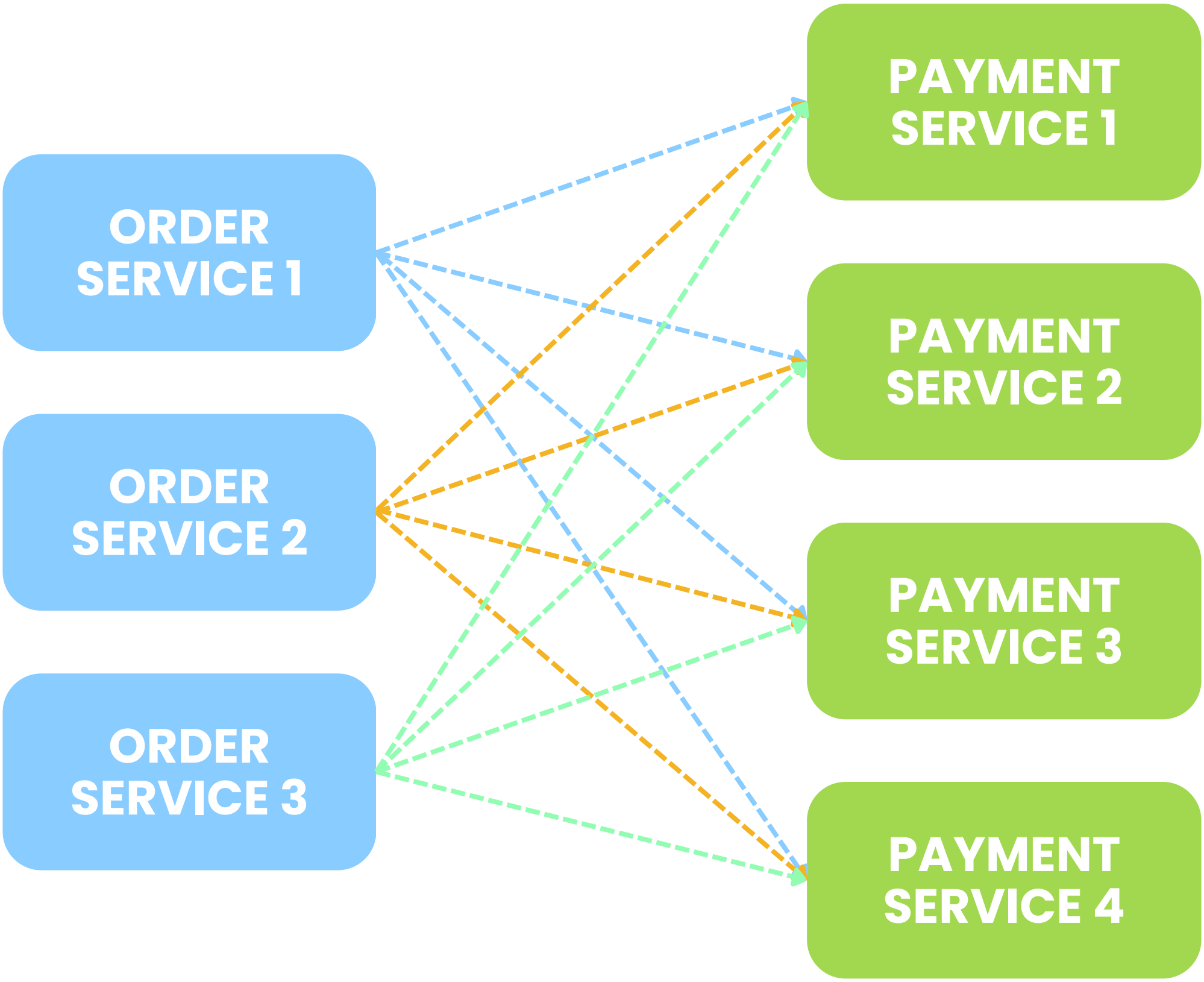
9 Software Architecture Patterns for Distributed Systems

