**DATA PLATFORMS, DATA STORES AND SECURITY**

**ARCHITECTING THE DATA PLATFORM**

A data platform architecture. A layer represents functional components that perform a specific set of tasks in the data platform. The layers that we’re going to explore, include:

Data Ingestion or Data Collection Layer, Data Storage and Integration Layer, Data Processing Layer, and Analysis and User Interface Layer.

**The Data Collection Layer** is responsible for connecting to the source systems and bringing the data from these systems into the data platform. This layer performs the following key tasks: Connect to data sources. Transfer data from these data sources to the data platform in streaming, batch, or both modes. Maintain information about the data collected in the metadata repository. For example, how much data was ingested in a batch, data source, and other descriptive information. Google Cloud DataFlow, IBM Streams, IBM Streaming Analytics on Cloud, Amazon Kinesis, and Apache Kafka are some of the tools used for data ingestion, supporting both batch and streaming modes. Once data is ingested, it needs to be stored and integrated.

**The Storage and Integration layer** in a data platform needs to: Store data for processing and long-term use. Transform and merge extracted data, either logically or physically. Make data available for processing in both streaming and batch modes. The storage layer needs to be reliable, scalable, high-performing, and also cost-efficient. IBM DB2, Microsoft SQL Server, MySQL, Oracle Database, and PostgreSQL are some of the popular relational databases. Cloud-based relational databases, also referred to as Database-as-a-Service, have gained great popularity over the recent years. Such as IBM DB2 on Cloud, Amazon Relational Database Service (RDS), and Google Cloud SQL, and SQL Azure. In the NoSQL, or non-relational database systems on the cloud, we have IBM Cloudant, Redis, MongoDB, Cassandra, and Neo4J. Tools for integration include IBM’s Cloud Pak for Data and Cloud Pak for Integration; Talend’s Data Fabric and Open Studio. Open-source tools such as Dell Boomi and SnapLogic are also very popular integration tools. There are a number of vendors offering cloud-based Integration Platform as a Service (or iPaaS). For example, Adeptia Integration Suite, Google Cloud's Cooperation 534, IBM's Application Integration Suite on Cloud, and Informatica's Integration Cloud. Once the data has been ingested, stored, and integrated, it needs to be processed.

**The processing layer** should be able to: Read data in batch or streaming modes from storage and apply transformations. Support popular querying tools and programming languages. Scale to meet the processing demands of a growing dataset. Provide **a way for analysts and data scientists to work with data in the data platform**. Some of the transformation tasks that occur in this layer include: **Structuring,** essentially, actions that change the form and schema of the data. This change may be as simple as changing the order of fields within a record or dataset or as complex as combining fields into complex structures using joins and unions. **Normalization**, which focuses on cleaning the database of unused data and reducing redundancy and inconsistency. **Denormalization**, which combines data from multiple tables into a single table so that it can be queried more efficiently for reporting and analysis. And **Data Cleaning,** which fixes irregularities in data to provide credible data for downstream applications and uses. There are a host of tools available for performing these transformations on data, selected based on the **data size**, **structure**, and **specific capabilities of the tool**. Such as **spreadsheets, OpenRefine, Google DataPrep, Watson Studio Refinery, and Trifacta Wrangler.** Python and R also offer several libraries and packages that are explicitly created for processing data. **It’s important to note that storage and processing may not always be performed in separate layers.** For example, in relational databases, storage and processing can occur in the same layer, while in Big Data systems, data can be first stored in the Hadoop File Distribution System, or HDFS, and then processed in a data processing engine like Spark. And, **the data processing layer can also precede the data storage layer, where transformations are applied before the data is loaded, or stored, in the database.**

**The Analysis and User Interface Layer** delivers processed data to data consumers. Data consumers can include: Business Intelligence Analysts and business stakeholders who consume this data through interactive visual representations, such as dashboards and analytical reports. Data Scientists and Data Analytics that further process this data for specific use cases. Other applications and services that may need this data as input for further use. The Analysis and UI Layer needs to support: **Querying tools and programming languages**. For example, SQL for querying relational databases and SQL-like querying tools for non-relational databases, such as CQL for Cassandra, Programming languages such as Python, R, and Java, **APIs** that can be used to run reports on data for both online and offline processing. **APIs** that can consume data from the storage in real-time for use in other applications and services. **Dashboarding and Business Intelligence applications**. For example, IBM Cognos Analytics, Tableau, Jupyter Notebooks, Python and R libraries, and Microsoft Power BI.

Overlaying the Data Ingestion, Data Storage and Integration, and Data Processing layers is the **Data Pipeline layer** with the Extract, Transform, and Load tools. This layer is responsible for implementing and maintaining a continuously flowing data pipeline. There are a number of data pipeline solutions available, most popular among them being Apache Airflow and DataFlow.

**Factors for selecting and Designing Data Stores.**

**A data store, or data repository**, is a general term used to refer to data that has been collected, organized, and isolated so that it can be used for business operations or mined for reporting and data analysis.

A repository can be **a database, data warehouse, data mart, big data store, or a data lake**. A well-designed data repository is essential for building a system that is scalable and capable of performing during high workloads.

The primary considerations for designing a data store are:

* The type of data you want to store.
* Volume of data.
* Intended use of data
* Storage considerations.
* Privacy, security, and governance needs of your organization.

