**15**

**Data Lake Patterns – Integrating Your Data across the Enterprise**

Today, technology companies like Amazon, Google, Netflix, and Facebook drive immense success as they can get insight from their data and understand what customers want. They personalize the experience in front of you, for example, movie suggestions from Netflix, shopping suggestions from Amazon, and search selections from Google. All of their success is credited to being able to dig into data and utilize that for customer engagement. That’s why data is now considered the new oil.

Picture this – you are getting ready to watch television, excited to see your favorite show. You sit down and try to change the channel, only to find that the remote control is not working. You try to find batteries. You know you have some in the house but you don’t remember where you put them. Panic sets in, and you finally give up looking and go to the store to get more batteries.

A similar pattern repeats over and over in today’s enterprises. Many companies have the data they need to survive and thrive but need help accessing the data effectively, turning it into actionable and valuable information, and getting that information to the right people on a timely basis.

The data lake pattern is handy in today’s enterprise to overcome this challenge. In this chapter, you will learn about data lakes through the following main topics:

* The definition of a data lake
* The purpose of a data lake
* Data lake components
* AWS Lake Formation
* Data lake best practices
* Key metrics of a data lake
* Lakehouse and data mesh architecture
* Choosing between a data lake, lakehouse, and data mesh architecture

Let’s dive deep into the world of data and ways to get meaningful data insight.

**Definition of a data lake**

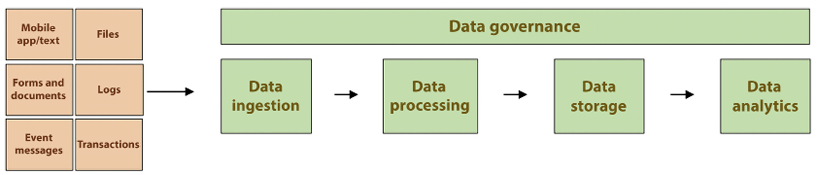
Data is everywhere today. It was always there, but it was too expensive to keep it. With the massive drops in storage costs, enterprises keep much of what they were throwing away before. And this is the problem. Many enterprises are collecting, ingesting, and purchasing vast amounts of data but need help to gain insights from it. Many Fortune 500 companies are generating data faster than they can process it. The maxim *data is the new gold* has a lot of truth, but just like gold, data needs to be mined, distributed, polished, and seen.

The data that companies are generating is richer than ever before, and the amount they are generating is growing at an exponential rate. Fortunately, the processing power needed to harness this data deluge is increasing and becoming cheaper. Cloud technologies such as AWS allow us to process data almost instantaneously and in a massive fashion.

A **data lake** is an architectural approach that helps you manage multiple data types from a wide variety of structured and unstructured sources through a unified set of tools. A data lake is a centralized data repository containing structured, semi-structured, and unstructured data at any scale. Data can be stored in its raw form without any transformations, or some preprocessing can be done before it is consumed. From this repository, data can be extracted and consumed to populate dashboards, perform analytics, and drive machine learning pipelines to derive insights and enhance decision-making. Hence, the data stored in a data lake is readily available to be categorized, processed, analyzed, and consumed by diverse organizational groups.

Data lakes allow you to break down data silos and bring data into a single central repository, such as Amazon S3. You can store various data formats at any scale and at a low cost. Data lakes provide you with a single source of truth and allow you to access the same data using a variety of analytics and machine learning tools.

The following diagram shows the key components of modern data architecture:



*Figure 15.1: Key components of a modern data lake*

The above diagram shows modern data architecture where data is ingested in different formats such as logs, files, messages, documents, and so on. After that, this data is processed as per business needs, and processed data get stored and consumed by various businesses for data analytics.

The key considerations of modern data lakes include the ability to handle the increasing volume, velocity, and variety of data where each component, such as data storage and processing, should be independently scalable and data should be easily accessible by various stack holders.

**The purpose of a data lake**

You might not need a data lake if your company is a bootstrap start-up with a small client base. However, even the smaller entities that adopt the data lake pattern in their data ingestion and consumption will be nimbler than their competitors. Especially if you already have other systems in place, adopting a data lake will come at a high cost. The benefits must outweigh these costs, but this might be the difference between crushing your competitors and being thrust into the pile of failed companies in the long run.

The purpose of a data lake is to provide a single store for all data types, structures, and volumes, to support multiple use cases such as big data analytics, data warehousing, machine learning, and more. It enables organizations to store data in its raw form and perform transformations as needed, making it easier to extract value from data. When you are building a data lake, consider the following five V’s of big data:

1. **Volume**: Refers to the sheer amount of data generated and stored by various sources, such as social media, IoT devices, and transactional systems. For example, a large retailer may generate and store petabytes of data from online and in-store sales transactions, customer behavior data, and product information.
2. **Velocity**: Refers to the speed at which data is generated and processed. Data can be generated in real time, such as stock market prices or weather readings from IoT devices. For example, a financial firm may need to process high-frequency stock market data in real time to make informed trading decisions.
3. **Variety**: Refers to the different types of data generated by various sources, including structured data (for example, relational databases), semi-structured data (for example, XML, JSON), and unstructured data (for example, text, images, and audio). For example, a healthcare organization may need to process and analyze a variety of data types, including electronic medical records, imaging data, and patient feedback.
4. **Veracity**: Refers to the uncertainty, ambiguity, and incompleteness of data. Big data often comes from sources that could be controlled better, such as social media, and may contain errors, inconsistencies, and biases. For example, a political campaign may use social media data to gain insights into public opinion but must be aware of the potential for false or misleading information.
5. **Value**: Refers to the potential of data to provide insights and drive business decisions. The value of big data lies in its ability to reveal patterns, trends, and relationships that can inform strategy and decision-making. For example, a retail company may use big data analytics to identify purchasing patterns and make personalized product recommendations to customers.

Some of the benefits of having a data lake are as follows:

* **Increasing operational efficiency**: Finding your data and deriving insights from it becomes more accessible with a data lake.
* **Making data more available across the organizations and busting silos**: Having a centralized location will enable everyone in the organization to access the same data if they are authorized to access it.
* **Lowering transactional costs**: Having the correct data at the right time and with minimal effort will invariably result in lower costs.
* **Removing load from operational systems such as mainframes and data warehouses**: Having a dedicated data lake will enable you to optimize it for analytical processing and enable you to optimize your operational systems to focus on their primary mission of supporting day-to-day transactions and operations.

C-Suite executives are no longer asking, *“Do we need a data lake?”* but rather, *“How do we implement a data lake?”* They realize that many of their competitors are doing the same, and studies have shown that organizations derive real value from data lakes. An Aberdeen survey saw that enterprises that deploy a data lake in their organization could outperform competitors by 9% in incremental revenue growth. You can find more information on the Aberdeen survey here: <https://tinyurl.com/r26c2lg>.