Follow For More

Hasnain Ahmed Shaikh

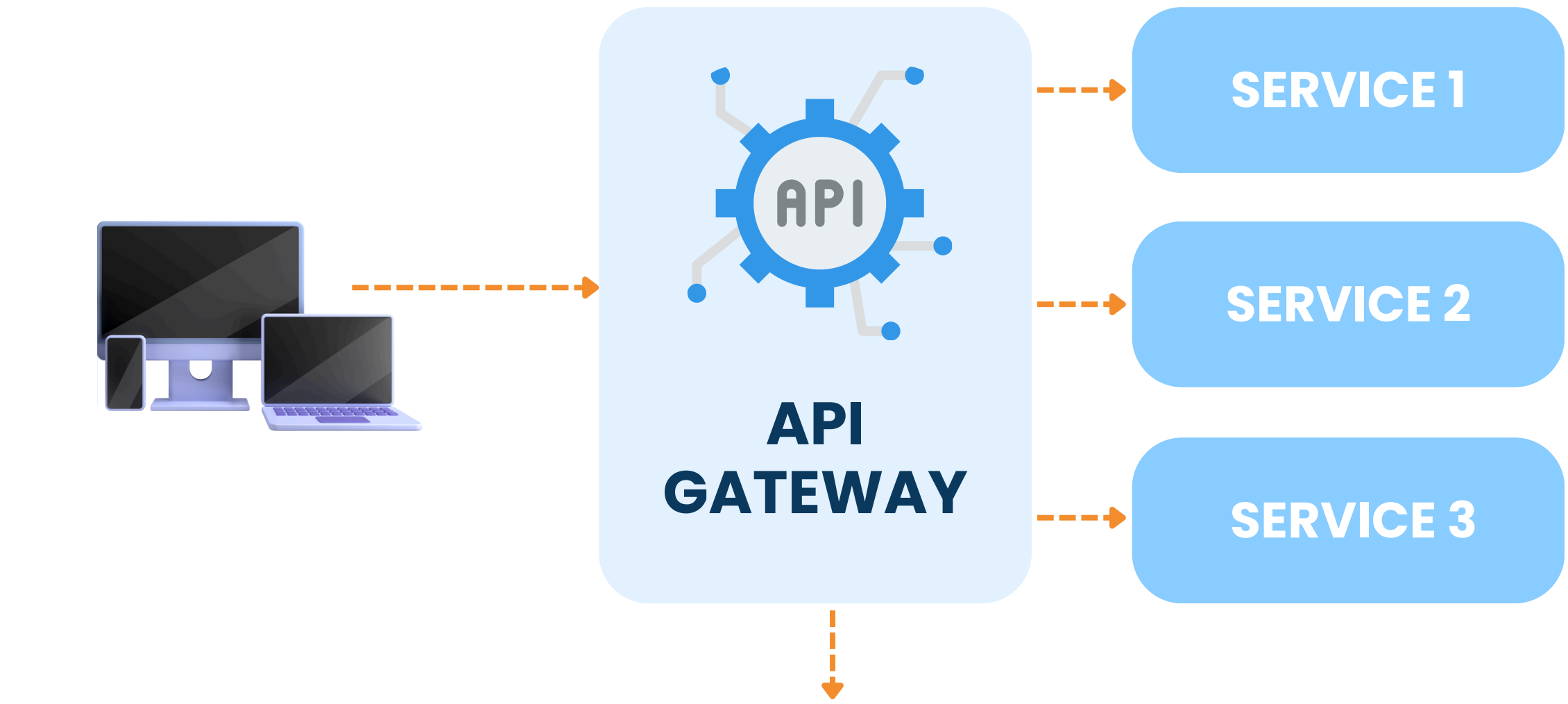
1.Peer To Peer

Services communicate directly with each other, creating a web of interdependencies. While flexible, this can lead to tight coupling and complex service management.



