consider another transaction or sets of transactions that happened over fourteen years ago. The customer has made a complaint to a regulatory authority about overcharging (such as payment protection insurance, or PPI claims in the UK), and you have been given six weeks to respond to the complaint or reimburse the customer based on this claim.

FIGURE 1.5 Data characteristics for hot, warm, and cold data

	Hot	Warm	Cold
Volume	MB–GB	GB–TB	PB–EB
Item size	B–KB	KB–MB	KB–TB
Latency	ms	ms, sec	min, hrs
Durability	Low–high	High	Very high
Request rate	Very high	High	Low
Cost/GB	$$–$	$–¢¢	¢
	Hot data	Warm data	Cold data

This data is cold by nature, as this is not frequently accessed or updated, and the turn-around time does not have to be sub-seconds, and hence the technological choice will vary compared to the earlier use-case. Figure 1.5 shows how data temperature and the typical attributes of data like volume, item size, latency, durability, request rates, and cost/GB at certain temperature compare.

As seen in Figure 1.5, hot data has typically lower volume, a smaller item size, a lower latency, and typically lower durability, whereas the request rate can often be very high at a higher cost of storage per gigabyte.

At the other end of the spectrum, the cold data has a much larger volume, but the individual items can be of much larger size (sometimes up to a few terabytes), and the latency required is on the higher side. Since cold data is often stored for historical trend analysis or regulatory compliance purposes, the durability needs to be very high but the request rate is on the lower side. You can therefore expect to query your hot data more often than your warm or cold data. You'll see in later chapters that typically the cost per gigabyte is on the lower side, and in fact there are tools available that can use machine learning to move the data from the hot to cold and vice versa based on the request rate, or defined periods of time.

Collection/Ingest

A number of different technologies can be used to collect/ingest data. The type of technology you choose can depend on a variety of factors, including the volume of data being captured, the variety of the data, and the velocity in which it needs to be captured. Generally speaking, there are three major types of data that need to be collected from sources, including but not limited to the following types:

Transactions Transactions include data arriving in various data structures, such as from POS systems, web apps, mobile applications, and so on. Transactions can be captured by various technologies like AWS Glue, AWS Data Pipeline, AWS Data Migration Service, and other partner ETL technologies available on AWS marketplace. An example of this would be your existing data relational database that is perhaps running MySQL and you would like to capture the transactions to perform analytics inside your analytics platform. The data can be captured with the technologies mentioned above. You might also need to capture changes happening to the main data store, and these are also captured using a variety of technologies that we will discuss later in the book.

Files Files include data that is being captured from logging files, media files, and website logs, which often arrive in the form of data files. There are various technologies that allow you to capture the data from files including Flume, AWS CloudTrail, and Amazon CloudWatch. Typically, this is the case when data is generated from media streams or application log files.

Events Events include data that is being generated by your IoT platforms and devices such as noise sensors and temperature sensors and often arrives in the form of streams. However, these events can be any data that is arriving in high velocity that needs to be captured for on-stream or down-stream processing. A number of technologies can be used to capture such data from AWS, including but not limited to Amazon Kinesis, Amazon Managed Streaming for Kafka, Apache Kafka, and some partner solutions.

Storage

AWS offers a variety of storage options. One of the best practices for building a future-proof, scalable, secure, and high-performance analytics platform is to choose the right tool for the right job. Today there is a large choice of tools that you can use, depending on the requirements of your use case. The storage choices include in-memory storage for high-speed process, NoSQL for well-defined access paths, SQL for ad hoc access, file/object storage for data files without any defined structure and for huge volumes at a relatively cheaper price point, and stream storage for low-latency access and analytics.
The AWS platform offers stream storage for hot data, which can either be Kafka-based or Amazon Kinesis-based. Stream storage allows you to decouple your producers and consumers and provide a persistent buffer for your data and ensures client ordering.

Stream storage options:

- Kafka
  - Apache Kafka
  - Managed Streaming for Kafka
- Kinesis
  - Amazon Kinesis Data Streams
  - Amazon Kinesis Firehose

For warm and cold data, Amazon supports Amazon S3 (Simple Storage Service), which is natively supported by major ETL tools and also most popular big data frameworks like Spark, Hive, and Presto. Amazon S3 allows you the option to scale your compute independently of storage. We will look at the data lake architecture later in this chapter and explain how S3 forms the key pillar of an analytics ecosystem built on the cloud. You must have heard a lot about S3and possibly used it as well. The key reasons customers decide to use S3 are as follows:

- Support for infinite volumes of data (the ability to easily scale your storage to exabytes)
- Support for big data frameworks (Spark, Hive, Presto, and so on)

- High durability
- Intelligent tiering (moving the data from high-durability/high-cost to low-durability/low-cost automatically)
- Secure
- Low cost per gigabyte

Some customers tend to look at Apache Hadoop Distributed File System (HDFS) on a Hadoop cluster as a persistent store; however, there are certain cases where HDFS can be used for the hottest datasets (e.g., when working iteratively on a certain dataset), and Amazon S3 can be used for frequently accessed data. We'll discuss this in more detail in Chapter 4, "Data Processing and Analysis."

