Storing Big Data

Learning objectives

We will go over the following concepts:

- Understand the need to store data in systems other than databases;
- Revisit block and object storage; and
- Discuss the data lake.

Introduction

Let's consider the following questions:

- What do we do with data if we are not putting it into a database?
- Where are we consuming data from to start with?
- Where do we put data we want to use later on? Can we just save it on the desktop?

These are some of the questions we may have at the moment. Dealing with big data can be both daunting and confusing, especially with so many options available to store, retrieve, and process data.

Fortunately, data storage, transformation, and retrieval are some of the areas in big data which have seen massive growth in the last decade. This growth can be attributed to the significant reductions in hardware and software costs and the increasing demand for flexible data solutions at the enterprise and business unit levels.

In this train, we are going to break down the principles of data lakes and data storage to further our understanding of what services and solutions should be used to solve which business challenge.

Data storage not in databases

Sometimes data just does not fit into a table, a relational database, a non-relational database, or any other indexed database. In these instances, we may want to store the data in a system that is similar to a hard drive on a local computer. In this case, we are dealing with unstructured data, where the organisation of the underlying data is ambiguous. Examples are simple to find â€“ images, videos, binary files, blobs of text from logs, or even Microsoft Word documents. In the modern big data era, it obviously does not make sense to have data stored on a local disk drive and uploading and downloading as we need it. Rather, we need a solution that is highly scalable, easily accessible, and always available to our applications.

Data storage solutions in the cloud

As we are aware, multiple cloud solutions exist to store data in an unstructured format. Let's revise block and object storage.

Block storage

Block storage is a storage method in which each storage volume acts as a separate hard drive. Block storage breaks data up into blocks and then stores the blocks as separate pieces, each with a unique identifier. Each block can then be placed on the drives in the most efficient location. Each storage volume can be treated as a disk drive and can be controlled by an external server operating system. Thus, we can mount as we would a physical disk.

Block storage has the following advantages:

- It is possible to take snapshots of the whole drive for backup and disaster recovery purposes;
- It enables us to resize block storage as our needs change;
- It can easily be detached and moved between machines;
- Since it is based on a familiar paradigm, it can integrate easily with almost any system while being supported by almost every programming language;
- Access and permissions protocols are well established and understood for filesystems; and
- Block storage provides low I/O latency.

Some disadvantages are:

- Storage can only be tied to one server;
- Blocks and file systems have limited metadata information about the blobs that are stored on them;
- We can only access block storage through a running server; and
- Block storage needs more hands-on work and setup than object storage.

Use cases:

- Due to low latencies, block storage can be used to power databases;
- Running mission-critical workloads;
- Used as physical disks for RAID volumes;
- For applications that require server-side processing or the deployment of containers; and
- When attaching volumes to a virtual machine in the cloud.

Examples:

- Azure Premium Storage: Provides disks attached to Azure Virtual Machines for I/O intensive workloads.
- AWS Elastic Block Storage (EBS): Provides raw block storage that we can attach to Elastic Cloud Compute (EC2) instances.

Object storage

Object storage is the go-to storage of unstructured blobs of data and metadata in the cloud. Instead of breaking files into blocks, object storage allows us to store whole objects within a single repository. Object storage can then be accessed through an HTTP API and does not have to be attached to a cluster to be accessed. The objects can be anything â€“ images, logs, CSVs, binary files, and more. The object store guarantees that the data will not be lost and can be replicated across different data centres.

It becomes obvious why this type of storage has been quite popular in the cloud. It lets us store nearly anything, and we're able to store using a simple HTTP POST request and retrieve using a GET request. This also allows us to expose files publicly without having to maintain a serving web server. Not just that, but most providers charge only for storage, which is typically almost negligible in cost. Object storage also has an almost infinite scaling capability, being able to scale into the petabyte range and beyond.

Other than the aforementioned, the key **advantages** of using object storage are:

- Object storage has a simple HTTP API, with clients for most programming languages;
- Strictly pay-per-use, offering storage at a very low cost;
- We can expose the object store to the public;
- Some object stores have a built-in content delivery network. *(Content delivery networks are networks that cache data around the world, ensuring the data are always close to our users to guarantee very low latency, for example, if we have a website serving pictures, we want a picture close to the user so that the network latency for getting the picture to the user is not too high.)*;
- Some stores enable versioning, archiving older versions of the objects;
- Object stores can scale out quite easily, quickly, and most importantly, can be configured to scale automatically without any maintenance overhead to the developer; and
- We can store all data and associated metadata and versions thereof in one location. This provides a very rich metadata space, for example, for video files, we can store metadata relating to where the video was shot, the camera used, and attributes down to the detail of every frame of the video.

