02. The Data Engineering Lifecycle

Lifecycle:

- Generation
- Storage
- Ingestion
- Transformation
- Serving data

Undercurrents of the data engineering lifecycle:

- Security
- Data management
- DataOps
- Data architecture
- Orchestration
- Software engineering

Generation

Key engineering considerations for generation:

- Is it application/IoT/database?
- At what rate is data generated.
- Quality of the data.
- Schema of ingested data.
- How frequently should data be pulled from the source system?
- Will reading from a data source impact its performance?

Storage

Key engineering considerations for storage:

- Data volumes, frequency of ingestion, files format.
- Scaling (total available storage, read operation rate, write volume, etc.).
- Capturing metadata (schema evolution, data flows, data lineage)
- Is this a pure storage solution (object storage), or does it support complex query patterns (i.e., a cloud data warehouse)?
- Is the storage system schema-agnostic (object storage)? Flexible schema (Cassandra)? Enforced schema (a cloud data warehouse)?

- How are you tracking master data, golden records data quality, and data lineage for data governance?
- How are you handling regulatory compliance and data sovereignty?

Temperatures of data:

- hot data
- lukewarm data
- cold data

Ingestion

Ingestion part is usually located biggest bottlenecks of the lifecycle. The source systems are normally outside your direct control and might randomly become unresponsive or provide data of poor quality.

Key engineering considerations for the ingestion phase:

- Data availability and source reliability.
- How sink will handle volume, format and frequency?
- Batch or streaming?
- Push or Pull?

Batch ingestion: convenient way of processing this stream in large chunks—for example, handling a full day's worth of data in a single batch.

Streaming ingestion:allows to provide data to downstream systems in a continuous, real-time fashion. Real-time (or near real-time) means that the data is available to a downstream system a short time after it is produced (e.g., less than one second later).
Micro-batching: used in ex. Spark Streaming with data taken from 1 second period.

Push model: a source system writes data out to a target, whether a database, object store, or filesystem. Example is standard ETL process.

Pull model: data is retrieved from the source system. Example is CDC with logs.

Transformation

Examples of transformations:

- mapping data into correct types,
- transforming the data schema and applying normalization,
- large-scale aggregation for reporting,
- featurizeing data for ML processes,
- enriching the data.

Other terms

Reverse ETL: takes processed data from the output side of the data engineering lifecycle and feeds it back into source systems. It allows us to take analytics, scored models, etc., and feed these back into production systems or SaaS platforms. For some engineers view as a anti-pattern.

Security

Security good practices:

- The principle of least privilege means giving a user or system access to only the essential data and resources to perform an intended function
- The first line of defense for data security is to create a culture of security that permeates the organization. All individuals who have access to data must understand their responsibility in protecting the company's sensitive data and its customers.
- Data security is also about timing—providing data access to exactly the people and systems that need to access it and only for the duration necessary to perform their work. Data should be protected from unwanted visibility, both in flight and at rest, by using encryption, tokenization, data masking, obfuscation, and simple, robust access controls.
- Knowledge of user and identity access management (IAM) roles, policies, groups, network security, password policies, and encryption are good places to start.

Data Management

Disciplines of Data Management

```
Data management has quite a few facets, including the following:
- Data governance, including data quality, integrity, security, discoverability
and accountability
- Data modeling and design
- Metadata management
- Data lineage
- Storage and operations
- Data integration and interoperability
- Data lifecycle management
- Data systems for advanced analytics and ML
- Ethics and privacy
```

Name most important book of Data Management and it's definition of a DM.

```
The Data Management Association International (DAMA) Data Management Body of Knowledge (DMBOK), which we consider to be the definitive book for enterprise data management, offers this definition:

Data management is the development, execution, and supervision of plans, policies, programs, and practices that deliver, control, protect, and enhance the value of data and information assets throughout their lifecycle.
```

Data governance

```
According to Data Governance: The Definitive Guide, “Data governance is, first and foremost, a data management function to ensure the quality, integrity, security, and usability of the data collected by an organization.”
```