For the warm and hot data, Amazon supports various options that fall into three major categories:

SQL-Based Systems Amazon offers Amazon RDS, which is a managed relational database service for SQL-based operations to store the data.

NoSQL-Based Systems Amazon offers Amazon DynamoDB and Amazon DocumentDB, which are managed NoSQL services. We will learn more about them in Chapter 3, "Data Storage."

Cache Repositories Amazon also offers Amazon ElastiCache, which is a managed Memcached and Redis service, and Amazon DynamoDB Accelerator (DAX), which is an in-memory cache for DynamoDB to offer low latency and high throughput.

As discussed earlier in this chapter, it is important to pick the right tool for the right job. The decade-old tradition of trying to retrofit every use case in a relational engine, a NoSQL engine, or any other technology that you have had previous success with on a different use case makes things complicated not only from initial implementation but for long-term management as well. The abundance of managed solutions and partner products from the marketplace on the AWS platform allows you to pick the right tool for the job. Your application architecture can now benefit from a variety of in-memory, graph, NoSQL, SQL, and search technologies, thus allowing you to focus on the correct long-term architecture rather than working around the technological limits to ensure that you can focus on the true essence of your business problem. The amount of integration effort that organizations have spent in retrofitting their use cases to a technological limitation has been huge. and the breadth and depth of the AWS ecosystem allows us to shun these anti-patterns.

Process/Analyze

Once the data has been ingested and stored based on the optimum storage option, you can either process the data for downstream analytics or run analytics as is if the data is in a suitable format. The types of processing and analytics depends on a number of factors but can be categorized into the following categories:

Batch Processing and Analytics Batch processing includes workloads that take minutes to hours to run and often include things like daily, weekly, or monthly reports. For batch processing, the technologies that generally work well include Amazon EMR. Customers often use AWS Glue, which is a fully managed (serverless) ETL service that makes it simple and cost effective to categorize your data.

Interactive Analytics Interactive analytics includes workloads that take seconds, and examples include self-service dashboards built using technologies like Amazon QuickSight or technologies like Tableau and ad hoc access with Amazon Athena and Amazon EMR with Presto and Spark.

Stream Processing and Analytics Stream processing analytics includes workloads like fraud detection where the latency is seconds to milliseconds and includes technologies like Spark Streaming (within Amazon EMR), Amazon Kinesis Analytics, KCL, AWS Lambda, Apache Flink, and Kafka.

Predictive Analytics Predictive analytics includes workloads like churn prediction, fraud detection, anomaly detection, demand forecasting, and speech recognition with response time from milliseconds (real-time predictive analytics) to minutes (batch) using technologies like Lex, Polly, Amazon Rekognition, and Amazon machine learning (ML) and deep learning AMIs with frameworks like MxNet, TensorFlow, Theano, Torch, CNTK, and Caffe.

Consumption

Once an entire data pipeline is established to capture, store, and process the data, you then have a number of options to consume the data in various forms. There are various personas who consume the data and insights generated on AWS, and it is important to understand what those personas are and how they consume the information and insights generated on your analytical platform.

Developers and Operations Developers and operations teams often develop, manage, and run apps on EC2 or ECS containers consuming the data assets generated via the data pipeline.

Data Scientists Data scientists work with the data to understand patterns within it, build models, and deploy models in production. They like to work with interactive interfaces like Notebooks (Jupyter, Zepplin), SageMaker, and other data science platforms and IDEs.

Data scientists also like to work directly with APIs (e.g., Spark DataFrame API, Glue DynamicFrame API) to work natively with them and have more control over how the data engineering and data modeling happens.

Business Users Business users consume the data to build reports and dashboards using a variety of tools available on AWS, such as first-class services like Kibana or QuickSight or tools available on the AWS marketplace like Tableau, Qlik, MicroStrategy, and Looker.

Data Lakes and Their Relevance in Analytics

Data is growing at enormous rates, and all industry reports point toward exponential growth of data. The variety of data types include structured and semi-structured data and multiple open-source formats. Based on what we have seen at various customer sites, data grows tenfold every 5 years, and with the average life of a platform around 15 years, a platform built for 1 terabyte today would need to host around 1 petabyte (a thousandfold increase) in 15 years' time. The growth of data with the rise of open-source formats, along with the growth in the number of people accessing the data in a variety of ways, places greater demand on the flexibility in the ways in which the data is accessed, the service-level agreements (SLAs) in which the data is made available, and the massive scale of operation required to achieve optimal results. All of this needs to be done within the security parameters defined for an organization. Customers are demanding an architecture that solves these problems, and one of the answers the industry has to offer for these requirements is a data lake architecture. This is an advanced-level book, and I expect more people to be aware of what a data lake is, but I would still like to recap some of the major concepts.