**Components of a data lake**

The concept of a data lake can vary in meaning to different individuals. As previously mentioned, a data lake can consist of various components, including both structured and unstructured data, raw and transformed data, and a mix of different data types and sources. As a result, there is no one-size-fits-all approach to creating a data lake. The process of constructing a clean and secure data lake can be time-consuming and may take several months to complete, as there are numerous steps involved in the process. Let’s take a look at the components that need to be used when building a data lake:

* **Data ingestion**: The process of collecting and importing data into the data lake from various sources such as databases, logs, APIs, and IoT devices. For example, a data lake may ingest data from a relational database, log files from web servers, and real-time data from IoT devices.
* **Data storage**: The component that stores the raw data in its original format without any transformations or schema enforcement. Typically, data is stored in a distributed file system such as Hadoop HDFS or Amazon S3. For example, a data lake may store petabytes of data in its raw form, including structured, semi-structured, and unstructured data.
* **Data catalog**: A metadata management system that keeps track of the data stored in the data lake, including data lineage, definitions, and relationships between data elements. For example, a data catalog may provide information about the structure of data in the data lake, who created it, when it was created, and how it can be used.
* **Data processing**: The component that performs transformations on the data to prepare it for analysis. This can include data cleansing, enrichment, and aggregation. For example, a data processing layer may perform data cleansing to remove errors and inconsistencies from the data or perform data enrichment to add additional information to the data.
* **Data analytics**: The component that provides tools and technologies for data analysis and visualization. This can include SQL engines, machine learning libraries, and visualization tools. For example, a data analytics layer may provide an SQL engine for running ad-hoc queries or a machine learning library for building predictive models.
* **Data access**: The component that provides access to the data in the data lake, including data APIs, data virtualization, and data integration. For example, a data access layer may provide APIs for accessing data in the data lake, or data virtualization to provide a unified view of data from multiple sources.

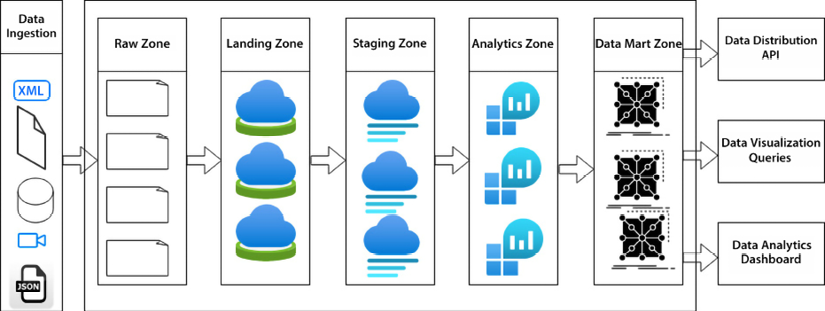
It is common to divide a data lake into different zones based on data access patterns, privacy and security requirements, and data retention policies. Let’s look at various data lake zones.

**Data lake zones**

Creating a data lake can be a lengthy and demanding undertaking that involves substantial effort to set up workflows for data access and transformation, configure security and policy settings, and deploy various tools and services for data movement, storage, cataloging, security, analytics, and machine learning. In general, many data lakes are implemented using the following logical zones:

* **Raw zone**: This is the initial storage area for incoming data in its original format without any modification or transformation. The data is stored as-is, allowing for easy access and preservation of its original state. An example use case for this zone could be storing social media data as raw JSON files for later analysis.
* **Landing zone**: The landing zone is a temporary storage area for incoming data that undergoes basic validation, quality checks, and initial processing. Data is moved from the raw zone to the landing zone before being moved to the next stage. An example use case for the landing zone could be performing data quality checks such as duplicate removal and data type validation on incoming sales data before it is moved to the next stage.
* **Staging zone**: The staging zone is where data is transformed and integrated with other data sources before being stored in the final storage area. The data is processed in this zone to prepare it for analysis and to help ensure consistency and accuracy. An example use case for the staging zone could be transforming incoming sales data into a common format and integrating it with other data sources to create a single view of the data.
* **Analytics zone**: The analytics zone is optimized for data analysis and exploration. Data stored in this zone is made available to data scientists, business analysts, and other users for reporting and analysis. An example use case for this zone could be storing sales data in a columnar format for faster querying and analysis.
* **Data mart zone**: The data mart zone is used to create isolated and curated data subsets for specific business or operational use cases. This zone is optimized for specific business requirements and can be used to support reporting, analysis, and decision-making. An example use case for the data mart zone could be creating a data subset of sales data for a specific product line for analysis.
* **Archive zone**: The archive zone is the final storage area for data that is no longer needed for active use but needs to be retained for compliance or historical purposes. Data stored in this zone is typically rarely accessed and is optimized for long-term storage. An example use case for this zone could be storing customer data that is no longer needed for active use but needs to be retained for compliance reasons.

Each zone in a data lake has specific security and access controls, data retention policies, and data management processes, depending on the specific use case and data requirements.



*Figure 15.2: The different zones of a data lake*

As shown in the diagram above, the flow of data among the different zones in a data lake is a key aspect of the architecture. It typically follows the steps below:

1. Raw data is collected and stored in the raw zone in its original format.
2. The data is then moved to the landing zone, where it undergoes initial processing such as basic validation, quality checks, and removal of duplicates.
3. After the initial processing, the data is moved to the staging zone, where it undergoes transformation and integration with other data sources to prepare it for analysis.
4. The processed data is then moved to the analytics zone, where it is optimized for analysis and made available to data scientists, business analysts, and other users for exploration and reporting.
5. Based on specific business or operational requirements, subsets of the data may be created and stored in the data mart zone for specific analysis and reporting needs.

This flow ensures that the data is processed and stored to support its intended use and meets the required security, privacy, and compliance requirements. The flow also helps ensure data consistency and accuracy as the data moves through the different processing stages. If data is no longer needed for active use but needs to be retained for compliance or historical purposes, you can also create an archive zone for long-term storage.

These zones and their names should not be taken as dogma. Many folks use other labels for these zones and might use more or fewer zones. But these zones capture the general idea of what is required for a well-architected data lake.

An analogy that can help to understand how the various zones in a data lake work is how gold is mined, distributed, and sold. Gold is a scarce resource and is often found in small quantities combined with many other materials that have no value to the people mining it.

When it’s mined in industrial quantities, excavators dump dirt into a truck or a conveyor belt (this is the ingestion step in the landing zone and raw zone). This dirt goes through a cleansing process (analogous to the data quality step in the curated or staging zone). The gold is set aside, turned into ingots or bars, and transported for further processing (like the curation in the analytics zone). Finally, these gold bars may be melted down and turned into jewelry or industrial parts so individuals can use them for different purposes, which can relate to the data mart zone.