However, there are some **disadvantages**:

- There is an increase in latency compared to what we would require when working with transactional databases;
- We need to access the whole blob at a time, and any transactions on the blob will result in the creation of a new object, even if we want to just alter a part thereof. This becomes difficult when we deal with blobs larger than 1GB and is mostly solved by using object storage alongside distributed computing systems, which typically output one or more files per processing unit meaning that files will not get as large; and
- Operating systems cannot mount object stores, but rather need specific software or programming languages to interact with the object store.

Use cases:

- Storing unstructured data like sound, images, video, and documents;
- Storing backup files, database dumps, logs, and metrics;
- Used as a big data store â€“ storing datasets that have a high velocity and volume; and
- Archiving data.

Examples:

- Amazon S3
  - Amazon Simple Storage Service is an object store with high scalability, availability, security, and performance. Data in S3 are organised into buckets, which is a logical grouping of objects that are defined by the user.
- Azure Blob Storage
  - Similarly, Azure Blob Storage allows the storage of objects in the cloud and is designed to be scalable, durable, available, and secure. Here, data are organised into containers, similar to buckets in Amazon S3.

Object storage vs. block storage vs. file storage

Below is a comparison of object, block, and file storage in a concise table that highlights the benefits and advantages of each of the three technologies.

Storage solution	Description	Interface	Cost	Performance	Use case	Scale
Block storage	Data stored in fixed-size block in a rigid arrangement â€“ ideal for enterprise databases	Operating system	Relatively expensive	Highly performant	OS, Database	Low scalability
File storage	Data stored as files in hierarchically nested folders â€“ ideal for active documents	User	Expensive	Performant	Web content	Medium scalability
Object storage	Data stored as objects in scalable buckets â€“ ideal for big data, analytics, and archiving	Program (API)	Relatively inexpensive	Performant	Images, PDFs, videos	Highly scalable

When we have lots of data - the data lake

Figure 1: Let's talk about lakes. Image by:Dirk von Loen-Wagner

What is a data lake?

A data lake is a centralised data repository that allows us to store, govern, transform, discover, share, and centralise all of our data at any scale.

Being able to leverage the data generated in-house is fundamental to the growth of companies, and companies that do this tend to grow exponentially. While most institutions end up generating massive amounts of data, these datasets end up being siloed across the organisation and traditionally there is poor integration between silos. By integrating these datasets and making them available to a plethora of analysts, it would be possible to explore new business opportunities, respond to changes in real-time, reduce operational costs, and apply machine learning to anticipate change.

Previously, the way to centralise the data and make it available to the wider business would be through a data warehouse. However, when working with big data, data cleaning and wrangling occur at the same time as asking questions, searching for anomalies in the dataset, detecting insights, and making inferences. This means that we cannot provide data scientists with cleaned, structured, and normalised data as present in a data warehouse, but rather need a solution where we can provide them with uncleaned and raw data.

This is where the data lake comes in â€“ a single point of data storage. It is important to note that the data lake is a concept of design rather than a specific technology, and there are many incarnations of it in the various cloud platforms. The main idea behind the data lake is having a low-cost data repository storing data from varied sources and varied levels of processing in a single location. Having such an architectural approach breaks down the data silos that exist by centralising them at the core. As mentioned, this allows the advance of machine learning, data science, data analytics, and streaming use cases.

As with other data solutions and technologies, a data lake needs active maintenance to ensure it does not turn into a *data swamp*. These maintenance activities include:

- Cataloguing the data being ingested with sufficient metadata;
- Applying a data cleaning strategy;
- Ensuring semantic consistency; and
- Applying sufficient governance, access control, and security.

Some characteristics of the data lake include:

- Collecting and storing all data types (structured, semi-structured, and unstructured) at any scale, at a fraction of other storage costs;
- Securing data in a central repository;
- Being able to search the data within the central repository;
- Practices and processes are in place to govern, move, transform, and catalogue the data;
- Able to ingest new data sources simply and apply new transformations easily;
- Has access points into the data lake, allowing analytics and advanced intelligence to be performed at scale;
- Systems that feed and use the data lake are decoupled to enable faster time to market;
- Cost-consciousness; and
- Allows AI/ML-enabled outcomes.