**A database** is essentially a collection of data designed for the input, storage, search and retrieval, and modification of data. Depending on the type of data, databases can be categorized in two primary ways – **relational and non-relational.** Relational databases, or RDBMSes, are best used for structured data, which has a well-defined schema and can be organized into a **tabular format**. Non-relational databases, or NoSQL, are best for semi-structured and unstructured data, that is, **schema-less and free-form data**. Non-relational databases, based on the type of data and how you want to query the data, are of **four different types**—**key-value, document, column, and graph-based**. If you're looking to **run complex search queries and multi-operation transactions**, for example, a document-based database may not be the best option for you. Like you would not opt for a graph-based database if you need to process **high-volume transactions** because graph-based databases are not optimized for large-volume analytics queries.

**The volume**, or scale, of data. When you require to store large volumes of raw data in its native format, straight from its source, **a data lake** would be the appropriate choice for you. With a data lake, you can store both relational and non-relational data at scale without defining the data's structure and schema. Or when you're dealing with Big Data, which is data that is not only **high-volume but also high-velocity, of diverse types**, and needs **distributed processing for fast analytics**, then a big data repository would be an option you would explore. Big data stores split large files across multiple computers allowing parallel access to them. Computations run in parallel on each node where data is stored.

**How you intend to use the data you are collecting is also an important consideration for the choice and design of a data store**. The number of transactions, frequency of updates, type of operations performed on the data, response time, and backup and recovery requirements all need to be provisioned for in the design of a data store.

1. **Transactional systems**, that is systems used for capturing high-volume transactions, need to be designed for high-speed read, write, and update operations.
2. **Analytical systems**, on the other hand, need complex queries to be applied to large amounts of historical data aggregated from transactional systems. They need faster response times to complex queries. Schema design, indexing, and partitioning strategies have a big role to play in performance of systems based on how data is getting used.

The intended use of data also drives **scalability** as a design consideration. **The scalability** of a data store is its capability to handle growth in the amount of data, workloads, and users. Normalization of the database is another important consideration at the design stage. **Normalization** is the process of efficiently organizing data in a database. Done right, it helps in the optimal use of storage space, makes database maintenance easier, and provides faster access to data. Normalization is important for systems that **handle transactional data**. But for systems designed for handling analytical queries, normalization can lead to performance issues.

**The perspective of storage**. These are **Performance, Availability, Integrity, and Recoverability of Data**. PAIR

1. **Performance** parameters include throughput and latency. That is, the rate at which information can be read from and written to the storage and the time it takes to access a specific location in storage.
2. **Availability** - Your storage solution must enable you to access your data when you need it, without exception. There should be no downtime.
3. **Integrity** - Your data must be safe from corruption, loss, and outside attack.
4. **Recoverability** - Your storage solution should ensure that you can recover your data in the event of failures and natural disasters.

**A secure data strategy** is a layered approach. It includes **access control, multizone encryption, data management, and monitoring systems**.

**Regulations** such as GDPR, CCPA, and HIPAA restrict the ownership, use, and management of personal and sensitive data. Data needs to be made available through controlled data flow and data management by using multiple data protection techniques. This is an important part of a data store design. Strategies for data privacy, security, and governance regulations need to be a of a data store's design from the start. Done at a later stage it results in patchwork.

**SECURITY**

Three key components to creating an effective strategy for information security in general these are popularly referred to as the **CIA** triad the **C** in this triad stands for **confidentiality** through controlling unauthorized access **I** for integrity through validating that your resources are trustworthy and have not been tampered with and **A** is for availability by ensuring authorized users have access to resources when they need it the

CIA triad is applicable to all facets of security be it infrastructure network application or data security.

The four different facets or levels of security are **Physical Infrastructure**, **Network security, Application security, and Data security.**

**Physical infrastructure** security a key component of security for an it system is the security of the physical infrastructure and facilities that house the system in the case of cloud computing this extends to the **cloud service providers infrastructure and facilities** here are some of the measures that are taken to ensure physical infrastructure security access to the perimeter of the facility based on **authentication and round-the-clock surveillance for entry and exit points** **multiple power feeds** from independent utility providers with dedicated generators and **ups battery backup heating and cooling mechanisms** for managing the temperature and humidity levels in the facility and by factoring in environmental threats before considering the location of the facility for example infrastructure facilities are never housed in **flood plains in areas prone to earthquakes** the infrastructure is housed in an earthquake-resistant structure multi-level lightning protection and earthing systems are also installed in such facilities

**Network security**: network security is vital to keep interconnected systems and data safe network security solutions include **firewalls** to prevent unauthorized access to private networks that are connected to the internet, **network access control** to ensure endpoint security by allowing only authorized devices to connect to the network, for example, a corporate network may not allow devices with outdated service packs to connect to their network segmentation to create silos or virtual local area networks within a network so that you can segregate your assets into individual silos based on the level of security required for different assets **security protocols** to ensure attackers cannot tap into data while it is in transit and **intrusion detection and intrusion prevention** systems that inspect incoming traffic for intrusion attempts and vulnerabilities.