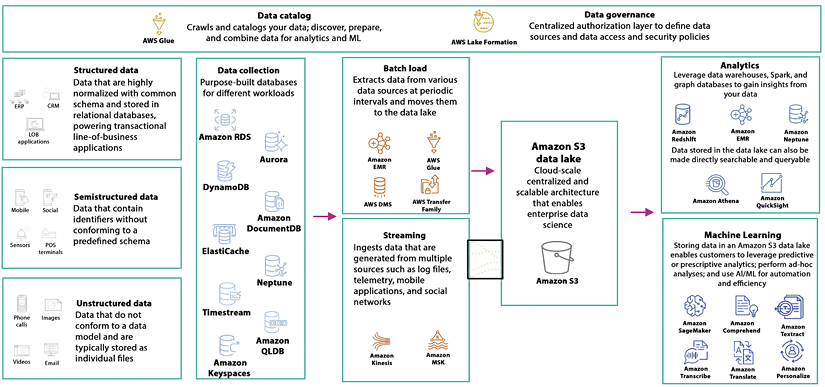
**Data lakes in AWS with Lake Formation**

Lake Formation is a fully managed data lake service provided by AWS that enables data engineers and analysts to build a secure data lake. Lake Formation provides an orchestration layer combining AWS services such as S3, RDS, EMR, and Glue to ingest and clean data with centralized fine-grain data security management.

Lake Formation lets you establish your data lake on Amazon S3 and begin incorporating readily accessible data. As you incorporate additional data sources, Lake Formation will scan those sources and transfer the data into your Amazon S3 data lake. Utilizing machine learning, Lake Formation will automatically structure the data into Amazon S3 partitions, convert it into more efficient formats for analytics, such as Apache Parquet and ORC, and eliminate duplicates and identify matching records to enhance the quality of your data.

It enables you to establish all necessary permissions for your data lake, which will be enforced across all services that access the data, such as Amazon Redshift, Amazon Athena, and Amazon EMR. This eliminates the need to reapply policies across multiple services and ensures consistent enforcement and adherence to those policies, streamlining compliance.

Lake Formation relies on AWS Glue behind the scenes, where Glue crawlers and connections aid in connecting to and identifying the raw data that requires ingestion. Glue jobs then generate the necessary code to transfer the data into the data lake. The Glue data catalog organizes the metadata, and Glue workflows link together crawlers and jobs, enabling the monitoring of individual work processes. The following diagram shows a comprehensive view of the AWS data lake:



*Figure 15.3: AWS data lake components*

As shown in the preceding diagram, below are the steps to create a data lake in AWS using Lake Formation:

1. Set up an AWS account and create an **Identity and Access Management** (**IAM**) role with the necessary permissions to access Lake Formation and other AWS services.
2. Launch the Lake Formation console and create a new data lake.
3. Ingest data into the data lake from a variety of sources such as Amazon S3, Amazon Redshift, Amazon RDS, and others.
4. Define the data catalog by creating tables, columns, and partitions and registering them with the Lake Formation data catalog.
5. Set up data access and security policies using IAM and Lake Formation to control who can access the data and what actions they can perform.
6. Perform transformations on the data in the data lake using AWS Glue or other tools.
7. Analyze the data using Amazon QuickSight, Amazon Athena, or other analytics tools.

Let’s say you want to create a data lake to store customer data from an e-commerce website. For that, you need to ingest customer data into the data lake from your e-commerce website’s database, which is stored in Amazon RDS. Then you can define the data catalog by creating tables for customer information, purchase history, and product information and registering them with the Lake Formation data catalog. You can perform transformations on the customer data in the data lake using AWS Glue to convert the data into a common format, remove duplicates, and perform data validation. Finally, you can analyze the customer data using Amazon QuickSight or Amazon Athena to create visualizations and reports on customer behavior and purchase patterns.

The steps and tools used may vary depending on the requirements and data sources. These steps are provided as a general guide and may need to be adapted based on the specific needs of your use case. You can refer AWS Lake formation guide for more details – <https://docs.aws.amazon.com/lake-formation/latest/dg/getting-started.html>.

From a data security perspective, the Lake Formation admin set up permission on the database, table, and column and granted granular row and column level permission to data lake users for data access. Lake Formation works in conjunction with AWS IAM to build security controls. Lake Formation enhances the search functionality by enabling text-based and faceted search across all metadata. It also adds attributes such as data owners and stewards as table properties, along with column properties and definitions such as data sensitivity level. Additionally, it offers audit logs for data lake auditing during data ingestion and cataloging while notifications are published to Amazon CloudWatch events and the console.

**Data lake best practices**

In this section, we will analyze best practices to improve the usability of your data lake implementation that will empower users to get their work done more efficiently and allow them to find what they need more quickly.

**Centralized data management**

Depending on your company culture, and regardless of how good your technology stack is, you might have a mindset roadblock among your ranks, where departments within the enterprise still have a tribal mentality and refuse to disseminate information outside of their domain.

For this reason, when implementing your data lake, it is critical to ensure that this mentality does not persist in the new environment. Establishing a well-architected enterprise data lake can go a long way toward breaking down these silos.

Centralized data management refers to the practice of storing all data in a single, centralized repository rather than in disparate locations or silos. This makes managing, accessing, and analyzing the data easier and eliminates the risk of data duplication and inconsistency.

A use case for centralized data management could be for a large e-commerce company that has customer data stored in multiple systems, such as an online store, a call center database, and a mobile app. The company could centralize this data in a data lake to improve data accuracy, ensure consistency, and provide a single source of truth for customer data.

The process of centralized data management in this scenario would involve extracting data from the various systems, cleaning and transforming the data to ensure consistency, and then storing it in a data lake. This centralized repository could be accessed by various departments within the company, such as marketing, sales, and customer service, to support their decision-making and improve customer experience.

By centralizing data, the company can improve data governance, minimize the risk of data duplication and inconsistency, and reduce the time and effort required to access and analyze the data. This ultimately leads to improved business outcomes and competitive advantage.

**Data governance**

One of the biggest challenges when implementing a data lake is the ability to fully trust the current data’s integrity, source, and lineage.

For the data in a lake to provide value, more is needed than just dumping the data into the lake. Raw data will not be valuable if it does not have structure and a connection to the business and is not cleansed and deduplicated. If data governance were built for the lake, users would be able to trust the data in the lake. Ungoverned data that does not possess data lineage is much less valuable and trustworthy than data with these qualities. Data lineage refers to the complete history of a data element, including its origin, transformations, movements, and dependencies. Ungoverned data increases regulatory and privacy compliance risks. Analysis and transformation of initially incorrect and incomplete data will result in incorrect and incomplete data, and most likely, any insights derived from this data will be inaccurate.

To fully trust and track the data in the lake, we need to provide context to the data by instituting policy-driven processes to enable the classification and identification of the ingested data. We need to put a data governance program in place for the data lake and leverage any existing data governance programs. Wherever possible, we should use existing data governance frameworks and councils to govern the data lake.