The best data lakes are built using a holistic inclusion of architecture, security, network, storage, and data governance.

Generic data lakes

Let's have a look at a very generic data lake architecture, all the basic required components, and how all the different areas within the data lake connect. The figure below shows the most basic data lake, with data sources, an ingestion layer, an operation layer, data storage, and an analytics layer.

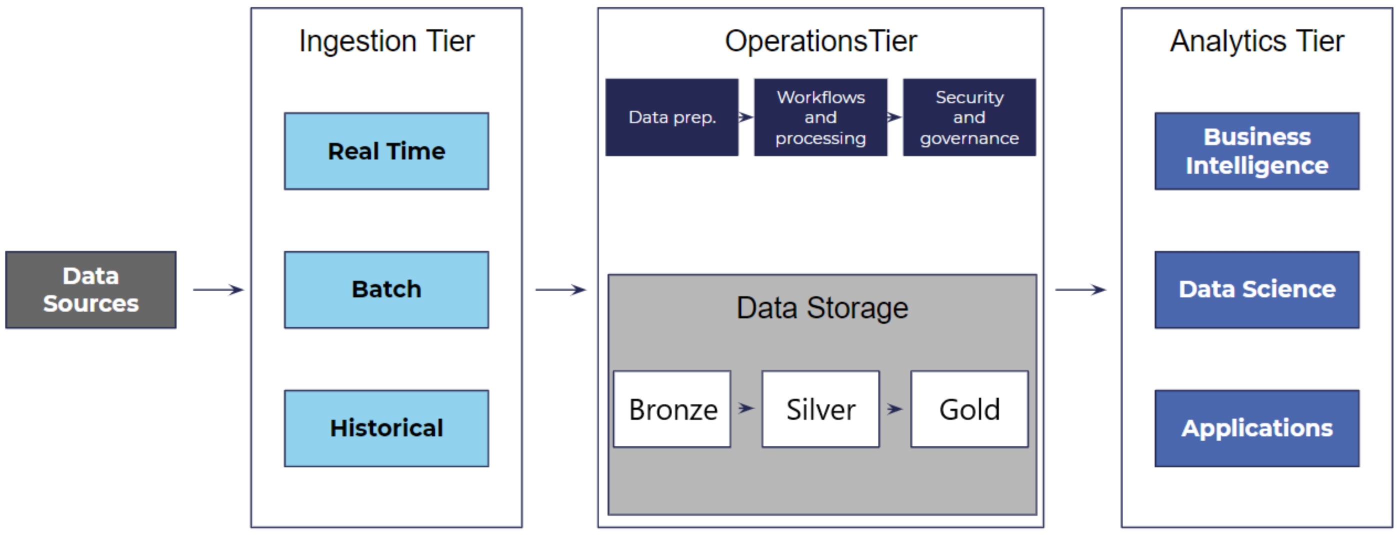

Figure 2: An architectural representation of a generic data lake.

To set up such a generic data lake requires several design considerations: architecture, storage, ingestion, data processing, consumption, data governance, and security.

Architecture

Most modern data lakes are built using a microservice architecture, enabling multiple pieces of the data lake to be added or removed whenever a need arises. Some of the ideal practices for a data lake are:

- **Decoupled applications:** At the base of the microservice paradigm, a failure of one process should not hold up another process. Or, in the data lake, a failure of one data processing pipeline should not stop another data pipeline from running.
- **Remove single points of failure:** Single points of failure are why applications crash, and they refer to components that will no longer function in the face of a single unexpected event. The solution is to ensure that no component can be brought down by a single event. In other words, multiple layers of redundancy are introduced which allows bugs to be fixed while the application continues to run. This should be on the software and hardware level, for example, replicating the storage layer across the region to ensure availability even when the data centre may be experiencing downtime.
- **Agile:** Microservice architectures are commonly built using agile approaches, getting to a minimum viable product (MVP) in the shortest amount of time possible. Working with a diverse and cross-functional team allows us to have everyone needed to deliver the MVP in as little time as possible while also allowing for teams to adapt to change and changes in requirements very quickly.
- **Decoupled compute and storage:** Having compute and storage decoupled allows us to scale each independently. This means that we can have massive amounts of data and we will not underutilise our compute during times where requests are low (for example, at night). Or it enables us to serve requests when demand is high, but the total data and thus storage requirements are low. Formally, decoupled compute and storage leads to optimal resource provisioning.
- **Auditing and logging:** For a team that rapidly develops decoupled services and services running in the cloud, it is essential to log events to find and resolve bugs as soon as they come in. This is also important for data governance â€“ always being aware of the flow and usage of data, and ensuring there is no data breach or unauthorised access.