What Is a Data Lake?

Wikipedia's definition of *data lake*, which can be found at `en.wikipedia.org/wiki/Data_lake`, is quite a comprehensive definition. I often talk to customers about data lakes, and I often emphasize the fact that data lake is a not a technology but rather an architectural concept. The objective of a data lake is to create a central repository where you can store data in a variety of formats, and the repository itself is built on a technology that is scalable in nature and provides ways and means to load and access the data from a variety of tools.

In summary any data lake platform needs to have the following properties:

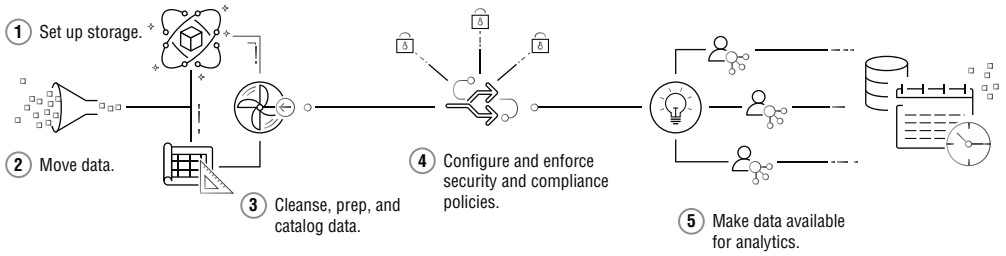
- All data in a single place
- Handles structured/semi-structured/unstructured/raw data
- Supports fast ingestion and consumption
- Schema on read vs. Schema on write
- Designed for low-cost, multi-tiered storage
- Decouples storage and compute—the key objective is to scale the storage tier independently of the compute tier. This is required in cases where your storage would grow exponentially but your compute grows linearly.
- Supports protection and security rules

Now that you have a brief idea of what a data lake is, you might be interested in the key steps involved in building one data lake. The next section, "Building a Data Lake on AWS," explains on a high level the key steps required to build a data lake. Whatever technological choice that you make for a data lake, the steps would remain the same. Figure 1.6 is a pictorial representation of various important steps in building a data lake and has been referenced from AWS's public-facing material on Data Lakes.

There are various steps involved in building a data lake; the major ones are as follows:

1. **Set up storage:** Data lakes hold a massive amount of data. Before doing anything else, customers need to set up storage to hold all of that data in raw formats. If they are using AWS, they would need to configure S3 buckets and partitions. If they are doing this on-premises, they would need to acquire hardware and set up large disk arrays to hold all of the data for their data lake.
2. **Move data:** Customers need to connect to different data sources on-premises, in the cloud, and on IoT devices. Then they need to collect and organize the relevant datasets from those sources, crawl the data to extract the schemas, and add metadata tags to the catalog. Customers do this today with a collection of file transfer mechanisms and ETL tools, like AWS Glue, Informatica, and Talend.
3. **Clean and prepare data:** Next, the data must be carefully partitioned, indexed, and transformed to columnar formats to optimize for performance and cost. Customers need to clean, deduplicate, and match related records. Storing data in columnar formats benefits the analytical workloads, which often operate on a subset of the columns rather than the entire list. Today this is done using rigid and complex SQL statements that only work so well and are difficult to maintain. This process of collecting, cleaning, and transforming the incoming data is complex and must be manually monitored in order to avoid errors.
4. **Configure and enforce policies:** Sensitive data must be secured according to compliance requirements. This means creating and applying data access, protection, and compliance policies to make sure you are meeting required standards: for example, restricting access to personally identifiable information (PII) at the table, column, or row level; encrypting all data; and keeping audit logs of who is accessing the data. Today customers use access control lists on S3 buckets, or they use third-party encryption and access control software to secure the data. And for every analytics service that needs to access the data, customers need to create and maintain data access, protection. and compliance policies for each one. For example, if you are running analysis against your data lake using Redshift and Athena, you need to set up access control rules for each of these services.
5. **Make data available to analysts:** Different people in your organizations, like analysts and data scientists, may have trouble finding and trusting datasets in the data lake. You need to make it easy for those end users to find relevant and trusted data. To do this you must clearly label the data in a catalog of the data lake and provide users with the ability to access and analyze this data without making requests to IT.

FIGURE 1.6 Typical steps in building a data lake



The steps in the data lake are defined clearly. The following list presents a set of best practices that are important while building a data lake project. The key architectural principles for your data lake and analytics project are as follows:

Decouple your data bus: data > store > process > analyze > answers. For solution architects, this is the key to building a scalable architecture. Each individual piece of the data bus needs to be decoupled, thus allowing each of the pieces to be replaced by different components if and when needed or scaled independently of the other. If your storage is growing faster than your compute, it is important to separate both of them to ensure that you can scale one of them independently of the other.

