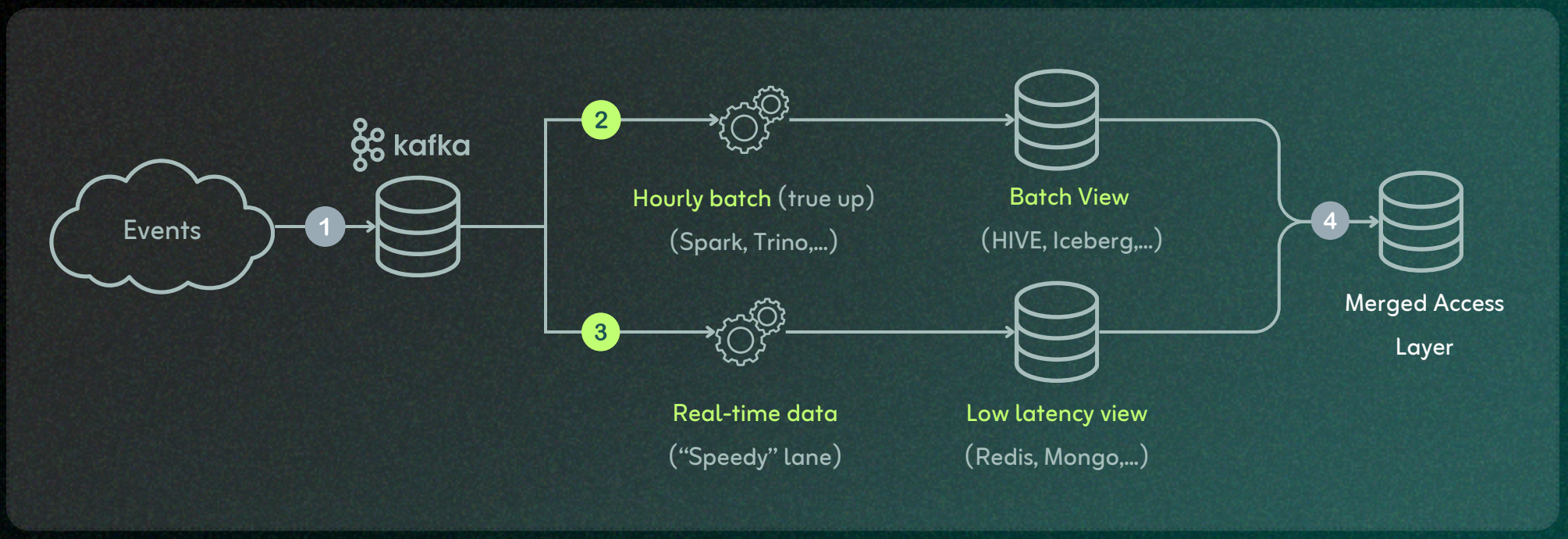
LAMBDA ARCHITECTURE

☑ LAMBDA ARCHITECTURE (2-code bases)

Data processing pattern designed to handle massive quantities of data by leveraging both batch and stream processing.



📌 How does it work?

1. Raw events ingested into Kafka (acts as a distributed message queue, handling high-throughput data streams)
   - Durable, fault-tolerant buffer for incoming data.
   - Enables parallel processing paths.
2. Batch layer (upper path):
   Processes large amounts of historical data periodically.
   - Generates pre-computed batch views.
   - Optimized for accuracy & completeness.
3. Real-time layer (lower path):
   Processes events immediately as they arrive ('speedy lane' for real time data) & writes to low-latency view.
   - Compensates for high latency of batch layer.
   - Creates real-time views.
4. Serving layer
   Merges outputs from both and speed layers.
   Provides query interface & responds to queries with minimal latency.

📌 How Lambda handles

Small Files management:

- The streaming pipeline generates small files, but ONLY WITHIN a limited time window (day/hour).
- The batch pipeline performs a "true up" (daily or hourly) which:
  - Compacts the small files & consolidates them into larger, more efficient files.
  - Optimizes storage and query performance.
- The time-bound nature prevents unbounded small file proliferation.