Storage

Storage is the essence of data lakes. It is where data gets ingested to and is consumed from. To operate effectively, a data lake should provide the following:

- **Scalability:** Since the data lake acts as a central repository that could potentially be used across the entire enterprise, the data lake should be able to provide data as the number of users increases, but also save cost when the users do not access the data. This is only possible knowing that our data lake can scale with demand and usage.
- **High data availability:** Data lakes are only effective when they are used across the entire organisation. However, when we create a data lake as the data backbone of our organisation, we need to ensure that it is always functioning correctly and always available. Technically, it should have both timeliness and high availability. Strategies include replication across geographical regions, which also reduces latency for anyone trying to access the data.
- **Data durability:** Data should never get lost once it's arrived in the data lake, seeing as it's the central data repository. This is achieved through geographical replication.
- **Security:** When we expose or leak data, we lose reputation and, by implication, we lose business. Thus, we need to ensure the total security of our data assets, including:
    - Encryption at rest and in transit;
    - Data centre security;
    - Immutable data; and
    - Data masking and compliance with local laws and data protection regulations.

- **Governance and auditing:** Since the start of the information age, people have become more and more aware of the risks associated with sharing personal data, and governing bodies and governments have started to legislate regulations on data protection. This means that local regulations around data immutability, personally identifiable information, auditing, and data usage should be possible within our data lakes.
- **Storage flexibility:** We are ingesting data from theoretically an infinite amount of datasets, which can even be in a diverse number of formats. Thus, we have to be able to store data in any possible format and retrieve the data in that exact format at low latencies. *Data lakes are typically built using object storage, which stores any object and can provide low latencies.*
- **Storage file size and format:** While we *can* store anything in a data lake, some formats and object sizes are more desirable than others. When working with a data lake, data scientists typically use tools that employ massively parallel processing, and thus also leverage a distributed file system storage methodology. This allows for better distribution of files when read in and processed. File types include:
  - Parquet;
  - ORC; and
  - Avro and others, each of which has features such as being row orientated or column orientated.

Ingestion

Ingestion is the process of taking data from source systems (for example, APIs, IoT devices, databases, or the web) and bringing the data into the data lake in its original format.

- **Data sources:** Where are the data coming from? This will wholly depend on the type of business and ranges from sensors, payroll applications, logs from applications, weblogs, videos, and so on.
- **Ingestion:**
  - Ingestion can be performed through traditional batch processes, ingesting data at a given frequency. Batch sources can be pulls from a database, calls to an API, or retrieval from an FTP site.
  - When initially setting up the data lake or a new data source, we may want to ingest a historical load, which can be massive and may require a process separate from the normal batch process. This can range from extended ingestion, ingestion via a route other than the batch process, or even ingesting using a physical device.
  - Finally, streaming allows us to ingest real-time data into our data lake. This is part of what makes the data lake so powerful â€“ consolidating and providing a live view of the real world. There are also multiple methods for stream ingestion, but the process typically happens through a message broker.

At the ingestion stage, transformations are not typically applied and raw data are archived in a manner that makes it easy to go back in time to a specific record. A common pattern is to structure the folder path as follows: source/object/year/month/day/file.ext. This allows us to go back to a specific ingestion year, month, day, and even a specific point in time if multiple files are ingested every day (with files being named according to DateTime â€“ filename_2021_01_01_1400000.json).