Don't retrofit use cases into the tool: data structure, latency, throughput, access patterns. There is an old saying that if all you own is hammer, everything looks like a nail. This architectural principle is the key to building an architecture that meets the needs of your business rather than limiting your implementation because of a choice of a tool that was made based on certain other requirements in the past. Companies were traditionally limiting themselves to a single tool because of the huge costs involved in acquiring new software. It is always important to remember that with the AWS cloud, you have a number of different tools available, each meeting a particular business need, and hence using the appropriate tool is not only cost effective but also provides better performance. It goes without saying that this helps the organization to focus on the actual business problem and innovation rather than spending time building the data pipelines. The benefit of the AWS platform is that you will only be paying for the amount of usage rather than the number of tools you are using, and hence the cost of failure is minimal, thus allowing you to experiment and innovate.

Use managed services whenever and wherever possible: scalable/elastic, available, reliable, secure, no/low admin. The most successful organizations I have worked with are the ones that are focusing on solving their business problems rather than spending resources on the technical plumbing, which is not their core business. For example, if your core business is running a financial services organization, you would gain little value by running your own fleet of clusters of a particular technology. Most successful customers would opt for managed services rather than opting to build complex technology stacks that are hard to build but even harder to maintain and extend.

Use log-centric design patterns: immutable logs (data lake), materialized views. It is important to ensure that the original data loaded to the data lake or arriving at the

analytics platform remains immutable and does not change. I often see a few customers who change the data from one format to another using any given transformation tool but then discarding the original source data. This is a non-recommended approach for most use cases as it leads to an inability to trace any reporting inaccuracies back to the original source. The value of keeping the original data in its original state cannot be emphasized more, other than stating that it is quintessential to keep the raw data in the raw format for data lineage and compliance reasons.

Be cost conscious: big data should not be big cost. Big data does not imply that the cost of building an analytics pipeline should be big too. Decoupling storage and compute and using multi-tiered storage would allow you to reduce your storage and compute costs. Furthermore, using managed services would allow you to reduce your operational and management costs.

AI/ML enable your applications. As an end user developing products, you need to understand that having AI/ML in your applications and products is no more a nice-to-have, and in fact you need to rely on the data that you produce to build more effective models and use them to improve your products.

Now that we have looked at the concepts of a data lake, the steps behind building a data lake, and some of the key architectural patterns, let us look at how you can build a data lake on AWS.

Building a Data Lake on AWS

AWS offers a depth and breadth of different services that are useful in building the data lake on the platform. While Figure 1.7 looks at the AWS platform, a number of these technologies can be replaced by other products from the partner ecosystem that offer the same features and functions. We'll look at the different steps required to build a data lake on AWS.

Step 1: Choosing the Right Storage – Amazon S3 Is the Base

Amazon S3 is the central pillar of the AWS data lake story. Amazon S3 is secure and highly scalable and offers durable object storage with millisecond latency for data access. It allows you to store any type of data, from unstructured (logs, dump files) and semi-structured (XML, JSON) to structured data (CSV, Parquet), including data from websites, mobile apps, corporate applications, and IOT sensors at any scale.

Amazon S3 integrates storage lifecycle management and uses machine learning to move data from S3-Standard Access to S3-Infrequent Access, thus optimizing the cost for your analytics platform. S3 also allows you to create lifecycle policies for deep archive, ensuring that all data can be stored at an optimum price point.

Amazon S3 includes inbuilt disaster recovery by having the data replicated across multiple AZs and provides very high aggregated bandwidth.

In addition, Amazon S3 natively supports big data frameworks like Spark, Hive, Presto, and many others, which makes the partner ecosystem very strong. A lot of companies have moved