The enormous volume and variability of data in today’s organizations complicate the tagging and enrichment of data with the data’s origin, format, lineage, organization, classification, and ownership information. Most data is fluid and dynamic, and performing exploratory data analysis to understand it is often essential to determine its quality and significance. Data governance provides a systematic structure to gain an understanding of and confidence in your data assets. To set a foundation, let’s agree on a definition of data governance.

Data governance refers to the set of policies, processes, and roles that organizations establish to ensure their data’s quality, security, and availability. Data governance aims to improve data management and decision making by ensuring that data is accurate, consistent, secure, and accessible to those who need it.

A use case for data governance could be for a healthcare organization that collects patient data from various sources, such as electronic medical records, clinical trials, and wearable devices. The organization needs to ensure that this data is protected and used in a manner that complies with privacy regulations and patient consent.

The process of implementing data governance in this scenario would involve defining policies and processes for data collection, storage, use, and protection. This could include processes for data classification, data access controls, data quality control, and data auditing. The organization would also establish roles and responsibilities for data governance, such as data stewards, administrators, and security personnel.

By implementing data governance, the healthcare organization can ensure that patient data is protected and used responsibly, improve the quality and consistency of its data, and reduce the risk of data breaches and regulatory violations. This ultimately leads to improved patient trust and better decision-making based on accurate and secure data.

If the data’s integrity can be trusted, it can guide decisions and gain insights. Data governance is imperative, yet many enterprises need to value it more. The only thing worse than data that you know is inaccurate is data that you think is accurate, even though it’s inaccurate.

Here are a few business benefits of data lake governance:

* Data governance enables the identification of data ownership, which aids in understanding who has the answers if you have questions about the data. For example, were these numbers produced by the CFO or an external agency? Did the CFO approve them?
* Data governance facilitates the adoption of data definitions and standards that help to relate technical metadata to business terms. For example, we may have these technical metadata terms: f\_name, first\_name, and fn, but they all refer to the standardized business term “*First Name*.” They have been associated via a data governance process.
* Data governance aids in the remediation processes that need to be done for data by providing workflows and escalation procedures to report inaccuracies in data. For example, a data governance tool with workflows, such as Informatica MDM, Talent Data Stewardship, or Collibra, may be implemented to provide this escalation process. Has this quarter’s inventory been performed, validated, and approved by the appropriate parties?
* Data governance allows us to make assessments of the data’s usability for a given business domain, which minimizes the likelihood of errors and inconsistencies when creating reports and deriving insights. For example, how clean is the list of email addresses we received? If the quality is low, we can still use them, knowing we will get many bounce-backs. You can use tools like ZeroBounce or NeverBounce to validate emails.
* Data governance enables the lockdown of sensitive data and helps you implement controls on the authorized users of the data. This minimizes the possibility of data theft and the theft of trade secrets. For example, for any sensitive data, we should consistently implement a “need to know” policy and lock down access as much as possible. A “need to know” policy is a principle that restricts access to information only to those individuals who need it in order to perform their job responsibilities effectively.

**Data cataloging**

Data cataloging refers to the process of organizing, documenting, and storing metadata about data assets within an organization. The purpose of data cataloging is to provide a central repository of information about data assets, making it easier for organizations to discover, understand, and manage their data. For example, a company maintains a customer information database, including customer name, address, phone number, and order history. The data catalog for this database might include metadata about the database itself, such as its purpose, data sources, update frequency, data owners and stewards, and any known data quality issues. It would also include information about each data element, such as its definition, data type, and any transformations or calculations performed.

It would be best if you used metadata and data catalogs to improve discovery and facilitate reusability. Let’s list some of the metadata that is tracked by many successful implementations and that we might want to track in our implementation:

* **Access Control List (ACL)**: Access list for the resource (allow or, in rare cases, deny). For example, Joe, Mary, and Bill can access the inventory data. Bill can also modify the data. No one else has access.
* **Owner**: The responsible party for this resource. For example, Bill is the owner of the inventory data.
* **Date created**: The date the resource was created. For example, the inventory data was last updated on 12/20/2022.
* **Data source and lineage**: The origin and lineage path for the resource. In most cases, the lineage metadata should be included as part of the ingestion process in an automated manner. In rare cases where metadata is not included during ingestion, the lineage metadata information may be added manually. An example of this would be when files are brought into the data lake outside of the normal ingestion process. Users should be able to quickly determine where data came from and how it got to its current state. The provenance of a certain data point should be recorded to track its lineage.
* **Job name**: The name of the job that ingested and/or transformed the file.
* **Data quality**: For some of the data in the lake, data quality metrics will be applied to the data after the data is loaded, and the data quality score will be recorded in the metadata. The data in the lake is only sometimes perfectly clean, but there should be a mechanism to determine the data quality. This context will add transparency and confidence to the data in the lake. Users will confidently derive insights and create reports from the data lake with the assurance that the underlying data is trustworthy. For example, the metadata may be that a list of emails had a 7% bounce rate the last time it was used.
* **Format type**: With some file formats, it is not immediately apparent what the file format is. Having this information in the metadata can be helpful in some instances. For example, types may include JSON, XML, Parquet, Avro, and so on.
* **File structure**: In the case of JSON, XML, and similar semi-structured formats, a reference to a metadata definition can be helpful.
* **Approval and certification**: Once either automated or manual processes have validated a file, the associated metadata indicating this approval and certification will be appended to the metadata. Has the data been approved and/or certified by the appropriate parties?

Datasets should only be moved to the trusted data zone once this certification has been achieved. For example, inventory numbers may be approved by the finance department.

* **Business term mappings**: Any technical metadata items, such as tables and columns, always have a corresponding business term. For example, a table cryptically called SFDC\_ACCTS could have an associated corresponding business term, such as Authorized Accounts. This business term data doesn’t necessarily have to be embedded in the metadata. We could reference the location of the definition for the business term in the enterprise business glossary.
* **Personally Identifiable Information (PII), General Data Protection Regulation (GDPR), confidential, restricted, and other flags and labels**: Sometimes, we can determine whether data contains PII depending on where the data landed, but to increase compliance further, data should be tagged with the appropriate sensitivity labels.
* **Physical structure, redundancy checks, and job validation**: Data associated with data validation. For example, this could be the number of columns, rows, and so on.
* **Business purpose and reason**: A requirement to add data to a lake is that the data should be at least potentially useful. Minimum requirements should be laid out to ingest data into the lake, and the purpose of the data or a reference to the purpose can be added to the metadata.
* **Data domain and meaning**: It is only sometimes apparent what business terms and domains are associated with data. It is helpful to have this available.

There are a variety of ways that data governance metadata can be tracked. The recommended approaches are as follows:

* S3 metadata
* S3 tags
* An enhanced data catalog using AWS Glue

Data cataloging plays a crucial role in modern data management, enabling organizations to understand their data assets better, improve data quality, and support data-driven decision-making.