Data processing

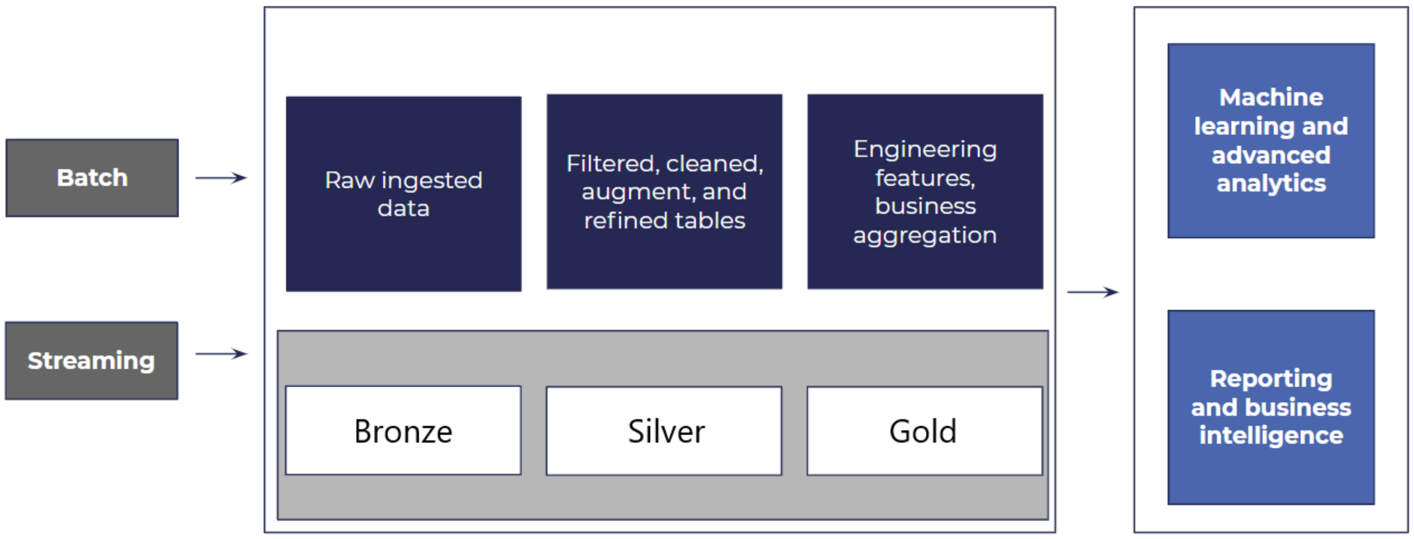
Batch
Streaming
Raw ingested data
Filtered, cleaned, augment, and refined tables
Engineering features, business aggregation
Bronze
Silver
Gold
Machine learning and advanced analytics
Reporting and business intelligence

Figure 3: Possible stages of data within a data lake.

Within data analytics, cloud data storage, and specifically the data lake, data can be classified as being bronze, silver, or gold standard. Bronze data are typically ingested directly as raw data. Silver data are transformed in some way, including through cleaning, translating, refining, augmenting, and filtering. Gold data are ready to be directly consumed by analytic or reporting tools, typically after feature engineering or other business aggregations have been performed.

Data processing is responsible for taking raw (bronze) data and transforming it into a more structured format usable by a wider audience and users of the data lake (silver or gold standard). The first step in setting up a data lake is to create a *data catalogue* (which should be continuously maintained).

> A data catalogue is a collection of metadata, combined with data management and search tools. This allows us to find the specific data we require, serving as an inventory for all the available data.

Data processing can include some data cleansing â€“ transforming data types between formats, handling missing values, filtering outliers, and even some aggregations. It is also advisable to generate multiple layers of separate data during this process, to ensure that teams can pick up the data at a stage which is most advantageous for them. Examples of this may include taking a list of JSON files returned from an API, consolidating and cleansing the data, and then persisting the data to a single file that has a structured schema and format. This is then a silver dataset.

Workflows and processing then run more complex processes, including analytical algorithms, imputations, feature extractions, and other aggregations. This will generate an augmented dataset that can be used for advanced analytics â€“ a gold dataset.

A data science team may want full control and require a bronze dataset. A reporting and business analyst team may, in turn, require a silver dataset, and finally, a dashboard end-user may require a gold dataset. All of the processing can be done in a suite of tools:

- **Big data framework:** As mentioned, when working with big data we prefer to do processing using distributed frameworks. The detailed inner workings of distributed frameworks are described here, but in short, files are split up and computations are applied to each 'partition' of the dataset separately, greatly speeding up processing and increasing throughput. Many frameworks implement this process, including Apache Spark, Apache Hadoop, and some commercial offerings. To make life more convenient, cloud providers also supply managed versions of these services, for example, Azure Databricks and AWS EMR.

- **ETL tools:** ETL tools allow us to pipe the entire data flow together, which may include ingesting the data from the source, transforming it from bronze, and finally performing machine learning with it as it comes in. Again, there are various implementations â€“ Apache Airflow, Amazon Glue, and Azure Data Factory â€“ each with its advantages and disadvantages.

