1 Data Pipelines

Data pipelines move data from sources to consumers. This often needs to be done reliabilty, efficiencly, and with a certain amount of auditability.

Logical steps are often:

- ingestion: consuming the business events. These can be batches of data, or individual events.
- transport: how data movement is organized, via files, APIs, message queues, etc.
- transform: ETL the data, apply validation, enrichments, etc.
- storage: data warehouse, data lake, etc.
- serving: enable data use via APIs, queries, BI tools, etc.

Key issues:

- schemas: The data layout can change, at input, output, intermediate steps, etc.
- idempotency: As much as practicable, data operations should be idemponent, meaning that re-running the step does nothing. This enables easier recovery modes, such as simply restarting upon failure.
- exactly-once vs at-least-once: Semantically, at-least-once may introduce duplicates in case of retries. Some systems are OK with that. exactly-once ensures that the business events are processed exactly once. For example, a bank transaction should only be processed once, even if it's retransmitted.
- data ordering: some workflows have to be done in a particular order.
- partitioning: how is data partitioned.
- latency: every step adds a delay—how much is tolerable.
- throughput: how much can be pushed through the pipeline.
- retries: how are retries kicked off, processed.
- checkpoints: how are checkpoints handled, so retries don't start from nothing.
- lineage: need to maintain the source of data.
- metadata: data about data
- observability: how often things are logged, and how opaque is the pipeline.
- SLAs: what are agreements with upsteams/downstreams.
- orchestration: what is running the pipeline (framework that executes the workflow). cost optimization.

- scheduling: often part of the orchestration framework, what runs and when.
- security: authentication and authorization, as well as logging. access control. privacy.
- encryption: often data needs to be encrypted at-rest and in-flight.
- testing: need ability to test pipeline
- versioning: data moving through the pipeline may be versioned.
- rollback: in case of failure, can restart from last good version.
- disaster recovery: often involves moving over to another data center or availability zone.
- deduplication: how are duplicates handled.
- data quality: data integrity checks
- late arrivals: reprocess or just log/insert?
- scalability/parallelism: if data doubles or quadruples overnight, will system keep up?
- fault tolerance: what to do in case of failures.
- compliance: what needs to be logged/investigated to ensure compliance.