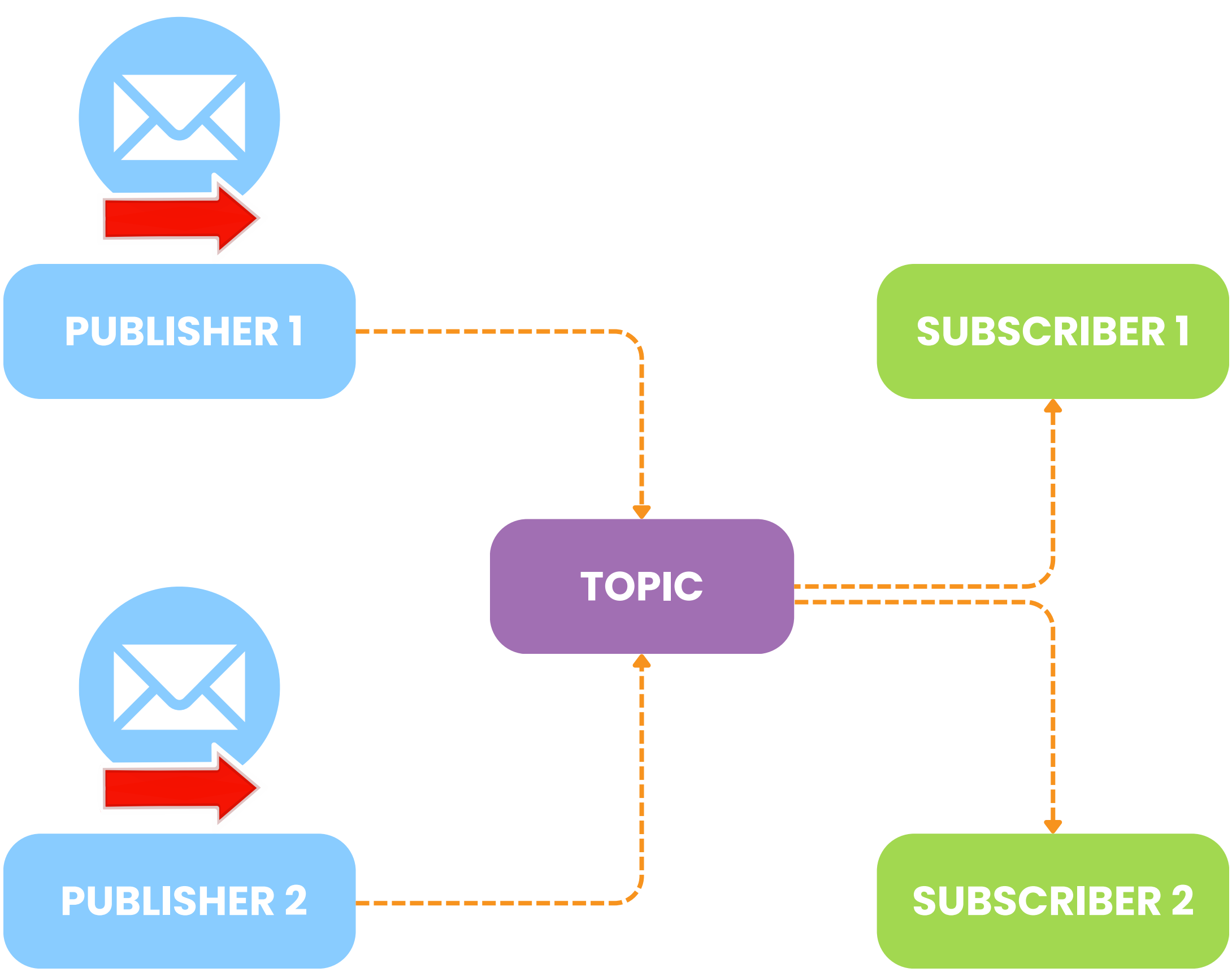
2. API GATEWAY

A centralized gateway handles incoming client requests, routing them to appropriate services while managing cross-cutting concerns like authentication, rate limiting, and protocol translation.