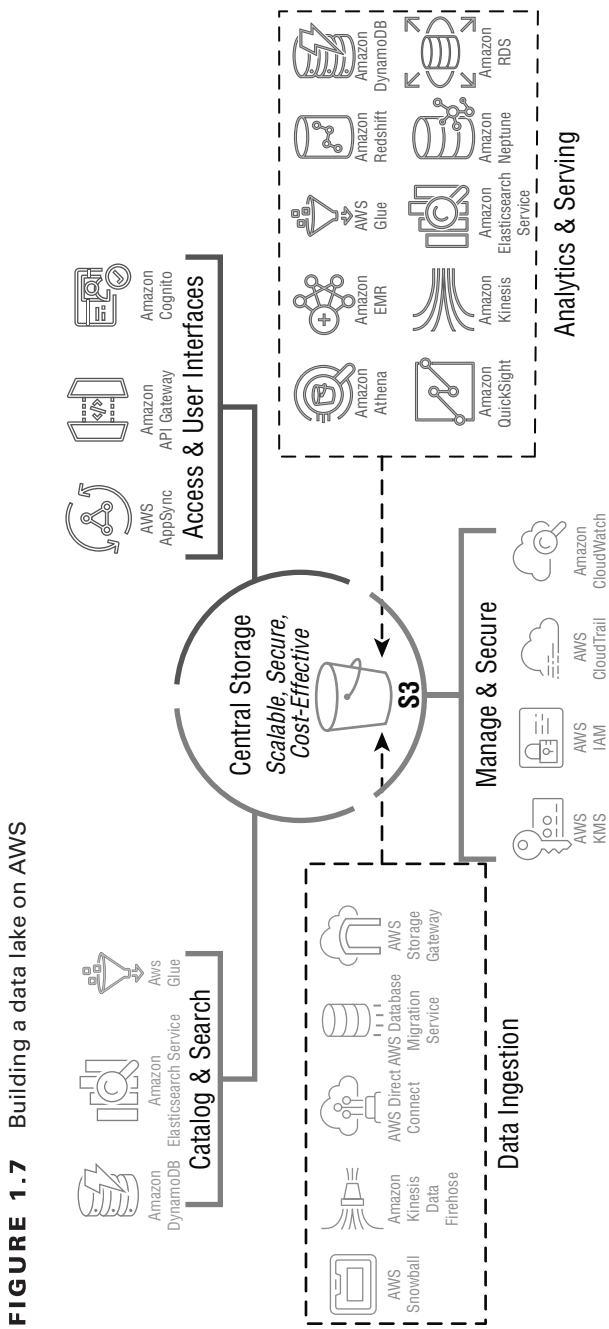


FIGURE 1.7 Building a data lake on AWS

to S3 from Hadoop Distributed File System (HDFS), with one of the major reasons being S3's ability to apply various computation techniques on the same dataset, thus effectively separating compute and storage. Amazon S3 supports 99.999999999% (eleven nines) of durability and provides huge volumes of storage at a fraction of the cost of traditional data storage options.

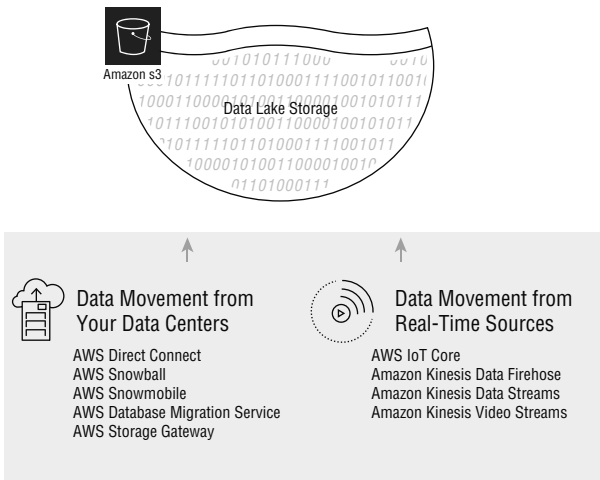
Step 2: Data Ingestion – Moving the Data into the Data Lake

Once you have identified S3 as your core storage engine, you need to find ways to move the data into S3. Data movement depends on a variety of factors, including the volume, variety, and velocity at which it is appearing. If you look at the left-hand side of Figure 1.8, you can see that AWS provides a number of data ingestion options, with each technology working best in certain situations.

If you need to move data from on-premises data centers to S3 and you require high throughput and bandwidth, you might need to provide a dedicated connection between the source data center and your data lake in S3. AWS Direct Connect would be the right choice in such a case. We would look at Direct Connect in more detail during in Chapter 2, "Data Collection" later in the book.

Moving large amounts of data is always tricky in data migrations. If you would like to move multiple terabytes of data into S3, the S3 CLI, which operates over an Internet connection, might not be the fastest option, and hence options like AWS Snowball make more sense. If your data exceeds a few terabytes and is in the petabyte or exabyte zone, AWS provides options like AWS Snowmobile. We'll look at each of these options in Chapter 2, "Data Collection" later in the book.

FIGURE 1.8 Moving data into S3



Data from IOT devices like sensors arrives in high velocity and needs to be captured for further processing. Amazon offers engines like Kinesis Firehose, which can be used to capture the data and store it in Amazon S3 for other analytical processing.

When moving data from databases onto Amazon S3, there are a couple of options when it comes to AWS native services. AWS Glue and AWS Database Migration Service can be used to move data from existing SQL database engines into an S3 data lake.

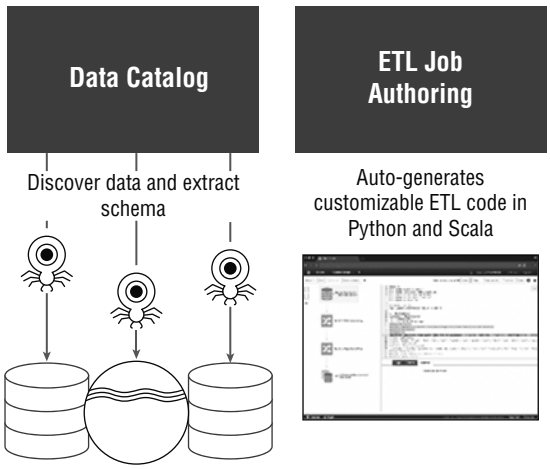
We'll look into each of these services in more detail in Chapter 2, "Data Collection" and Chapter 4, "Data Processing and Analysis."