  We require certain features from these frameworks and tools:

- **Scalability:** Load may increase at any point in time. If our source suddenly provides data at half the previous interval, we need to be able to increase the velocity with which we process data, possibly by scaling out. Otherwise being able to scale up may allow us to do more complex computations as our business understanding of the domain increases (or we hire a brand new data scientist). Most vendors offer auto-scaling clusters.

- **Managed infrastructure:** Depending on our business domain, we may have data flowing or streaming 24 hours a day and require computations to run continuously. Other domains may only receive data once a day, and having a cluster that is continuously running will be a great wastage. Cloud providers solve this through auto-terminating clusters. Other innovations since the inception of the cloud era are the advent of managed resources, where we are spared from all the effort of setting up a Hadoop or Spark cluster, and we can spend all our time building pipelines and data flows instead of bashing our heads against a figurative cluster configuration wall.

Consumption

Most of the value realised from the data lake is where people actually use the data that is held in it. Since the aim of a data lake is to centralise all the data sources within the enterprise, we expect the users of the data lake to have varying use cases, domains, and skillsets for accessing the data. Some of the features of the data lake that allow a diverse set of users to access it are:

- **Data dictionaries and catalogues:** A place to look up data and where it can be found.
- **Immutable raw data:** The assurance that the data we access in one instance will be exactly the same the next time it is accessed.
- **A central data repository:** All the data are available from a central location and only requires a central access point.
- **Layering data processing:** Creating a bronze -> silver -> gold schema for processing data, allowing access to the data on all three levels.
- **Schema-on-read:** Only defining the schema as the data are read, meaning data can be stored without knowing the schema and retrieval is flexible since the schema is defined at retrieval time.

Let's get into what some of the use cases for the data lake look like.

- **Data warehousing:** Business and data analysts have the requirement of a data warehouse, a central store that contains a LOT of data to run complex SQL queries on to return complex analytics as output. Such a data warehouse will act as the central source of truth across the business, and as such, a lot of thought has to be put into its construction and the data model that will be used. These data warehouses are typically in the petabyte-scale, with warehouses featuring distributed storage and computation, decoupled compute and storage, huge compression ratios, zone mapping, and columnar storage to improve the performance for returning data in a highly performant manner. Some of the cloud implementations are AWS Redshift, Google BigQuery, and Azure Synapse.

- **Interactive querying:** Data analysts might also require access to data sources in the data lake that are not within the data warehouse or that are similar to the data used by data scientists. Data analysts typically do not have the skillsets to access the data through APIs or programming language SDKs, rather they require being able to access the data using SQL. By allowing interactive querying (through SQL) of the underlying data of the data lake, we unlock its full potential to a very wide user base by providing a friendly, logical schema for underlying data stored in a wide range of data formats. Some tools that allow this are Apache Hive, Apache Presto, Databricks SQL Analytics, Amazon Athena, and Azure Synapse.

- **Data science and machine learning:** Data scientists want to be in control of data all the way from source â€“ cleaning, transforming, creating features, and building machine learning models. Data lakes allow them access to massive, raw, enterprise-wide datasets, but also to data that are in various stages of processing and cleansing. Data scientists have a large toolkit, making use of Python, R, Scala, and other languages, as well as a range of libraries like Tensorflow, PyTorch, and Spark, among others. Fortunately, since data lakes are built on top of simple object storage in cloud platforms, they can be connected to a range of tools for the data lake, including Azure Databricks, AWS EMR, or even simple virtual machines.

- **AI engineering and automation:** Within enterprises, we may want to build AI solutions that come prebuilt, such as chatbots, speech-to-text systems, or image processing and characterisation. When integrated, these services can pick up audio streams, website-submitted images, or messages â€“ directly from our data lake. This reduces the operational overhead and latency and centralises the storage of the raw data and predictions for these services.

Data governance

For most organisations data is a central commodity, and if we lose or leak data, we lose business and revenue. Further, when centralising and consuming data from across an enterprise, we are bound to work with data that contain personally identifiable information (PII). PII data have to be treated differently due to their sensitive nature which could put individuals at risk. In response, various privacy laws and regulations have been put in place by governments globally.

To achieve data protection for the above two reasons, data governance is central to constructing a data lake. Data governance is the management of availability, usability, integrity, access, and security of data in the organisation. It has many facets, but let's consider the three primary ones.