**Data quality control**

You need to validate and clean data before it is stored in the data lake to ensure data accuracy and completeness. Data quality control refers to the set of processes, techniques, and tools used to ensure that data is accurate, complete, consistent, and reliable. Data quality control aims to improve the quality of data used in decision-making, reducing the risk of errors and increasing trust in data.

For example, a retail company wants to ensure that the data it collects about its customers is accurate and up to date. The company might implement data quality control processes such as data profiling, data cleansing, and data standardization to achieve this. Data profiling involves analyzing the data to identify patterns and anomalies, while data cleansing involves correcting or removing inaccuracies and duplicates. Data standardization involves ensuring that data is consistently formatted and entered in a standardized manner. The following are some use cases for data quality control:

* **Decision-making**: By ensuring that data is accurate, complete, and consistent, data quality control enables organizations to make informed decisions based on reliable data.
* **Data integration**: Data quality control is critical for successful data integration, as it helps to ensure that data from different sources can be seamlessly combined without errors.
* **Customer relationship management**: High-quality data is critical for effective customer relationship management, as it allows enterprises to understand their customers better and provide them with personalized experiences.
* **Fraud detection**: Data quality control helps organizations to detect and prevent fraud by identifying and correcting errors and inconsistencies in data.
* **Compliance**: By ensuring that data is accurate and consistent, data quality control helps organizations to meet regulatory compliance requirements and avoid penalties.

Data quality control is a crucial aspect of modern data management for helping organizations ensure their data is reliable, and supporting informed decision-making.

**Data security**

You should implement security measures to ensure your data’s confidentiality, integrity, and availability. Data security best practices for data lakes can be divided into several categories, including:

* **Access control**: Control access to the data lake and its contents through **role-based access management** (**RBAC**) and multi-factor authentication. Use access control policies that specify who can access what data and what actions they can perform. For example, an online retailer can use access control to restrict access to its customer data to only those employees who need it to perform their job functions.
* **Data encryption**: Encrypt sensitive data at rest, in transit, and while being processed. The data lake can use encryption keys managed by **Key Management Service** (**KMS**) to encrypt data. For example, a healthcare organization can use data encryption to secure patient health information stored in the data lake.
* **Data masking**: Mask sensitive data elements within the data lake to prevent unauthorized access. Masking can be applied to columns, rows, or entire tables. For example, a financial organization can use data masking to protect sensitive customer information like account numbers and personal details within the data lake.
* **Data auditing**: Monitor and log all access to data in the data lake, including who accessed it, when, and what actions were performed. This helps to detect and respond to security incidents. For example, an energy company can use data auditing to monitor and log access to its data in the data lake to help detect and respond to security incidents.

Security is always a critical consideration when implementing search projects across the enterprise. AWS realized this early on. Like many other services in the AWS stack, many AWS offerings in the search space integrate seamlessly and easily with the **AWS Identity and Access Management** (**AWS IAM**) service. Having this integration does not mean we can push a button, and our search solution will be guaranteed secure. Similar to other integrations with IAM, we still have to ensure that our IAM policies match our business security policies. We have robust security to ensure that authorized users can only access sensitive data and that our company’s system administrators can only change these security settings.

As mentioned previously in this chapter, AWS Lake Formation is a service that makes it easier to build, secure, and manage data lakes. It also provides several security features to ensure that the data stored in the data lake is secure:

* **Access control**: Lake Formation provides fine-grained access control to data in the lake using AWS IAM policies. You can grant or revoke permissions to access the data lake and its contents, such as data tables and columns, based on user identities, such as AWS accounts or AWS **Single Sign-On** (**SSO**) identities.
* **Data encryption**: Lake Formation integrates with AWS KMS to provide encryption of data at rest and in transit. You can encrypt data in the lake using encryption keys managed by KMS to secure sensitive data.
* **VPC protection**: Lake Formation integrates with Amazon VPC to provide network-level security for data in the lake. You can secure access to the data lake by limiting access to specific VPC or IP addresses.
* **Audit logging**: Lake Formation provides audit logging for data access, modification, and deletion. You can use audit logs to monitor and track activities performed on the data lake and its contents.
* **Data masking**: Lake Formation provides data masking to protect sensitive data in the lake. You can mask sensitive data elements within the data lake to prevent unauthorized access, including columns, rows, or entire tables.
* **Data governance**: Lake Formation provides data governance features to manage and enforce data usage and protection policies. This includes classifying data based on its sensitivity, implementing retention policies, and enforcing data retention schedules.

These security features in Lake Formation help you secure data in the data lake and meet regulatory requirements for protecting sensitive data. They allow you to control access to the data lake and its contents, encrypt sensitive data, and monitor and log all activities performed on the data lake.

**Data ingestion**

You should automate data ingestion from various sources to ensure timely and consistent data loading. To ensure efficient, secure, and high-quality data ingestion, it’s important to follow best practices, including:

* **Data validation**: Validate the data before ingestion to ensure it’s complete, accurate, and consistent. This includes checking for missing values, incorrect data types, and out-of-range values. For example, an e-commerce company can use data validation to ensure that customer data is complete, accurate, and consistent before storing it in the data lake.
* **Data transformation**: Transform the data into a consistent format suitable for storage in the data lake. This includes standardizing data types, converting data into a common format, and removing duplicates. For example, a telecommunications company can use data transformation to convert customer call records into a common format suitable for storage in the data lake.
* **Data normalization**: Normalize the data to ensure it’s structured in a way that makes it easier to analyze. This includes defining common data definitions, data relationships, and data hierarchies. For example, a financial organization can use data normalization to ensure that financial data is structured in a way that makes it easier to analyze.
* **Data indexing**: Index the data to make it easier to search and retrieve. This includes creating metadata indices, full-text indices, and columnar indices. For example, an online retailer can use data indexing to make it easier to search and retrieve customer data stored in the data lake.
* **Data compression**: Compress the data to reduce its size and improve ingestion performance. This includes using compression algorithms like Gzip, Snappy, and LZ4.

For example, a media company can use data compression to reduce the size of video files stored in the data lake, improving ingestion performance.

* **Data partitioning**: Partition of the data to improve performance and scalability. This includes partitioning the data by date, time, location, or other relevant criteria. For example, a logistics company can use data partitioning to improve the performance and scalability of delivery data stored in the data lake, partitioning the data by delivery location.

Your data lake may store terabytes to petabytes of data. Let’s see data lake scalability best practices.

**Data lake scalability**

You should design your data lake to be scalable to accommodate future data volume, velocity, and variety growth. Data lake scalability refers to the ability of a data lake to handle increasing amounts of data and processing requirements over time. The scalability of a data lake is critical to ensure that it can support growing business needs and meet evolving data processing requirements.