**Application security:** application security is critical for keeping customer data private and ensuring applications are fast and responsive security needs to be built into the foundation of an application in order to prevent other applications and services from introducing vulnerabilities you can make your application safe by following security engineering practices such as **threat modeling** to identify relative weaknesses and attack patterns related to the application, **secure design** that mitigates risks secure coding guides and practices that prevent vulnerabilities and **security testing** to fix problems before the application is deployed and to validate that it is free from known security issues.

**Data security** data is either at rest in storage or in transit between systems applications services and workloads be it at rest or in motion data needs to be protected one of the primary controls for data security is to enable access to data through a **system of authentication and authorization,** authentication systems verify that you are who you say you are and they accomplish this using **passwords tokens biometrics or a combination of these**. Authorization ensures that users access information based on their **roles and the privileges assigned to their roles**. Data at rest includes **files objects and storage** this type of data is stored physically such as in a database, data warehouse, tapes off-site, backups or on mobile devices. Organizations can use **encryption** to fight threats to their data at rest encrypting data protects information from disclosure even if that information is lost or intercepted data that is moving from one place to another such as when it is transmitted over the internet is referred to **as data in transit or data in motion,** encryption methods such as **https ssl and tls** are often used to protect data in motion. The vulnerabilities data can be exposed to and the security features that help protect data end to end through its life cycle it is vital to **proactively monitor track and react to security violations in time** and for that you need end-to-end visibility and integration of security processes and tools throughout the enterprise, **security monitoring and intelligence systems** create a complete audit history for triage and compliance purposes and provide reports and alerts that help enterprises react to security violations in time.

**VIEWPOINTS: IMPORTANCE OF DATA SECURITY**

If your data is not secured, nothing else matters. It can result in dire consequences such as company-ending.

Data security cannot be patched in, It has to be thought of every step of the way.

Give the least privilege needed in order to do their job

**Summary and Highlights**

In this lesson, you have learned:

The architecture of a data platform can be seen as a set of layers, or functional components, each one performing a set of specific tasks. These layers include:

Data Ingestion or Data Collection Layer, responsible for bringing data from source systems into the data platform.

Data Storage and Integration Layer, responsible for storing and merging extracted data.

Data Processing Layer, responsible for validating, transforming, and applying business rules to data.

Analysis and User Interface Layer, responsible for delivering processed data to data consumers.

Data Pipeline Layer, responsible for implementing and maintaining a continuously flowing data pipeline.

A well-designed data repository is essential for building a system that is scalable and capable of performing during high workloads.

The choice or design of a data store is influenced by the type and volume of data that needs to be stored, the intended use of data, and storage considerations. The privacy, security, and governance needs of your organization also influence this choice.

The CIA, or Confidentiality, Integrity, and Availability triad are three key components of an effective strategy for information security. The CIA triad is applicable to all facets of security, be it infrastructure, network, application, or data security.

**Data Collection and Data Wrangling**

Sources of data can be databases, the web, sensor data, data exchanges and several other sources leveraged for specific data needs.

Importing data into different types of data repositories:

**SQL, or Structured Query Language**, is a querying language used for extracting information from relational databases. SQL offers simple commands to specify what is to be retrieved from the database, the table from which it needs to be extracted, grouping records with matching values, dictating the sequence in which the query results are displayed, and limiting the number of results that can be returned by the query, amongst a host of other features and functionalities. **Non-relational databases** can be queried using SQL or SQL-like query tools. Some non-relational databases come with their own querying tools such as CQL for Cassandra and GraphQL for Neo4J.

**Application Programming Interfaces (or APIs)** are also popularly used for extracting data from a variety of data sources. APIs are invoked from applications that require the data and access an end-point containing the data. End-points can include databases, web services, and data marketplaces. APIs are also used for data validation. For example, a data analyst may utilize an API to validate postal addresses and zip codes.

**Web scraping**, also known as screen scraping or web harvesting, is used for downloading specific data from web pages based on defined parameters. Among other things, web scraping is used to extract data such as text, contact information, images, videos, podcasts, and product items from a web property.

**RSS** feeds are another source typically used for capturing updated data from online forums

and news sites where data is refreshed on an ongoing basis.

**Data streams** are a popular source for aggregating constant streams of data flowing from sources such as instruments, IoT devices and applications, and GPS data from cars.

**Data Exchange** platforms allow the exchange of data between data providers and data consumers. Data Exchanges have a set of well-defined exchange standards, protocols, and formats relevant for exchanging data. These platforms not only facilitate the exchange of data, they also ensure that security and governance are maintained. They provide data licensing workflows, de-identification and protection of personal information, legal frameworks, and a quarantined analytics environment. Examples of popular data exchange platforms include AWS Data Exchange, Crunchbase, Lotame, and Snowflake. Numerous other data sources can be tapped into for specific data needs. For marketing trends and ad spending, for example, research firms like Forrester and Business Insider are known to provide reliable data. Research and advisory firms such as Gartner and Forrester are widely trusted sources for strategic and operational guidance. Similarly, there are many trusted names in the areas of user behavior data, mobile and web usage, market surveys, and demographic studies. Data that has been identified and gathered from the various data sources now needs to be loaded or imported into a data repository before it can be wrangled, mined, and analyzed.

Data streams and feeds are also used for extracting data from social media sites and interactive platforms.

**The importing process** involves combining data from different sources to provide a combined

view and a single interface using which you can query and manipulate the data.

Depending on the data type, the volume of data, and the type of destination repository,

you may need varying tools and methods.