- **Data catalogue:** One of the first things to construct when building a data lake is a data catalogue or data dictionary. A data dictionary is a central source of truth for all datasets present in the data lake, along with information and metadata for all datasets. It ensures that users can quickly locate the data they are looking for as well as the state of the particular dataset. Put simply, it's a searchable inventory of all datasets with the fitness of the dataset for business users. Data catalogues can be constructed using data warehouse technologies such as Amazon Athena or big data tools like Apache Hive or Apache Spark.

- **Data quality:** More than knowing what data are available in the data lake, it is important to know that the available data are of sufficient quality for a particular use case. Moreover, we need to be confident that when new data comes in, it is still of the same quality. Generally, we define six dimensions along which we evaluate data quality:

    - Accuracy;
    - Completeness;
    - Timeliness;
    - Validity;
    - Uniqueness; and
    - Consistency.

  Through tracking data quality across these dimensions, we can assure business users that the quality of the data is not changing significantly with time.

- **Governance and regulations:** Governing bodies have put regulations in place to ensure data that are classified as PII are treated in a way that does not expose individuals to any risk. These regulations include GDPR, HIPAA, and ISO 27701. Failure to comply with these regulations can lead to fines or imprisonment, making it pertinent to ensure that our data lakes have sufficient protection measures in place. Cloud providers have various services simplifying this for us, including dynamic data masking, AI-driven PII data identifiers, and key management and access control services.

Security

For a large data-driven business â€“ data is money! As such, controlling and ensuring the security of our data lake means securing our revenue. Security should be considered from day one and should be closely integrated with our enterprise IT security policy. Good security will always be implemented in a layered manner, with penetration of the first layer requiring the toppling of several more before the attacker is actually within reach of any data. Let's look at some of the top considerations for data lake security.

- **Data security:** As part of the layered security approach, data should not be*ready-to-read* once someone gets access to it. This is effected on two levels: encryption at rest and encryption in transit. All data should be encrypted at rest, and all cloud providers provide a simple toggle to turn on encryption at rest for most available storage layers. Additional security can be achieved by using a key management service along with specific encryption algorithms, the specific method depending on the cloud platform in question. Data in transit is data moving over the network, APIs, or the internet and needs to be secured using TLS/SSL transmissions with certificates.

- **Network security:** Another layer in the layers of security. When working in the cloud, isolating the services to private networks allows us to only expose a single endpoint or very limited IP address range. Implementation of private networks will depend on the cloud provider (for example, Virtual Private Cloud in AWS or Azure Virtual Network) but should allow us to restrict access using user groups as well as other approaches such as Access Controlled Lists (ACLs). Having private networks isolate us from the public creates a network demilitarised zone (DMZ). Apart from isolating our services, we can also create network firewalls that control access and monitor web traffic. Combined, these services secure access and limit the number of users or agents that can access our data lake.

- **Access control:** Since data lakes contain data from across the organisation, it is very important to ensure that the correct people access the correct parts of the data â€“ we would not want an intern to access payroll information. To this end, it is best to integrate with existing enterprise-grade authentication services such as Azure Active Directory. Cloud providers allow direct integration between enterprise authentication services and cloud platforms. Cloud providers also allow fine-grained access control to users, for example, AWS IAM S3 bucket policies or Azure Access Control Lists (ACLs) and Role-Based Access Control (RBAC). The RBAC and ACL access control mechanisms offer different levels of granularity for providing access to files and resources. You can find some of the differences detailed in this resource.

- **Application security:** Applications are typically the most exposed part of the organisation, facing the outside world directly. Network firewalls do not have a mechanism for detecting and deferring attacks on applications. Instead, we need an application firewall, which can help protect applications against the most common web exploits and threads. Application firewalls control how traffic reaches the application, enabling us to detect attacks such as cross-site scripting and SQL injections. A decoupled infrastructure further helps to defer attacks, separating data from the application.

Conclusion

We revisited object and block storage and had a look at why we would want to place data in stores other than databases. Then we started looking at data lakes, how the need for them arose, how they work, typical data lake best practices, and finally, we fully described the optimal data lake architecture. You should now have a thorough understanding of the inner workings of block storage, object storage, and the design principles of data lakes.

References

- How big is big data?
- AWS data lakes
- Design patterns for data lakes
- Overview of storage implications