Data partitioning divides data into smaller chunks, allows for parallel processing, and reduces the amount of data that needs to be processed at any given time, improving scalability. You can use distributed storage to store the data across multiple nodes in a distributed manner, allowing for more storage capacity and improved processing power, increasing scalability. Compressing data can reduce its size, improve scalability by reducing the amount of storage required, and improve processing time.

From an AWS perspective, you can use Amazon S3 as your storage, which helps you with a serverless computing model for data processing and allows for the automatic scaling of resources based on demand, improving scalability. You can use EMR and Glue to process data from S3 and store it back whenever needed. In that way, you will be decoupling storage and compute, which will help achieve scalability and reduce cost. Let’s look at best practices to reduce costs.

**Data lake cost optimization**

Using cost-effective solutions and optimizing data processing jobs, you should minimize storage and processing costs. Data lake costs can quickly add up, especially as the amount of data stored in the lake grows over time. To reduce and optimize the cost of a data lake, it’s important to follow best practices such as: compressing data to reduce its size, reducing storage costs and improving processing time; partitioning data into smaller chunks to allow for parallel processing; and reducing the amount of data that needs to be processed at any given time, improving processing performance and reducing costs.

Further, you can optimize the use of compute and storage resources, reduce costs by reducing resource waste, and maximize resource utilization and cost-effective storage options, such as Amazon S3 object storage or tiered storage, which can reduce storage costs while still providing adequate storage capacity. Implementing a serverless computing model with AWS Glue for data processing can reduce costs by allowing for the automatic scaling of resources based on demand, reducing the need for expensive dedicated resources.

**Monitoring a data lake for performance optimization**

You should monitor the performance of your data lake and optimize it to ensure that it meets your performance requirements. Data lake monitoring and performance optimization are critical components of a well-designed data lake architecture. These practices help to ensure that the data lake is functioning optimally and that any performance issues are quickly identified and resolved.

Continuously monitoring the performance of the data lake, including storage and compute utilization, can help identify and resolve performance issues before they become critical. You should define and track performance metrics, such as query response time and data processing time, which can help identify performance bottlenecks and inform optimization efforts.

Further, analyzing log data can provide insight into the performance of the data lake and help identify performance issues by regularly loading and testing the data lake can help identify performance bottlenecks and inform optimization efforts. Automatically scaling resources, such as compute and storage, based on usage patterns can improve performance and prevent performance issues.

**Flexible data processing in the data lake**

You should choose a data processing solution that can handle batch, real-time, and streaming data processing to accommodate a variety of use cases. Flexible data processing is an important aspect of data lake design, allowing for processing different types of data using various tools and techniques. Data lake should support a variety of data formats, including structured, semi-structured, and unstructured data, to allow for flexible data processing and flexible use of open-source tools such as Apache Spark and Apache Hive, which can allow for flexible data processing and minimize the cost of proprietary tools.

A data lake should support multiple processing engines, such as batch processing, stream processing, and real-time processing, to allow for flexible data processing. A data lake with a decoupled architecture, where the data lake is separated from the processing layer, can allow for flexible data processing and minimize the impact of changes to the processing layer on the data lake.

The data lake should be integrated with data analytics tools, such as business intelligence and machine learning tools, to allow for flexible data processing and analysis.

Now that we have gone over some of the best practices to implement a data lake, let’s now review some ways to measure the success of your data lake implementation.

**Key metrics in a data lake**

Now more than ever, digital transformation projects have tight deadlines and are forced to continue doing more with fewer resources. It is vital to demonstrate added value and results quickly.

Ensuring the success and longevity of a data lake implementation is crucial for a corporation, and effective communication of its value is essential. However, determining whether the implementation is adding value or not is often not a binary metric and requires a more granular analysis than a simple “green” or “red” project status.

The following list of metrics is provided as a starting point to help gauge the success of your data lake implementation. It is not intended to be an exhaustive list but rather a guide to generate metrics that are relevant to your specific implementation:

* **Size**: It’s important to monitor two metrics when evaluating a lake: the overall size of the lake and the size of its trusted zone. While the total size of the lake alone may not be significant or informative, the contents of the lake can range from valuable data to meaningless information. Regardless, this volume has a significant impact on your billing expenses. Implementing an archival or purging policy is an effective method for controlling the volume and minimizing costs. Your documents can either be transferred to a long-term storage location like Amazon S3 Glacier or eliminated altogether. Amazon S3 offers an easy approach to erasing files using life cycle policies. A larger size of the trusted zone indicates a better scenario. It represents the extent of clean data within the lake. Although you can store massive amounts of data in the raw data zone, it only serves its purpose when it undergoes transformation, cleaning, and governance.
* **Governability**: Measuring governability can be challenging, but it is crucial. It’s important to identify the critical data that requires governance and add a governance layer accordingly, as not all data needs to be governed. There are many opportunities to track governability. The criticality of data is key to establishing an efficient data governance program. Data on the annual financial report for the company is more critical than data on the times the ping pong club meets every week. Data deemed critical to track is dubbed a **Critical Data Element** (**CDE**).

To ensure efficient governability, you can assign CDEs and associate them with the lake’s data at the dataset level. Then, you can monitor the proportion of CDEs matched and resolved at the column level. Another approach is to keep track of the number of authorized CDEs against the total number of CDEs. Lastly, you can track the count of CDE modifications made after their approval.

* **Quality**: Data quality is not always perfect, but it should meet the standards for its intended domain. For instance, when using a dataset to generate financial reports for the current quarter, the accuracy of the numbers used is crucial. On the other hand, if the use case is to determine recipients for marketing emails, the data still needs to be reasonably clean, but a few invalid emails may not significantly impact the results.
* **Usage**: Usage tracking is crucial for maintaining an effective data lake. It is important to keep track of data ingestion rate, processing rate, error and failure rate, as well as individual components of the lake. These metrics can provide valuable insights into where to focus your efforts. If a particular section of the data lake needs more traffic, you may want to consider phasing it out. AWS offers an easy way to track usage metrics through SQL queries against AWS CloudTrail using Amazon Athena.
* **Variety**: It is useful to assess the variety aspect of the data lake and evaluate the capability of the system to handle various types of data sources. It should be able to accommodate different input types such as relational database management systems and NoSQL databases like DynamoDB, CRM application data, JSON, XML, emails, logs, and more. While the data ingested into the lake can be of diverse types, it is recommended to standardize the data format and storage type as much as possible. For example, you may decide to standardize on the Apache Parquet format or ORC format for all data stored in your Amazon S3 buckets. This allows users of the data lake to access it standardly. Achieving complete uniformity in the data lake might not always be practical or necessary. It is important to consider the context and purpose of the data before deciding on the level of homogenization required. For instance, it may not make sense to convert unstructured data into Parquet. Therefore, it is best to use this metric as a general guideline rather than a rigid rule.
* **Speed**: When it comes to speed, two measurements are valuable. Firstly, track the time it takes to update the trusted data zone from the start of the ingestion process. Secondly, track the time it takes for users to access the data. It’s not necessary to squeeze every millisecond out of the process, but it should be good enough. For example, if the nightly window to populate the data lake is four hours, and the process takes two hours, it might be acceptable. However, if you expect the input data to double, you may need to find ways to speed up the process to avoid hitting the limit.