**Specific data repositories are optimized for certain types of data**. Relational databases store structured data with a well-defined schema. If you’re using a relational database as the destination system, you will only be able to store structured data, such as data from OLTP systems, spreadsheets, online forms, sensors, network and web logs. **Structured data can also be stored in NoSQL**. Semi-structured data is data that has some organizational properties but not a rigid schema, such as, data from emails, XML, zipped files, binary executables, and TCP/IP protocols. Semi-structured can be stored in NoSQL clusters. XML and JSON are commonly used for storing and exchanging semi-structured data. JSON is also the preferred data type for web services. Unstructured data is data that does not have a structure and cannot be organized into a schema, such as data from web pages, social media feeds, images, videos, documents, media logs, and surveys. NoSQL databases and Data Lakes provide a good option to store and manipulate large volumes of unstructured data. Data lakes can accommodate all data types and schema.

Tools such as Talend and Informatica, and programming languages such as Python and R, and their libraries, are widely used for importing data.

**Data Wrangling**

**Data wrangling, also known as data munging**, is an iterative process that involves data

exploration, transformation, validation, and making data available for a credible and meaningful

analysis.

Key transformation of raw data.

the first transformation task:

1. **Structuring**. This task includes actions that change the form and schema of your data. The incoming data can be in varied formats. You might, for example, have some data coming from a relational database and some data from Web APIs. In order to merge them, you will need to change the form or schema of your data. This change may be as simple as changing the order of fields within a record or dataset or as complex as combining fields into complex structures. **Joins and Unions** are the most common structural transformations used to combine data from one or more tables.

**Joins combine columns**. When two tables are joined together, columns from the first source table are combined with columns from the second source table—in the same row. So, each row in the resultant table contains columns from both tables. **Unions combine rows**. Rows of data from the first source table are combined with rows of data from the second source table into a single table. Each row in the resultant table is from one source table or another.

1. **Normalization and Denormalization of data**. Normalization focuses on cleaning the database of unused data and reducing redundancy and inconsistency. Data coming from transactional systems, for example, where a number of insert, update, and delete operations are performed on an ongoing basis, are highly normalized. Denormalization is used to combine data from multiple tables into a single table so that it can be queried faster. For example, normalized data coming from transactional systems is typically denormalized before running queries for reporting and analysis.
2. **Cleaning. Cleaning** tasks are actions that fix irregularities in data in order to produce a credible and accurate analysis. The first step in the data-cleaning workflow is to detect the different types of issues and errors that your dataset may have. You can use **scripts and tools** that allow you to define specific rules and constraints and validate your data against these rules and constraints. You can also use **data profiling and data visualization tools for inspection**. **Data profiling** helps you to inspect the source data to understand the structure, content, and interrelationships in your data. It uncovers anomalies and data quality issues. For example, blank or null values, duplicate data, or whether the value of a field falls within the expected range. Visualizing the data using statistical methods can help you to spot outliers. For example, plotting the average income in a demographic dataset can help you spot outliers. That brings us to the actual cleaning of the data. The techniques you apply for cleaning your dataset will depend on your use case and the type of issues you encounter.

Let’s look at some of the more common data issues. Let’s start with missing values. **Missing values** are very important to deal with as they can cause unexpected or biased results. You can choose to filter out the records with missing values or find a way to source that information in case it is intrinsic to your use case. For example, missing age data from a demographics study. A third option is a method known as **imputation,** which calculates the missing value based on statistical values. Your decision on the course of action you choose needs to be anchored in what’s best for your use case. You may also come across **duplicate data**, data points that are repeated in your dataset. These need to be removed. Another type of issue you may encounter is that of **irrelevant data**. Data that does not fit within the context of your use case can be considered irrelevant data. For example, if you are analyzing data about the general health of a segment of the population, their contact numbers may not be relevant for you. Cleaning can involve **data type conversion** as well. This is needed to ensure that values in a field are stored as the data type of that field—for example, numbers stored as numerical data type or date stored as a date data type. You may also need to clean your data in order **to standardize it**. For example, for strings, you may want all values to be in lower case. Similarly, date formats and units of measurement need to be standardized. Then there are **syntax errors**. For example, white spaces, or extra spaces at the beginning or end of a string is a syntax error that needs to be rectified. This can also include **fixing typos or format**, for example, the state name being entered as a full form such as New York versus an abbreviated form such as NY in some records. Data can also have **outliers**, or values that are vastly different from other observations in the dataset. Outliers may, or may not, be incorrect. For example, when an age field in a voters database has the value 5, you know it is incorrect data and needs to be corrected. Now let’s consider a group of people where the annual income is in the range of one hundred thousand to two hundred thousand dollars—except for that one person who earns a million dollars a year. While this data point is not incorrect, it is an outlier, and needs to be looked at.

**Tools for Data Wrangling**

Excel Power Query / Spreadsheets and Add-ins OpenRefine Google DataPrep Watson Studio Refinery Trifacta Wrangler Python R.

**Spreadsheets**. Spreadsheets such as Microsoft Excel and Google Sheets have a host of features and in-built formulae that can help you identify issues, clean, and transform data. Add-ins are available that allow you to import data from several different types of sources and clean and transform data as needed—such as Microsoft Power Query for Excel and Google Sheets Query function for Google Sheets.