Step 3: Cleanse, Prep, and Catalog the Data

Once the data has been acquired from the source systems and brought into Amazon S3, it is a common practice to cleanse, prep, and catalog the data. Amazon provides AWS Glue, which is a serverless ETL engine that allows you to automatically discover the data and identify its schema, making the metadata searchable to author other ETL jobs. Glue will allow you to generate code that can be customized, and schedule your ETL jobs, allowing you to bookmark your progress. Figure 1.9 shows the Data Catalog and ETL Job authoring aspects of AWS Glue. As Glue is a managed service, Glue users benefit from the serverless nature of the platform, where they develop their ETL on the console, and Glue runs the jobs in Spark clusters at the backend, managing the spin up of clusters, execution of the jobs, and shutdown of the clusters without manual user intervention. We'll look at AWS Glue in Chapter 2, "Data Collection" later in this book.

Building a catalog of your data is essential and invaluable as it allows other AWS services like Athena, EMR, and Redshift to reuse the catalog. We'll look into the details of the catalog and another service called AWS Lake Formation, which simplifies the process of building a data lake, later in this book.

FIGURE 1.9 AWS Glue



Step 4: Secure the Data and Metadata

Since your data is in a central place, and you have already extracted the metadata for your data lake in the form of a catalog, it becomes critical to secure not only the data but also the associated metadata. By default, all Amazon S3 buckets are private, and you have the ability to use resource-based policies (S3 bucket policies and IAM user policies) and KMS to enable client and server-side encryption for your data. S3 also allows you to use object tagging in conjunction with IAM. We'll look at all of these in Chapter 3, "Data Storage" and Chapter 6, "Data Security" where we will be discussing S3 in more detail.

As discussed earlier, you can use AWS Glue to furnish metadata for the data available without your data lake, and the metadata can be secured using various policies.

1. Use of identity-based policies to secure metadata:
   a. AWS IAM policies that are managed by IAM
   b. AWS IAM policies, which are attached to IAM principals such as IAM users and IAM roles
2. Use of resource-based policies to secure metadata:
   a. Resource-based policies are managed by AWS Glue.
   b. One policy per account/catalog
   c. Allows cross-account access

We'll look into these in much more detail in Chapter 6, "Data Security."

Step 5: Make Data Available for Analytics

Once the data has been acquired, stored, and preprocessed, the next step is to make data available for analytics. There are various analytics services available, like Redshift, Amazon ElasticSearch Service, EMR, and QuickSight, to name a few within AWS, with each one of them serving a variety of use cases. We've looked on a higher level at the types of services available in the Process/Analyze section earlier in this chapter, and will look to go into more detail during the Chapter 4, "Data Processing and Analysis" and Chapter 5, "Data Visualization" later in this book.

Using Lake Formation to Build a Data Lake on AWS

As you can see from the process just discussed, the building of a data lake is often a very involved process, and once the lake is built, setting up the security to ensure that different downstream services can access the data in the lake can be quite laborious, and it can often take months to set up the lake internally. Amazon announced AWS Lake Formation during

re:Invent 2018 to automate the process of building a data lake in a few clicks from a single unified dashboard. The objective of the service is to reduce the time to set up a data lake from months to days.

AWS simplifies the process of setting up a data lake by bringing the various stages involved to a single console. AWS Lake Formation allows you to crawl your data sources, move the data into an Amazon S3 data lake, and then uses machine learning to automatically lay out the data into Amazon S3 partitions and change it into a columnar format like Parquet and ORC for faster analytics.

Deduplication of data is one of the most critical data processing tasks that is required since the data is coming from a number of different sources. AWS Lake Formation allows you to deduplicate and find matching records to increase the data quality.

Setting up access and permissions at the user, table, and column level is also a challenge in a traditional data lake build and usually consumes lots of time during the data lake build phase.

AWS Lake Formation allows you to set up all the permissions for the data lake from a single screen, which will then be implemented across all services consuming the data from the data lake, including but not limited to Amazon Redshift, Amazon Athena, and Amazon EMR.

Now that we have looked on a higher level at the services, let's look at the exam objectives. This book is essentially written to help you learn more about AWS Analytics stack but also to pass the AWS Certified Data Analytics - Specialty certification.