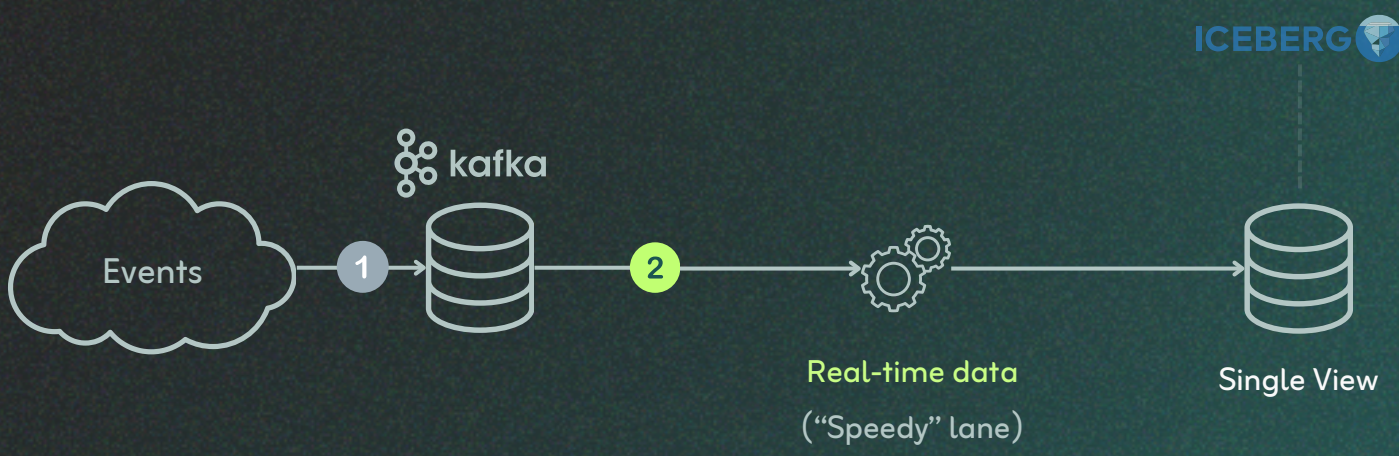
Data Quality management:

- DQ constraints intentionally pushed to the batch pipeline (batch is better at DQ)
- Batch pipeline advantages for quality:
  - Access to complete data sets.
  - Can perform comprehensive validation, cross-record validations.
  - Has more computing resources available.
  - Can handle complex data quality rules.

KAPPA ARCHITECTURE

☑ KAPPA ARCHITECTURE (1-code base)

Simplified pattern. Treats everything as a stream, eliminating the dual-path complexity of Lambda architecture.



📌 How does it work?

1. Raw events enter the system and are immediately stored in Kafka (acts as primary storage system & streaming backbone). Single source of truth for all data.

2. Real-time layer ("speedy" lane):
All data processing (historical & real-time data) through a single streaming pipeline, using the same code path.
   - Uses stream processing frameworks (ie. Apache Flink, Spark Streaming, or Kafka Streams)
   - Can reprocess entire historical data by replaying the Kafka topics.
   - Single View (serving layer), typically uses databases optimized for real-time access

📌 How Kappa handles

Small Files management:

- Streaming pipeline generates small files
- Iceberg acts as the single view layer with built-in compaction capabilities (automatically manages compression at regular timeframes).

Data Quality management:

A. Simple checks in streaming:

- Row-level basic validations (ie. nulls, enums)

B. Volumetric checks via observability tools:

- Enables alerting based on volume patterns.
- Provides operational visibility into data flow.
  ie. setting up alerts for unexpected volume changes

C. Complex checks in downstream batch layer:

- More sophisticated DQ checks
- Downstream consumers wait for validations
- WAP (Write-Audit-Publish) patterns still need to happen at the batch layer
- Data remains partitioned in Iceberg tables until "day is done"
- Batch pipeline can then perform DQ checks