**OpenRefine** is an open-source tool that allows you to import and export data in a wide variety

of formats, such as TSV, CSV, XLS, XML, and JSON. Using OpenRefine, you can clean data, transform it from one format to another, and extend data with web services and external data.

OpenRefine is easy to learn and easy to use. It offers menu-based operations, which means you don’t need to memorize commands or syntax.

**Google DataPrep** is an intelligent cloud data service that allows you to visually explore, clean, and prepare both structured and unstructured data for analysis. It is a fully managed service, which means you don’t need to install or manage the software or the infrastructure. DataPrep is extremely easy to use. With every action that you take, you get suggestions on what your ideal next step should be. DataPrep can automatically detect schemas, data types, and anomalies.

**Watson StudioIBM Data Refinery**, available via IBM Watson Studio or Cloud Pak for Data, allows you to discover, cleanse, and transform data with built-in operations. It transforms large amounts of raw data into consumable, quality information that’s ready for analytics. Data Refinery offers the flexibility of exploring data residing in a spectrum of data sources. It detects data types and classifications automatically and also enforces applicable data governance policies automatically.

**Trifacta Wrangler** is an interactive cloud-based service for cleaning and transforming data. It takes messy, real-world data and cleans and rearranges it into data tables, which can then be exported to Excel, Tableau, and R. It is known for its **collaboration features**, allowing multiple team members to work simultaneously.

**Python** has a huge library and set of packages that offer powerful data manipulation capabilities. Let’s look at a few of these libraries and packages. **Jupyter Notebook** is an open-source web application widely used for data cleaning and transformation, statistical modeling, also data visualization. **Numpy, or Numerical Python**, is the most basic package that Python offers. It is fast, versatile, interoperable, and easy to use. It provides support for large, multi-dimensional arrays and matrices, and high-level mathematical functions to operate on these arrays. **Pandas** is designed for fast and easy data analysis operations. It allows complex operations such as merging, joining, and transforming huge chunks of data, performed using simple, single-line commands. Using Pandas, you can prevent common errors that result from misaligned data coming in from different sources.

**R**, too, offers a series of libraries and packages that are explicitly created for wrangling messy data—such as **Dplyr, Data.table, and Jsonlite**. Using these libraries, you can investigate, manipulate, and analyze data. **Dplyr** is a powerful library for data wrangling. It has a precise and straightforward syntax. **Data.table** helps to aggregate large data sets quickly. **Jsonlite** is a robust JSON parsing tool, great for interacting with web APIs.

**Querying Data, Performance Tuning and Troubleshooting**

**Querying and Analyzing Data**

We will learn about some basic querying techniques for analyzing data in a database.

These include **counting and aggregating your data set**, **identifying extreme values**, **slicing**

**data, sorting data, filtering patterns, and grouping data.**

NoSQL databases, for example, have their own SQL-like query languages, such as **Cassandra CQL and Cypher for Neo4J**. Also APIs, that can be used to query data.

In SQL, for example, you can use **the COUNT() function** to count the number of rows, or records, you have in your data set.

**DISTINCT() function** to isolate the unique car dealers. If you combine **COUNT() with DISTINCT()**, you get a count of the total number of unique, or distinct.

**Aggregation functions** also help to provide an overview of the data set from different perspectives. Such as, using **the SUM() function** to calculate the sum of a numeric column in a table. Using **the AVG() function** to calculate the average value of a numeric column. And using the **STDDEV() function** to measure how spread out the numbers are in a data set.

You would use the **SLICE() function** if you want to retrieve data for customers who live in a certain area or have purchased their car from dealers in a specific area or those who spent between one thousand and two thousand dollars for their car.

**Sorting data**, which helps to arrange data in a meaningful order, making it easier to understand and analyze. For example, if you want to see if car sales go up around festival time, you can sort the data set on the date of purchase to see if a large number of sales transactions took place around that time. And you can do this using the ORDER BY() function. The ORDER BY() function also allows you to sort a data column in ascending or descending order.

**Technique is filtering patterns**, which allows us to perform partial matches of data values. Unlike an EQUAL TO operator, that returns records in which a data value matches a certain value, the LIKE operator helps you to specify a pattern to return records that match a data value partially.

**Grouping data**. Grouping of data is performed using the GROUP BY statement.

For example, you want to find out the total amount spent by customers, pincode-wise.

We can use SUM() and GROUP BY to find out the total amount spent by customers grouped

by their pincode.

**Performance Tuning and Troubleshooting**

One of the key responsibilities of a data engineer is to monitor and optimize systems

and data flows for performance and availability.

Data pipelines encompass the journey of data from source to destination through multiple systems, applications, and processes. A data pipeline typically runs with a combination of complex tools and can face several different types of performance threats, such as:

**Scalability**, in the face of increasing data sets and **workloads Application failures,** **Scheduled jobs not starting as per schedule**, waiting on dependencies, or some of the tasks not running in the correct sequence or not running at all and **Tool incompatibilities**, since data pipelines consist of a variety of tools handling different tasks.

When you need to benchmark or evaluate performance, you need to have a performance metrics. In the case of data pipelines, the performance metrics would include: **Latency**, that is the time it takes for a service to fulfill a request. **Failures**, which is the rate at which a service fails. **Resource utilization and utilization patterns And Traffic, or number of user requests received in a given period**.