Similarly, if user queries take a few seconds to populate reports, the performance might be acceptable, and optimizing the queries further might not be a priority.

* **Customer satisfaction**: Continuous tracking of customer satisfaction is crucial, as it is one of the most important metrics after security. The success of your data lake initiative depends on your users’ satisfaction, and a lack of users or unhappy users can lead to its failure. There are several ways to measure customer satisfaction, ranging from informal to formal approaches. The informal method involves periodically asking the project sponsor for feedback. However, a formal survey of the data lake users is recommended to obtain a more accurate metric. You can multiply the opinions of each survey participant by their usage level. For instance, if the lake receives low ratings from a few sporadic users but excellent ratings from hardcore users, it could imply that your data lake implementation has a steep learning curve, but users can become hyper-productive once they become familiar with it.
* **Security**: Security is a crucial aspect of data lake management, and compromising it is not an option. It is vital to ensure that the data lake is secure and that users only have access to their data to prevent any unauthorized access or data breaches. Even a single breach can lead to a significant loss of critical data, which can be used by competitors or other malicious entities. Another essential factor related to security is the storage of sensitive and **personally identifiable information** (**PII**). Mishandling PII data can result in severe penalties, including reputation damage, fines, and lost business opportunities. To mitigate this risk, AWS provides Amazon Macie, which can automatically scan your data lake and identify any PII data in your repositories, allowing you to take necessary actions to safeguard it. However, even with security metrics, there might be instances where good enough is acceptable. For example, banks and credit card issuers have a certain level of credit card fraud that they find acceptable. Eliminating credit card fraud might be a laudable goal, but it might not be achievable.

AWS provides a range of services for building and operating data lakes, making it an attractive platform for data lake implementations. Amazon S3 can be used as the primary data lake storage, providing unlimited, scalable, and durable storage for large amounts of data. AWS Glue can be used for data cataloging and ETL, providing a fully managed and scalable solution for these tasks. Amazon Athena can be used for interactive querying, providing a serverless and scalable solution for querying data in S3. You can use Amazon EMR for big data processing: Amazon EMR can be used for big data processing, providing a fully managed and scalable solution for processing large amounts of data.

The data lake can be integrated with other AWS services, such as Amazon Redshift for data warehousing and Amazon SageMaker for machine learning, to provide a complete and scalable data processing solution.

Now that you have learned about the various components of a data lake, and some best practices for managing and assessing data lakes, let’s take a look at some other evolved modern data architecture patterns.

**Lakehouse in AWS**

A lakehouse architecture is a modern data architecture that combines the best features of data lakes and data warehouses, while a data lake is a large, centralized repository that stores structured and unstructured data in its raw form. To have a structured view of data, you need to load data into the data warehouse. The lakehouse architecture combines a data lake with a data warehouse to provide a consolidated view of data.

The key difference between a lakehouse and a data lake is that a lakehouse architecture provides a structured view of the data in addition to the raw data stored in the data lake, while a data lake only provides the raw data. In a lakehouse architecture, the data lake acts as the primary source of raw data, and the data warehouse acts as a secondary source of structured data. This allows organizations to make better use of their data by providing a unified view of data while also preserving the scalability and flexibility of the data lake.

In comparison, a data lake provides a central repository for all types of data but does not provide a structured view of the data for analysis and reporting. Data must be prepared for analysis by cleaning, transforming, and enriching it, which can be time-consuming and require specialized skills.

Let’s take an example to understand the difference between a data lake and a lakehouse. A media company stores all of its raw video and audio content in a data lake. The data lake provides a central repository for the content, but the media company must perform additional processing and preparation to make the content usable for analysis and reporting. The same media company implements a lakehouse architecture in addition to the data lake. The data warehouse provides a structured view of the video and audio content, making it easier to analyze and report on the content. The company can use this structured data to gain insights into audience engagement and improve the quality of its content.

Here are the steps to implement a lakehouse architecture in AWS:

1. **Set up a data lake**: Start by setting up a data lake using Amazon S3 as the storage layer. This will provide a central repository for all types of data, including structured, semi-structured, and unstructured data.
2. **Define data ingestion pipelines**: Use tools like Amazon Kinesis, Amazon Glue, or AWS Data Pipeline to define data ingestion pipelines for your data sources. This will allow you to automatically collect and store data in the data lake as it becomes available.
3. **Create a data warehouse**: Use Amazon Redshift as your data warehouse. Amazon Redshift provides fast, managed data warehousing that can handle large amounts of data.
4. **Load data into the data warehouse**: Use Amazon Glue or AWS Data Pipeline to load the data from the data lake into the data warehouse. This will provide a structured view of the data for analysis and reporting.
5. **Perform data transformations**: Use Amazon Glue or AWS Data Pipeline to perform data transformations on the data in the data lake, if necessary. This will ensure that the data is clean, consistent, and ready for analysis.
6. **Analyze data**: Use Amazon Redshift to perform data analysis and reporting. Amazon Redshift provides fast and flexible data analysis capabilities, making it easy to perform complex data analysis. You can use Redshift Spectrum to join data residing in a data lake with Redshift data in a single query and get the desired result. You don’t need to load all data in the data warehouse for a query.
7. **Monitor and manage the lakehouse architecture**: Use Amazon CloudWatch and AWS Glue metrics to monitor and manage your lakehouse architecture. This will help ensure that the architecture performs optimally and that any issues are quickly identified and resolved.

These are the general steps to implement a lakehouse architecture in AWS. The exact implementation details will vary depending on the specific requirements of your organization and the data sources you are using.

**Data mesh in AWS**

While data lakes are a popular concept, they have their issues. While putting data in one place creates a single source of truth, you are also creating a single source of failure, violating standard architecture principles to build high availability.

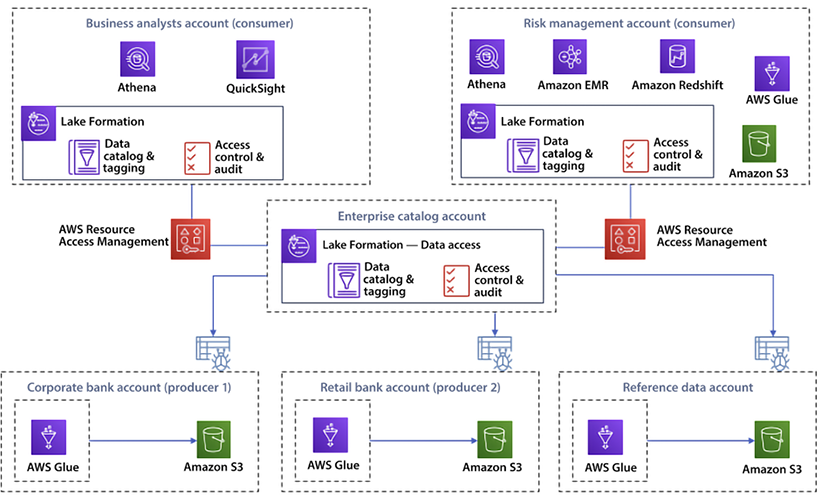
The other problem is that the data lake is maintained by a centralized team of data engineers who may need more domain knowledge to clean data. This results in back-and-forth communication with business users. Over time your data lake can become a data swamp.

