Your Kafka Credentials

- Please do not publish these credentials in any public repo anywhere!
- Your environment keys are:
  - KAFKA_WEB_TRAFFIC_SECRET
    - **Y05ZLtvCqopUbMkG2LcC24MUWocNuTYVKp1pf1U7YrkjJg9VBS1PBdGuSt3rx+mD**
  - KAFKA_WEB_TRAFFIC_KEY
    - **CAKHHIO74VLR7LYK**

What's the difference between streaming, near real-time and real-time?

- Streaming (or continuous)
  - Data is processed as it is generated
  - Example: Flink
- Near real-time
  - Data is processed in small batches every few minutes
  - Example: Spark Structured Streaming

Real-time and streaming are often synonymous but not always!

Less technical people might think real-time and batch are synonymous!

What does real-time mean from stakeholders?

- It RARELY means streaming
- It usually means low-latency or predictable refresh rate

Should you use streaming?

- Considerations
  - Skills on the team
  - What is the incremental benefit?
  - Homogeneity of your pipelines
  - The tradeoff between daily batch, hourly batch, microbatch, and streaming
  - How should data quality be inserted (batch pipelines have easier DQ paths)

Streaming-only Use Cases

- **KEY**: LOW LATENCY MAKES OR BREAKS THE USE CASE
- Examples:
  - Detecting fraud, preventing bad behavior
  - High-frequency trading
  - Live event processing

Gray-area use cases (micro-batch may work too)

- Data that is served to customers
- Reducing the latency of upstream master data
  - Notifications dataset had an 9 hour after midnight latency
  - Micro batch cut it to 1 hour

No-go Streaming use cases (use batch please!)

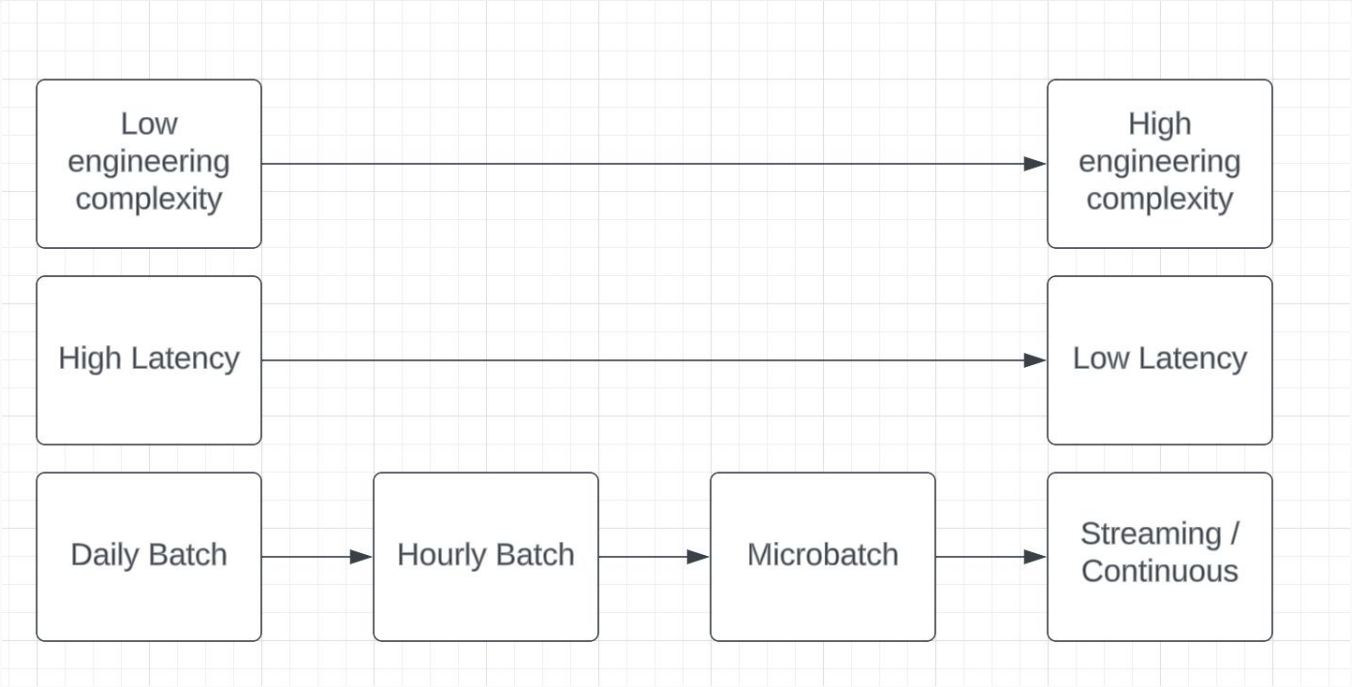
- Ask the question
  - What is the incremental benefit of reduced latency?
- Analysts complaining that the data isn't up-to-date
  - Yesterday's data by 9 AM is good enough for MOST analytical use cases

How are streaming pipelines different from batch pipelines?

- Streaming pipelines run 24/7! Batch pipelines run for a small percentage of the day
- Streaming pipelines are much more software engineering oriented
  - They act a lot more like servers than DAGs
- Streaming pipelines need to be treated as such and have more unit test and integration test coverage like servers!

The streaming -> batch continuum

EcZachly Inc



The streaming -> batch continuum

- Real time is a myth!
  - You'll have seconds of latency just for the event generation -> Kafka -> Flink -> sink
- Pipelines can be broken into 4 categories
  - Daily batch
  - Hourly batch (sometimes called near real-time)
  - Microbatch (sometimes called near real-time)
  - Continuous processing (usually called real-time)

The structure of a streaming pipeline

- The Sources
  - Kafka, RabbitMQ
- Enriched dimensional sources (i.e. side inputs)

The structure of a streaming pipeline

- The compute engine
  - Flink
  - Spark Structured Streaming
- These engines make sense of the incoming streams of data

The structure of a streaming pipeline

- The destination, also called “the sink”
- Common sinks
  - Another Kafka topic
  - Iceberg
  - Postgres

Streaming challenges

- Out of order events
- Late arriving data
- Recovering from failures

Out of order events

- How does Flink deal with out-of-order events?
  - WATERMARKING

Recovering from failures

- Flink manages this in a few ways
- Offsets
  - Earliest offset
  - Latest offset
  - Specific timestamp (maybe like when it failed)
- Checkpoints
- Savepoints

Late-arriving Data

- How late is too late?
- Batch handles this mostly by waiting, although batch has issues around midnight UTC too!

The Lab today

- Writing a job that connects to Kafka
- Filtering out events we don’t care about
- Writing the events to Postgres