So, what's the best way to go about troubleshooting performance issues in a data pipeline?

It depends on the issue, but generically speaking, these are some of the steps you will probably

take. **The first step** would be to collect as much information about the incident as possible,

most importantly, to ascertain if the observed behavior is actually an issue.

The issue could have been reported through an alerting system, by a user, or flagged during a maintenance check. **The next step** could be to check if you're working with all the right versions of software and source codes. And in case of any recent deployments, you could check what has changed and investigate if there could be a connection. You would also check the logs and metrics early on in your troubleshooting process to isolate whether an issue is related to infrastructure, data, software, or a combination of these. Error messages in logs and the network load, memory and CPU utilization at the time of the failure can help with this. But if the logs don't help you isolate the issue, then you'll possibly need to reproduce the issue in a test environment. This can be an iterative and time-consuming job.

Let's first understand the performance metrics that needs to be monitored in a database. These include: **System outages, Capacity utilization, Application slowdown Performance of queries And Conflicting activities and queries being executed based on multiple users giving requests at the same time, and batch activities causing resource constraints**.

Let's look at some of the best practices for database optimization such as **capacity planning** and **database design** activities such as indexing, partitioning, and normalization. **Capacity planning** includes the process of determining the optimal hardware and software resources required for performance, even as the load on the system continues to fluctuate on a day-to-day basis. Capacity planning also factors in future growth requirements. **Database indexing** helps to quickly locate data without searching each row in a database. It minimizes the number of times a disk needs to be accessed when a query is processed. **Database partitioning** is another feature that provides multiple performance benefits. It's a process whereby very large tables are divided into smaller, individual tables. Queries run faster because they're accessing a smaller part of the data. Partitioning also improves data manageability. **Database normalization** is a design technique to reduce inconsistencies arising out of data redundancy and anomalies arising out of update, delete, and insert operations on databases.

**Monitoring and alerting systems** help us collect quantitative data about our systems and applications in real time. In the data engineering lifecycle, these systems give us visibility into the performance of our data pipelines, platforms, databases, applications, tools, queries, scheduled jobs, and more.

**Database monitoring** tools take frequent snapshots of the performance indicators of a database. This can help you track when and how a problem really started to occur.

This can help you isolate and get to the root of the issue more efficiently.

**Application performance management tools** help us measure and monitor the performance of applications. They do this by tracking request response time and error messages.

These tools also track the amount of resources being utilized by each process, which helps

in the proactive allocation of resources to improve application performance.

Tools for monitoring the performance of queries gather statistics about query throughput,

execution performance, resource utilization and utilization patterns for better planning

and allocation of resources.

Data pipelines typically have long-running processes that take significant time to complete.

Which also means that the cost of failure is higher when errors are observed or flagged

at the end of a process.

**Job-level runtime monitoring** break up a job into a series of logical steps which are monitored

for completion and time to completion.

Monitoring the amount of data being processed through a data pipeline helps to assess if

size of the workload could be slowing down a system.

Another thing you can do to help you operate your systems at optimal levels is to run maintenance routines.

These routines can be: Time-based. That is, they could be planned as scheduled activities at pre-fixed time intervals. Or Condition-based, which means they are performed when there is a specific issue or when a decrease in performance has been noted or flagged.

**Governance and Compliance**

Data Governance is a collection of principles, practices, and processes to maintain the security,

privacy, and integrity of data through its lifecycle.

**Data that needs Governance:**

**personal and sensitive data**.

* That is data that can be traced back to an individual,
* can be used to identify an individual,
* or contains information that can be used to cause harm to the person,

For example data about race, sexual orientation, or genetic information. **The General Data Protection Regulation, or GDPR**, is a regulation specific to the European Union. It protects the personal data and privacy of EU citizens for transactions that occur within EU member states. In the United States, individual states have formulated their own regulations. For instance, **California created the California Consumer Privacy Act, or CCPA,** to better protect customer data.

There are also industry-specific regulations that govern the collection and use of sensitive and personal data:

* In Healthcare, **Health Insurance Portability and Accountability Act**, **HIPAA** privacy rules govern the collection and disclosure of protected health information.
* In Retail, the **Payment Card Industry Data Security Standard** **PCI DSS** standards govern credit card data.
* In Finance, **Sarbanes Oxley SOX** regulations govern the handling and reporting of financial information.

**Compliance** covers the processes and procedures through which an organization adheres to regulations and conducts its operations in a legal and ethical manner.

**Organizations need to establish controls and checks in order to comply with regulations. And maintain a verifiable audit trail that can establish their adherence to these regulations**.

Consequences of non-compliance with the standards can be severe. They can result in financial penalties, mar public perception, and result in loss of trust amongst clients and partners. It’s important to mention that compliance is not a one-time activity—it is an ongoing process requiring a blend of people, process, and technology that continues to evolve.

A typical data lifecycle in an organization includes the following stages In **the Data Acquisition stage**:

* You need to establish What data needs to be collected and the contracts and consent that give you a legal basis for procuring this data.
* The intended use of this data, published as a privacy policy and communicated internally and with individuals whose data is being collected.
* The amount of data you need to meet your defined purposes.

**In the Data Processing stage:**

* You will be required to establish The details of how you are going to process personal data.
* And your legal basis for the processing of personal data, such as a contract or consent.