The ultimate target of collecting data is to get business insight and retain business domain context while processing that data. What is the solution? That’s where data mesh comes into the picture. With data mesh, you can treat data as a product where the business team owns the data, and they expose it as a product that can be consumed by various other teams who need it in their account. It solves the problem of maintaining domain knowledge while providing the required isolation and scale for business. As data is accessed across accounts, you need centralized security governance.

Data mesh is an architectural pattern for managing data that emphasizes data ownership, consistency, and accessibility. The goal of data mesh is to provide a scalable and flexible data architecture that can support multiple domains, organizations, and products. In a data mesh architecture, data is treated as a first-class citizen and managed independently from applications and services. Data products are created to manage and govern data, providing a single source of truth for the data and its metadata. This makes it easier to manage data, reduces data silos, and promotes data reuse.

Data mesh also emphasizes the importance of data governance, providing clear ownership of data and clear processes for data management. This makes managing and maintaining data quality, security, and privacy easier. Organizations typically use a combination of data products, pipelines, APIs, and catalogs to implement a data mesh architecture. These tools and services are used to collect, store, and manage data, making it easier to access and use data across the organization. For example, an e-commerce company can use data mesh to manage customer, product, and sales data. This can include creating data products for customer profiles, product catalogs, and sales data, making it easier to manage and use this data across the organization.

The following diagram shows the data mesh architecture in AWS for a banking customer:



*Figure 15.4: Data mesh architecture in AWS*

As shown in the preceding diagram, the consumer account and consumer risk departments manage their data and expose it as a product consumed by the corporate account and retail account departments. Each of these departments operates in its account, and cross-account access is managed by a centralized enterprise account. The centralized account also manages the data catalog and tagging, and handles resource access management, which facilitates the producer and consumer of the data to talk to each other and consume data as per their need.

Here are the steps to implement data mesh in AWS:

1. **Define data domains**: The first step in implementing data mesh is to define the data domains in your organization. This involves identifying the different areas of the business that produce and use data and determining the relationships between these data domains.
2. **Create data products**: Once you have defined your data domains, the next step is to create data products. A data product is a self-contained unit of data that can be managed and governed independently from the rest of the organization. In AWS, you can create data products using AWS Glue, Amazon S3, and Amazon Redshift.
3. **Implement data pipelines**: To ensure that data is consistent and accurate, you need to implement data pipelines. A data pipeline is a series of steps that are used to extract, transform, and load data from various sources into a data lake or data warehouse. In AWS, you can implement data pipelines using AWS Glue, Amazon S3, and Amazon Redshift.
4. **Use data catalogs**: To make it easier to manage and access data, you need to use data catalogs. A data catalog is a metadata repository that provides a centralized location for storing and managing metadata about your data products. In AWS, you can use AWS Glue or Amazon Athena as your data catalog.
5. **Implement data APIs**: To make it easier to access data, you need to implement data APIs. A data API is a set of APIs that provides a programmatic interface for accessing data products. In AWS, you can implement data APIs using AWS Lambda and Amazon API Gateway.
6. **Ensure data security**: To ensure that data is secure, you need to implement data security measures. In AWS, you can use Amazon S3 bucket policies, IAM policies, and encryption to secure your data.

To implement data mesh in AWS, you need to define data domains, create data products, implement data pipelines, use data catalogs, implement data APIs, and ensure data security. These steps can help you build a scalable and flexible data architecture that can support multiple domains, organizations, and products.

**Choosing between a data lake, lakehouse, and data mesh architecture**

In a nutshell, data lake, lakehouse, and data mesh architectures are three different approaches to organizing and managing data in an organization.

A data lake is a centralized repository that allows you to store all your structured and unstructured data at any scale. A data lake provides the raw data and is often used for data warehousing, big data processing, and analytics. A lakehouse is a modern data architecture that combines the scale and flexibility of a data lake with the governance and security of a traditional data warehouse. A lakehouse provides raw and curated data, making it easier for data warehousing and analytics.

A data mesh organizes and manages data that prioritizes decentralized data ownership and encourages cross-functional collaboration. In a data mesh architecture, each business unit is responsible for its own data and shares data with others as needed, creating a network of data products. Here are some factors to consider when deciding between a data lake, data mesh, and lakehouse architecture:

* **Data governance**: Data lakes are great for storing raw data, but they can be challenging to manage, especially when it comes to data governance. Data mesh focuses on data governance and provides a way to manage and govern data across the organization. On the other hand, lakehouse combines the benefits of data lakes and data warehouses, providing a centralized repository for storing and managing data with a focus on data governance and performance.
* **Data processing**: Data lakes are ideal for big data processing, such as batch processing and real-time data processing. Data mesh focuses on enabling flexible data processing and creating data products that different parts of the organization can easily consume. On the other hand, lakehouse provides a unified platform for storing and processing data, making it easier to run ad-hoc and real-time queries.
* **Data access**: Data lakes can be difficult to access, especially for users who need to become more familiar with the underlying technologies. Data mesh focuses on providing easy access to data through data products and APIs. Lakehouse, on the other hand, provides a unified platform for accessing and querying data, making it easier for users to get the data they need.
* **Cost**: Data lakes can be more cost-effective than data warehouses, especially for large data sets. Data mesh can be more expensive due to the additional layer of management and governance. Lakehouse can also be more expensive, especially if you need to store and process large amounts of data.

Consider using a data lake if you need to store large amounts of raw data and process it using big data technologies. Consider using a data mesh if you need to manage and govern data across the organization. Consider using a lakehouse if you need a centralized repository for storing and managing data with a focus on data governance and performance.

**Summary**

In this chapter, you explored what a data lake is and how a data lake can help a large-scale organization. You learned about various data lake zones and looked at the components and characteristics of a successful data lake.

Further, you learned about building a data lake in AWS with AWS Lake Formation. You also learned about data mesh architecture, which connects multiple data lakes built across accounts. You also explored what can be done to optimize the architecture of a data lake. You then delved into the different metrics that can be tracked to keep control of your data lake. Finally, you learned about lakehouse architecture, and how to choose between data lake, lakehouse, and data mesh architectures.

In the next chapter, we will put together everything that we have learnt so far and see how to build an app in AWS.