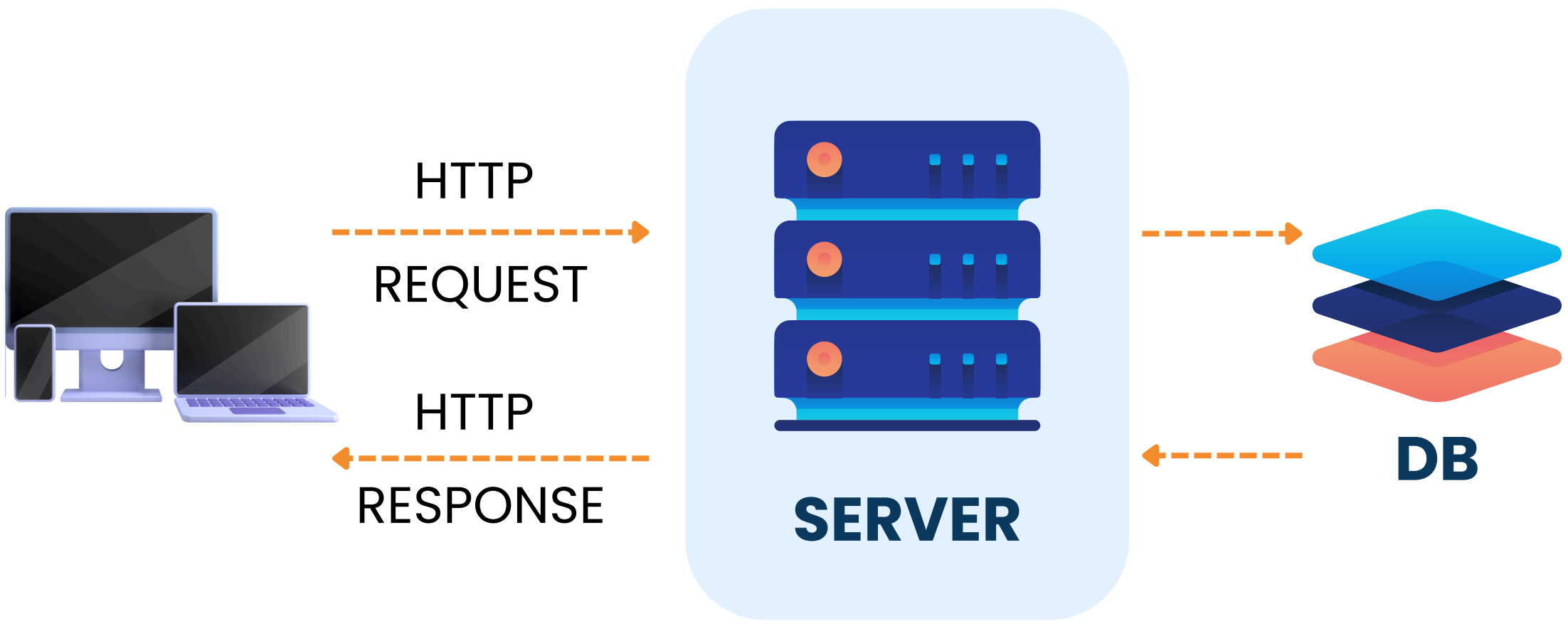
3. Pub-Sub

Publishers send messages to a topic without knowing who the subscribers are, promoting loose coupling and asynchronous communication between services.