**In the Data Storage stage**:

* You will be required to establish Where the data will be stored,
* Establish measures that will be taken to prevent internal and external security breaches.

**In the Data Sharing stage**:

* You will establish Which third-party vendors in your supply chain may have access to the data you are collecting.
* As well how will you hold them contractually accountable to the same regulations you are liable for.

**In the Data Retention and Disposal stages**:

* You will be required to establish What policies and processes you will follow for the retention and deletion of personal data after a designated time.
* And how will you ensure that in the case of data deletion, it will be removed from all locations, including third-party systems?

At each of these stages, you will be required to maintain an auditable trail of personal data acquisition, processing, storage, access, retention, and deletion.

**Technology as an Enabler**

Some of the controls made available through different tools and technologies for ensuring compliance to governance regulations.

**Authentication and Access Control**

Today's platforms offer **layered authentication processes**, such as a combination of passwords,

tokens, and biometrics, to provide foolproof protection against unauthorized access of

data. **Authentication systems are designed to verify that you are who you say you are**.

**Access control systems** ensure that authorized users have access to resources, both systems

and data, based on their user group and role.

Databases, for example, have the concept of roles and privileges so only authorized users

and applications can access specific objects, such as tables or rows or columns, in a database.

**Encryption and Data Masking**

Using encryption, data is converted to an encoded format that can only be legible once

it is decrypted via a secure key. Encryption of data is available both for **data at rest**, as it resides in the storage systems, and **data in transit**, as it moves through browsers, services, applications, and storage systems.

**Data Masking** provides anonymization of data for downstream processing and **pseudonymization**

of data.

Using Anonymization, the presentation layer is abstracted without changing the data in

the database itself. For example, replacing characters with symbols when they are displayed on the screen.

**Pseudonymization** of data is a de-identification process where personally identifiable information

is replaced with artificial identifiers so that a data set cannot be traced back to an individual's identity—for example, replacing the name with a random value from the names dictionary.

**Hosting options** in on-premise and cloud systems that comply with the requirements and restrictions for international data transfers.

**Monitoring and Alerting functionalities Security**

monitoring helps to proactively monitor, track, and react to security violations across infrastructure, applications, and platforms.

Monitoring systems also provide detailed audit reports that track access and other operations on the data. Alerting functionalities flag security breaches as they occur so that immediate remedial actions can be triggered. The alerts are based on the severity and urgency level of the breach, which is pre-defined in the system.

**Erasure Data erasure** is a software-based method of permanently clearing data from a system by overwriting. This is different from a simple deletion of data since deleted data can still be retrieved.

**Summary and Highlights**

In this lesson, you have learned:

Data Governance is a collection of principles, practices, and processes that help maintain the security, privacy, and integrity of data through its lifecycle.

Personal Information and Sensitive Personal Information, that is, data that can be traced back to an individual or can be used to identify or cause harm to an individual, needs to be protected through governance regulations.

General Data Protection Regulation, or GDPR, is one such regulation that protects the personal data and privacy of EU citizens for transactions that occur within EU member states.

Regulations, such as HIPAA (Health Insurance Portability and Accountability Act) for Healthcare, PCI DSS (Payment Card Industry Data Security Standard) for retail, and SOX (Sarbanes Oxley) for financial data are some of the industry-specific regulations.

Compliance covers the processes and procedures through which an organization adheres to regulations and conducts its operations in a legal and ethical manner.

Compliance requires organizations to maintain an auditable trail of personal data through its lifecycle, which includes acquisition, processing, storage, sharing, retention, and disposal of data.

Tools and technologies play a critical role in the implementation of a governance framework, offering features such as:

* Authentication and Access Control.
* Encryption and Data Masking.
* Hosting options that comply with requirements and restrictions for international data transfers.
* Monitoring and Alerting functionalities.
* Data erasure tools that ensure deleted data cannot be retrieved.

**Optional: Overview of the DataOps Methodology**

Gartner defines DataOps as a collaborative data management practice focused on improving the communication, integration, and automation of data flows between data managers and consumers across an organization. DataOps aims to create predictable delivery and change management of data, data models, and related artifacts. DataOps uses technology to automate data delivery with the appropriate levels of security, quality, and metadata to improve the use and value of data in a dynamic environment.” (Source: <https://blogs.gartner.com/nick-heudecker/hyping-dataops/>) A small team working on a simpler or limited number of use cases can meet business requirements efficiently. As data pipelines and data infrastructures get more complex, and data teams and consumers grow in size, you need development processes and efficient collaboration between teams to govern the data and analytics lifecycle. From data ingestion and data processing to analytics and reporting, you need to reduce data defects, ensure shorter cycle times, and ensure 360-degree access to quality data for all stakeholders. DataOps helps you achieve this through metadata management, workflow and test automation, code repositories, collaboration tools, and orchestration to help manage complex tasks and workflows. Using the DataOps methodology ensures all activities occur in the right order the right security permissions. It helps set in a continual process that allows you to cut wastages, streamline steps, automate processes, increase throughput, and improve continually. Several DataOps Platforms are available in the market, some of the popular ones being IBM DataOps, Nexla, Switchboard, Streamsets, and Infoworks.