Exam Objectives

The *AWS Certified Data Analytics – Specialty certification exam* is intended for people who have experience designing and building analytical projects on AWS. You should understand the following exam concepts:

- Designing, developing, and deploying cloud-based solutions using AWS
- Designing and developing analytical projects on AWS using the AWS technology stack
- Designing and developing data pipelines
- Designing and developing data collection architectures
- An understanding of the operational characteristics of the collection systems
- Selection of collection systems that handle frequency, volume, and the source of the data
- Understanding the different types of approaches of data collection and how the approaches differentiate from each other on the data formats, ordering, and compression
- Designing optimal storage and data management systems to cater for the volume, variety, and velocity
- Understanding the operational characteristics of analytics storage solutions

- Understanding of the access and retrieval patterns of data
- Understanding of appropriate data layout, schema, structure, and format
- Understanding of the data lifecycle based on the usage patterns and business requirements
- Determining the appropriate system for the cataloging of data and metadata
- Identifying the most appropriate data processing solution based on business SLAs, data volumes, and cost
- Designing a solution for transformation of data and preparing for further analysis
- Automating appropriate data visualization solutions for a given scenario
- Identifying appropriate authentication and authorization mechanisms
- Applying data protection and encryption techniques
- Applying data governance and compliance controls

Objective Map

The following table lists each domain and its weighting in the exam, along with the chapters in the book where that domain's objectives and subobjectives are covered.

Domain	Percentage of Exam	Chapter
Domain 1.0: Data Collection	18%	1, 2,3
1.1– Determine the operational characteristics of the collection system.		
1.2– Select a collection system that handles the frequency, volume and source of data.		
1.3– Select a collection system that addresses the key properties of the data, such as order, format and compression.		
Domain 2.0: Storage and Data Management	22%	3,4
2.1– Determine the operational characteristics of the analytics storage solution.		
2.2– Determine data access and retrieval patterns.		
2.3– Select appropriate data layout, schema, structure and format.		
2.4– Define a data lifecycle based on usage patterns and business requirements.		
2.5– Determine the appropriate system for cataloguing data and managing metadata.		

Domain	Percentage of Exam	Chapter
Domain 3.0: Processing	20%	3,4
3.1– Determine appropriate data processing solution requirements.		
3.2– Design a solution for transforming and preparing data for analysis.		
3.3– Automate and operationalize a data processing solution.		
Domain 4.0: Analysis and Visualization	16%	3, 4,5
4.1– Determine the operational characteristics of an analysis and visualization layer.		
4.2– Select the appropriate data analysis solution for a given scenario.		
4.3– Select the appropriate data visualization solution for a given scenario.		
Domain 5.0: Security	24%	2, 3, 4, 5, 6
5.1– Select appropriate authentication and authorization mechanisms.		
5.2– Apply data protection and encryption techniques.		
5.3– Apply data governance and compliance controls.		

It is important to understand and note that this book will provide you with an understanding of the exam objectives: however, the exam covers such a vast area of expertise that it is impossible to cover in a single book. It is highly recommended that you ensure reading additional topics as discussed in the further reading sections at the end of each chapter. We are also providing sample labs and workshops that will help you get acquainted with the technology under discussion. Finally, it is certainly likely that you will get questions in the exam which are around topics which are not directly addressed in the book, and hence a reading of the whitepapers, FAQs, and supplemental reading material is key to acing the exam.

Assessment Test

The following assessment test will give you a general idea about your analytics skillset on AWS and can identify areas where you should apply more focus. This test touches upon some basic concepts but will give you an indication of what types of questions you can expect from the AWS Analytics Certified Data Specialty certification exam.

1. An organization looking to build a real-time operational analytics dashboard for its mobile gaming application is looking at various options to build the dashboard. Which of the following options will provide the right performance characteristics for such an application?
   - **A.** Use Amazon S3 to power the dashboard.
   - **B.** Use Amazon Redshift to power the dashboard.
   - **C.** Use Amazon Elasticsearch Service to power the dashboard.
   - **D.** Use Amazon DynamoDB to power the dashboard.

2. An administrator has a 6 GB file in Amazon S3. The administrator runs a nightly COPY command into a 2-node (32 slices) Amazon Redshift cluster. The administrator wants to prepare the data to optimize performance of the COPY command. What is the best way for the administrator to prepare the data?
   - **A.** Compress the file using gzip compression.
   - **B.** Split the file into 64 files of equal size.
   - **C.** Split the file into 500 smaller files.
   - **D.** Split the file into 32 files of equal size.

3. A customer wants to build a log analytics solution on AWS with sub-second latency for the search facility. An additional requirement is to build a dashboard for the operations staff. Which of the following technologies would provide a more optimal solution?
   - **A.** Store the logs in Amazon S3 and use Amazon Athena to query the logs. Use Amazon QuickSight to build the dashboard.
   - **B.** Store the logs in Amazon Redshift and use Query Editor to access the logs. Use Amazon QuickSight to build the dashboard.
   - **C.** Store the logs in Amazon Elasticsearch Service and use Kibana to build a dashboard.
   - **D.** Store the logs in HDFS on an EMR cluster. Use Hive to query the logs. Use Amazon QuickSight to build the dashboard.