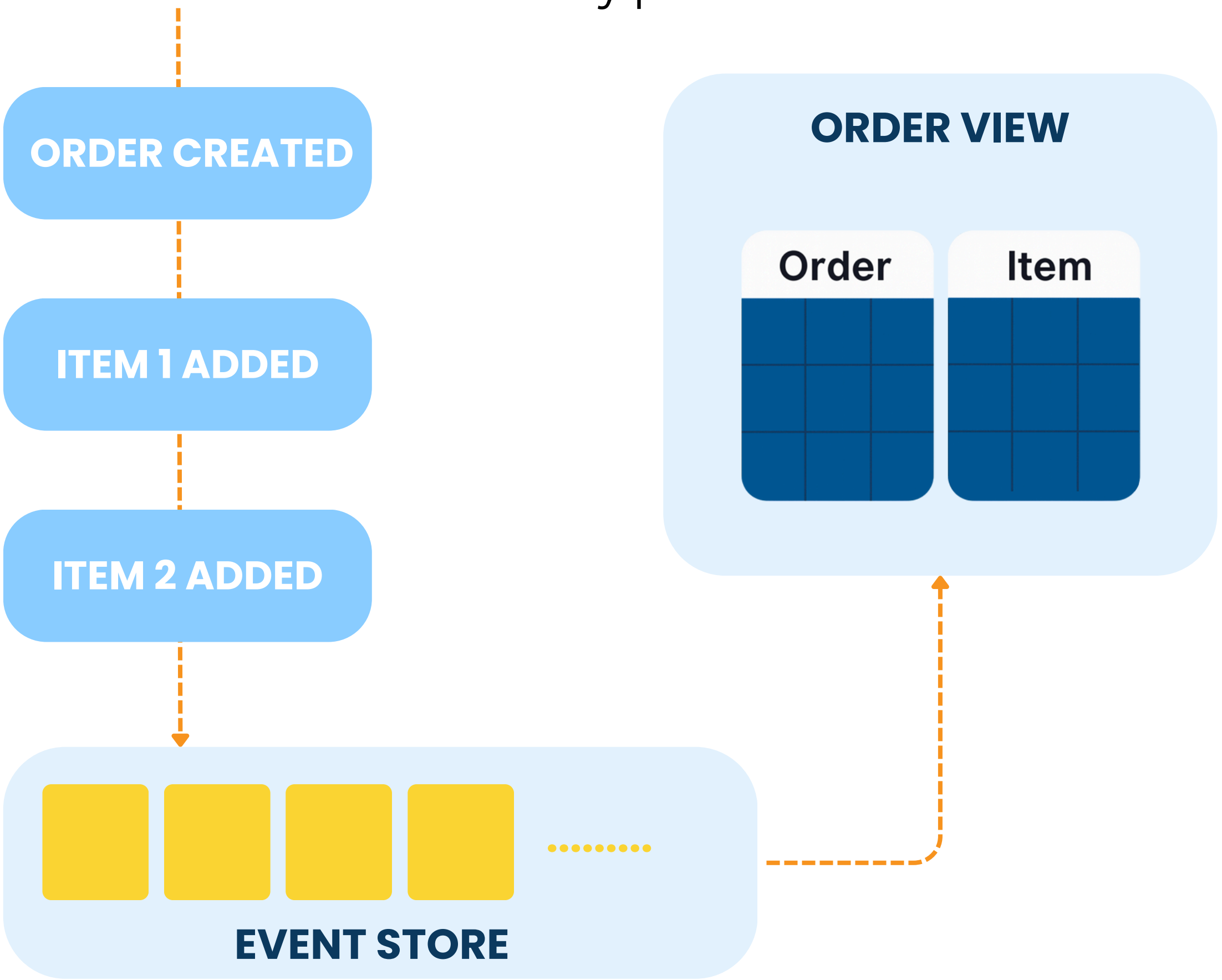
4. Request Response

Clients send a request and wait for a direct response from a server — a simple, synchronous interaction model, common in HTTP APIs.



5. Event Sourcing

State changes are captured as a sequence of events stored in an event store, allowing the system state to be rebuilt or audited at any point.



6. ETL

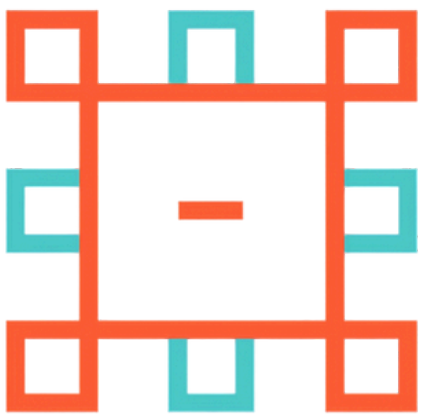
Data is extracted from multiple sources, transformed into a consistent format, and loaded into a target system like a data warehouse for analytics.