Master Data Management

```
Master data is data about business entities such as employees, customers, products, and locations. As organizations grow larger and more complex through organic growth and acquisitions, and collaborate with other businesses, maintaining a consistent picture of entities and identities becomes more and more challenging. Master data management (MDM) is the practice of building consistent entity definitions known as golden records.

MDM reaches across the full data cycle into operational databases. It may fall directly under the purview of data engineering but is often the assigned responsibility of a dedicated team that works across the organization.
```

Data lineage

```
As data moves through its lifecycle, how do you know what system affected the data or what the data is composed of as it gets passed around and transformed? Data lineage describes the recording of an audit trail of data through its lifecycle, tracking both the systems that process the data and the upstream data it depends on. Data lineage helps with error tracking, accountability, and debugging of data and the systems that process it.
```

Data integration and interoperability

```
The process of integrating data across tools and processes. As we move away from a
single-stack approach to analytics and toward a heterogeneous cloud environment in
which various tools process data on demand, integration and interoperability
occupy an ever-widening swath of the data engineer’s job. Increasingly,
integration happens through general-purpose APIs rather than custom database
connections. For example, a data pipeline might pull data from the Salesforce API,
store it to Amazon S3, call the Snowflake API to load it into a table, call the
API again to run a query, and then export the results to S3 where Spark can
consume them.
```

Data privacy

```
Data privacy and data retention laws such as the GDPR and the CCPA require
data engineers to actively manage data destruction to respect users’ “right to be
forgotten.” Data engineers must know what consumer data they retain and must have
procedures to destroy data in response to requests and compliance requirements.

How do ethics and privacy impact the data engineering lifecycle? Data engineers
need to ensure that datasets mask personally identifiable information (PII) and
other
sensitive information; bias can be identified and tracked in datasets as they are
transformed.
Regulatory requirements and compliance penalties are only growing. Ensure
that your data assets are compliant with a growing number of data regulations,
such
as GDPR and CCPA. Please take this seriously. We offer tips throughout the book to
ensure that you’re baking ethics and privacy into the data engineering lifecycle.
```

DataOps

DataOps

```
Whereas DevOps aims to improve the release and quality of software products,
DataOps does the same thing for data products.

DataOps is a collection of technical practices, workflows, cultural norms, and
architectural patterns that enable:
• Rapid innovation and experimentation delivering new insights to customers with
increasing velocity
• Extremely high data quality and very low error rates
• Collaboration across complex arrays of people, technology, and environments
• Clear measurement, monitoring, and transparency of results
```

```
DataOps has three core technical elements: automation, monitoring and
observability, and incident response.
```

DataOps elements


```
DataOps has three core technical elements: automation, monitoring and
observability, and incident response.
```


```
Automation:
- change management (environment, code, and data version control)
- continuous integration/continuous deployment (CI/CD)
- configuration as code
- monitor and maintain the reliability of technology and systems (data pipelines,
orchestration, etc.), with the added dimension of checking for data quality,
data/model drift, metadata integrity, and more

Observability and monitoring
- monitoring,
- logging,
- alerting,
- tracing are all critical to getting ahead of any problems along the data
engineering lifecycle.

Incident response is about using the automation and observability capabilities
mentioned previously to rapidly identify root causes of an incident and resolve it
as reliably and quickly as possible.
```

Data Architecture

What is orchestration?

```
- Orchestration is the process of coordinating many jobs to run as quickly and
efficiently
as possible on a scheduled cadence.
- Orchestration system stays online with high availability.
- Orchestration systems also build job history capabilities, visualization, and
alerting.
- Advanced orchestration engines can backfill new DAGs or individual tasks as they
are added to a DAG.
- Orchestration is strictly a batch concept.
```

Infrastructure as code (IaC)

```
IaC pplies software engineering practices to the configuration and management of
infrastructure. The infrastructure management burden of the big data era has
decreased as companies have migrated to managed big data systems—such as
Databricks and Amazon Elastic MapReduce (EMR)—and cloud data warehouses. When data
engineers have to manage their infrastructure in a cloud environment, they
increasingly do this through IaC frameworks rather than manually spinning up
instances and installing software. Several
```