**DataOps Methodology:**

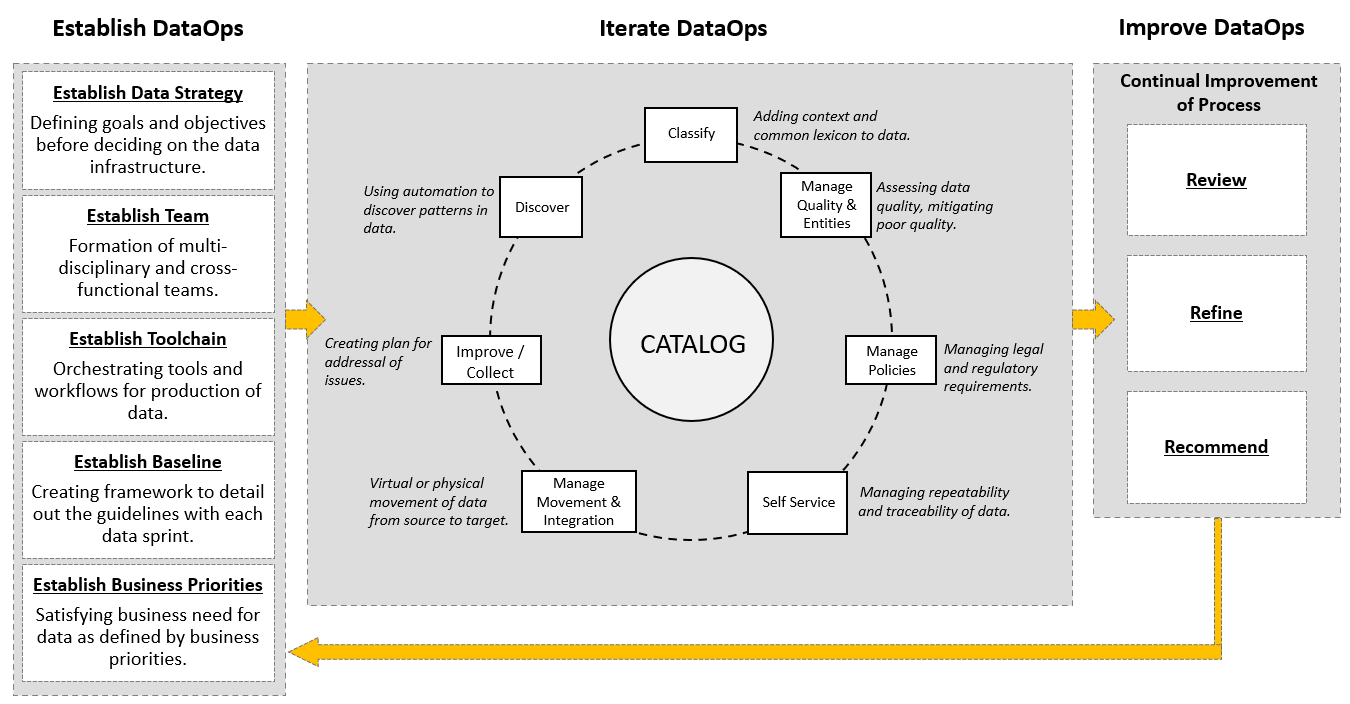
The purpose of the DataOps Methodology is to enable an organization to utilize a repeatable process to build and deploy analytics and data pipelines. Successful implementation of this methodology allows an organization to know, trust, and use data to drive value.

It ensures that the data used in problem-solving and decision making is relevant, reliable, and traceable and improves the probability of achieving desired business outcomes. And it does so by tackling the challenges associated with inefficiencies in accessing, preparing, integrating, and making data available.

In a nutshell, the DataOps Methodology consists of three main phases:

* The **Establish DataOps Phase** provides guidance on how to set up the organization for success in managing data.
* The **Iterate DataOps Phase** delivers the data for one defined sprint.
* The **Improve DataOps Phase** ensures learnings from each sprint is channeled back to continually improve the DataOps process.

The figure below presents a high-level overview of these phases and the key activities within each of these phases.



**Benefits of using the DataOps methodology:**

Adopting the DataOps methodology helps organizations to organize their data and make it more trusted and secure. Using the DataOps methodology, organizations can:

* Automate metadata management and catalog data assets, making them easy to access.
* Trace data lineage to establish its credibility and for compliance and audit purposes.
* Automate workflows and jobs in the data lifecycle to ensure data integrity, relevancy, and security.
* Streamline the workflow and processes to ensure data access and delivery needs can be met at optimal speed.
* Ensure a business-ready data pipeline that is always available for all data consumers and business stakeholders.
* Build a data-driven culture in the organization through automation, data quality, and governance.

As a data practitioner, using the methodology can help you reduce development time, cut wastages and duplication of effort, increase your productivity and throughput, and ensure that your actions produce the best possible quality of data.

With DataOps, data professionals, consumers, and stakeholders can collaborate more effectively towards the shared goal of creating valuable insights for business. While implementing the methodology will require systemic change, time, and resources, but in the end, it makes data and analytics more efficient and reliable.

Interestingly, it also opens up additional career opportunities for you as a data engineer. **DataOps Engineers** are technical professionals that focus on the development and deployment lifecycle rather than the product itself. And as you grow in experience, you can move into more specialist roles within DataOps, contributing to defining the data strategy, developing and deploying business processes, establishing performance metrics, and measuring performance.