DB

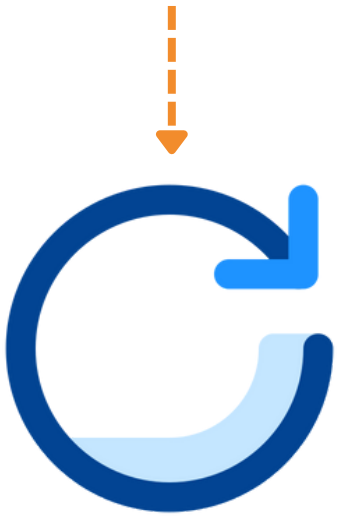
FILES

APPLICATIONS

EXTRACT



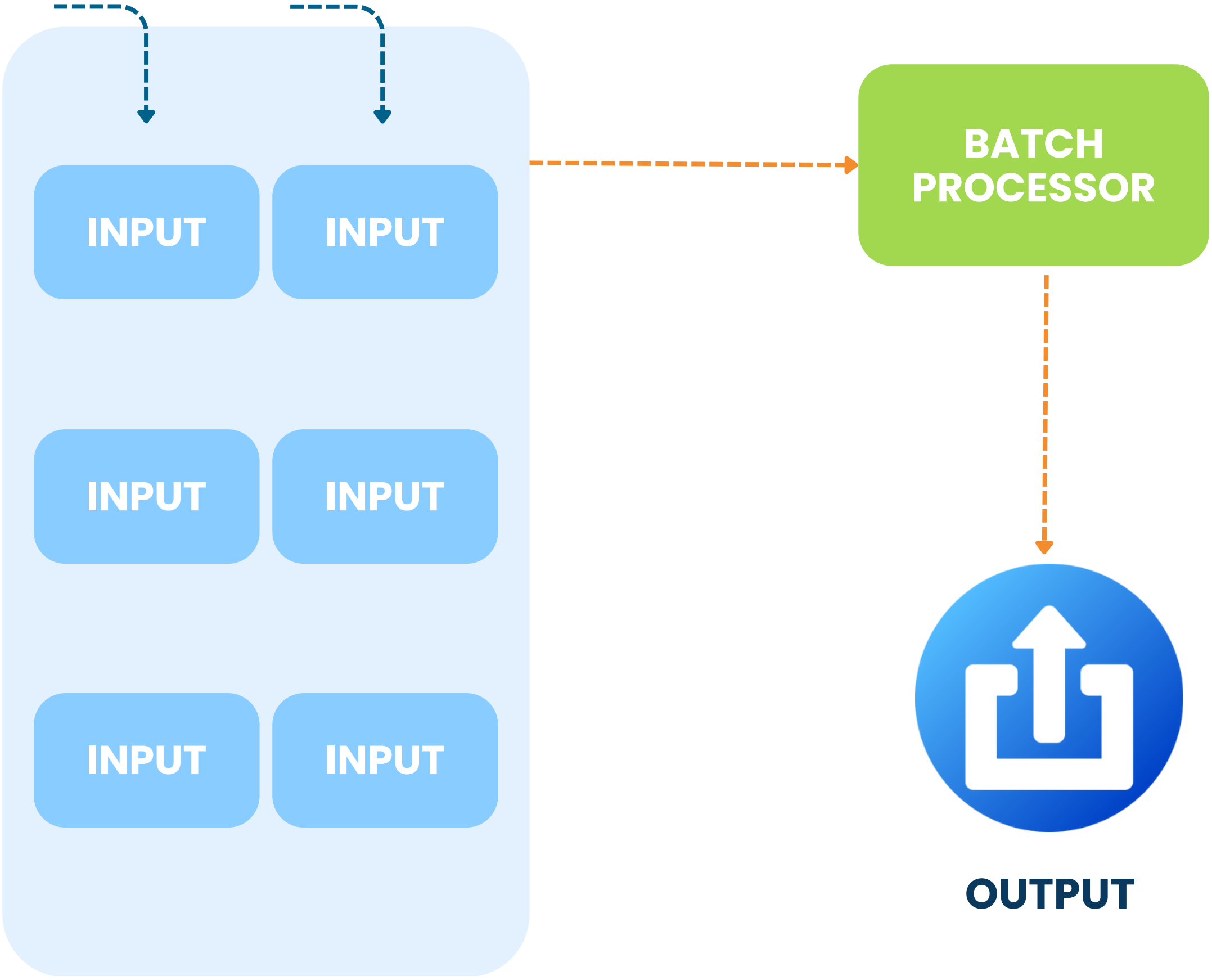
TRANSFORM



LOAD

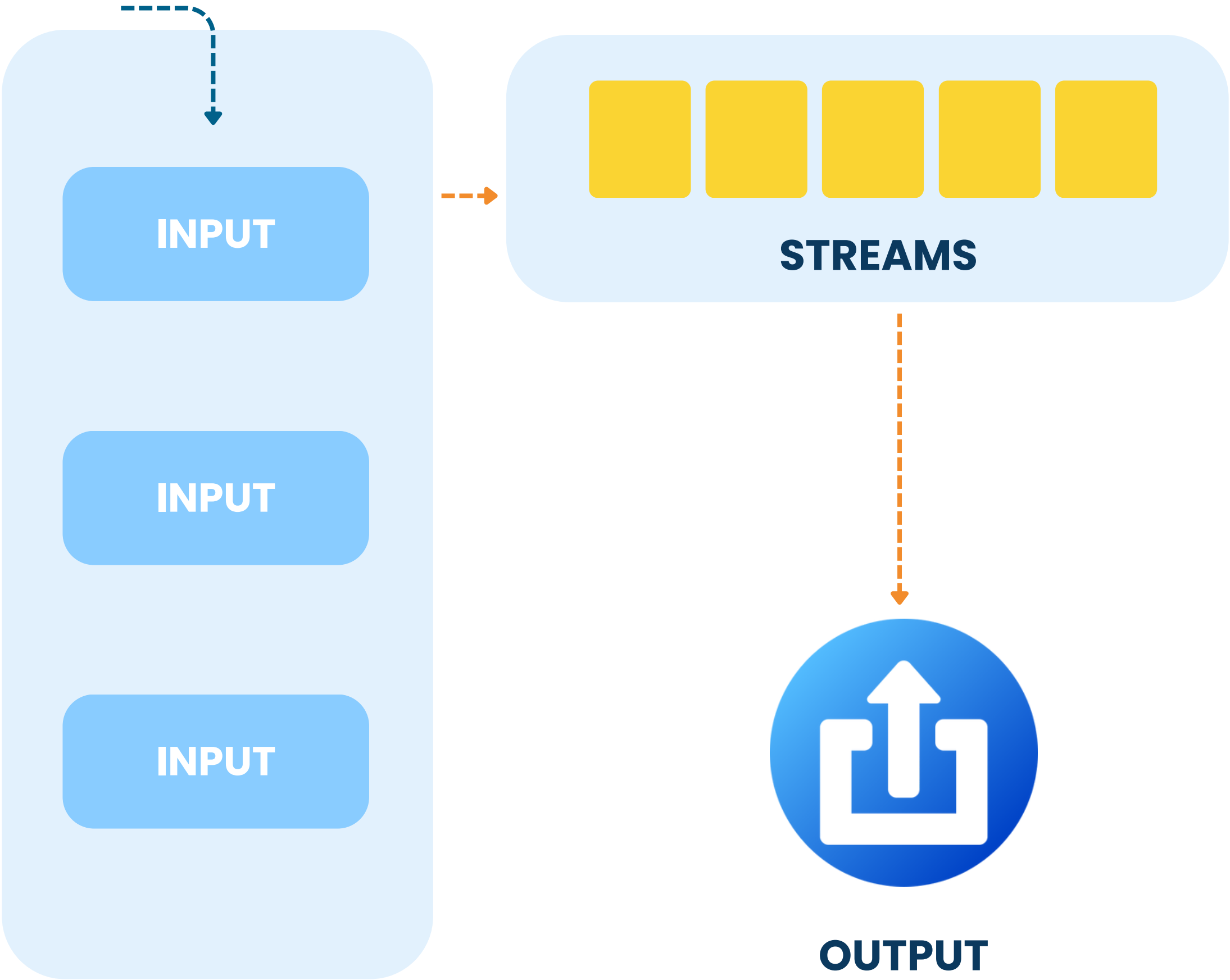
7. Batching

Data inputs are collected over time and processed together in bulk, improving efficiency and reducing the overhead of handling items individually.



8. Streaming Processing

Data is processed in real-time as it arrives via streams, enabling quick insights and actions with low latency.



9. Orchestration

A central orchestrator coordinates workflows between services, handling dependencies and execution order to ensure a business process is completed.