4. A leading telecommunications provider is moving to AWS and has around 50 TB of data in its on-premises Hadoop environment, stored in HDFS, and is using Spark SQL to analyze the data. The customer has asked for a cost-effective and efficient solution to migrate onto AWS quickly, over a 100 mbps (megabits per second) connection, to be able to build a catalog that can be used by other services and analyze the data while managing as few servers as possible. Which solution would be best for this customer?
   - **A.** Migrate the data using S3 commands using CLI interface to Amazon S3, use AWS Glue to crawl the data and build a catalog, and analyze it using Amazon Athena.

B. Migrate the data to S3 using AWS Snowball, use AWS Glue to crawl the data and build a catalog, and analyze it using Amazon Athena.

C. Migrate the data using Amazon Snowball to Amazon S3, use Hive on EMR running Spark to build a catalog, and analyze the data using Spark SQL.

D. Migrate the data using CLI interface to AMAZON S3, use Hive on EMR running Spark to build a catalog, and analyze the data using Spark SQL.

5. A leading financial organization is looking to migrate its enterprise data warehouse to AWS. It has 30 TB of data in its data warehouse but is only using 500 GB for reporting and dashboarding while the remaining data is occasionally required for compliance purposes. Which of the following is a more cost-effective solution?

A. Migrate the data to AWS. Create an EMR cluster with attached HDFS storage hosting the 30 TB of data. Use Amazon QuickSight for dashboarding and Hive for compliance reporting requirements.

B. Migrate the data to Amazon S3. Create a Redshift cluster hosting the 30 TB of data. Use QuickSight for dashboarding and Athena for querying from S3.

C. Migrate the data to Amazon S3. Create a Redshift cluster hosting the 500 GB of data. Use Amazon QuickSight for dashboarding and Redshift Spectrum for querying the remaining data when required.

D. Migrate the data to AWS in an Amazon Elasticsearch Service cluster. Use Kibana for dashboarding and Logstash for querying the data from the ES cluster.

6. An upcoming gaming startup is collecting gaming logs from its recently launched and hugely popular game. The logs are arriving in JSON format with 500 different attributes for each record. The CMO has requested a dashboard based on six attributes that indicate the revenue generated based on the in-game purchase recommendations as generated by the marketing departments ML team. The data is on S3 in raw JSON format, and a report is being generated using QuickSight on the data available. Currently the report creation takes an hour, whereas publishing the report is very quick. Furthermore, the IT department has complained about the cost of data scans on S3.

They have asked you as a solutions architect to provide a solution that improves performance and optimizes the cost. Which of the following options is most suitable to meet the requirements?

A. Use AWS Glue to convert the JSON data into CSV format. Crawl the converted CSV format data with Glue Crawler and build a report using Amazon Athena. Build the front end on Amazon QuickSight.

B. Load the JSON data to Amazon Redshift in a VARCHAR column. Build the report using SQL and front end on QuickSight.

C. Load the JSON data to Amazon Redshift. Extract the reportable attributes from the JSON column in the tables. Build the report using SQL and front end on QuickSight.

D. Use AWS Glue to convert the JSON data into Parquet format. Crawl the converted Parquet format with Glue Crawler, and build a report using Athena. Build the front end on Amazon QuickSight.

7. A leading manufacturing organization is running a large Redshift cluster and has complained about slow query response times. What configuration options would you check to identify the root cause of the problem?
   - **A.** The number and type of columns in the table
   - **B.** Primary and secondary key constraints
   - **C.** Alignment of the table's sort key with predicates in the SELECT statement
   - **D.** The number of rows in the table
   - **E.** The partition schema for the database

8. Your customer has asked you to help build a data lake on S3. The source system is MySQL, Oracle, and standard CSV files. The data does not have any incremental key. The customer expects you to capture the initial data dump and changes during the data lake build phase. What tools/technologies would you recommend to reduce the overall cost of migration?
   - **A.** Load the initial dump and changes using AWS Glue.
   - **B.** Load the initial dump and changes using DMS (Database Migration Service).
   - **C.** Load the initial dump with AWS Glue and capture changes with AWS Glue and DMS (Database Migration Service).
   - **D.** Load the initial dump using DMS (Database Migration Service) and changes in the data with AWS Glue.

9. A QuickSight dashboard allows you to view the source data but not make any changes to it.
   - **A.** True
   - **B.** False

10. Amazon QuickSight can interpret your charts and tables for you and suggest insights in plain English.
    - **A.** True
    - **B.** False

References

1. www.weforum.org/agenda/2015/02/a-brief-history-of-big-data-everyone-should-read/
2. www.kdnuggets.com/2017/07/4-types-data-analytics.html
3. research.google.com/archive/gfs-sosp2003.pdf
4. research.google.com/archive/mapreduce-osdi04.pdf
5. en.wikipedia.org/wiki/Amazon_Web_Services
6. open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/
7. AWS Re-Invent Presentation by Siva Raghupathy in 2017 www.youtube.com/watch?v=a3713oGB6Zk