Confluent Kafka — Complete Interview & Hands-On Guide

A Light■Theme Illustrated PDF covering Apache Kafka, Confluent Platform, Kafka Streams, KSQL, and Schema Registry with interview Q&A; and examples.

1. Introduction to Apache Kafka

Kafka is a distributed streaming platform enabling real-time messaging and data pipelines with high throughput and durability.

Component	Description
Producer	Publishes records to Kafka topics.
Consumer	Subscribes and reads messages from topics.
Broker	Kafka server managing topics and partitions.
Topic	Named data stream storing records.
Partition	Parallelism unit with ordered records.
Offset	Position of a record within a partition.
Consumer Group	Group of consumers sharing topic partitions.

Q: What is Kafka used for?

A: Used for real-time data streaming, log aggregation, event sourcing, and metrics collection.

Q: How does Kafka ensure durability?

A: Kafka writes to disk and replicates data across brokers for fault tolerance.

Q: What is a partition?

A: Partition is a parallelism unit ensuring ordered messages within itself.

Q: Difference between 'at-least once' and 'exactly once' delivery?

A: At-least once guarantees no data loss but possible duplicates; exactly once ensures no duplicates or loss.

2. Confluent Platform Overview

Confluent enhances Kafka with enterprise-grade components for schema management, connectors, stream processing, and monitoring.

Component	Purpose
Schema Registry	Manages Avro/JSON/Protobuf schemas for compatibility.
Kafka Connect	Framework for source and sink connectors.
KSQL / ksqlDB	SQL engine for Kafka Streams operations.
Kafka Streams	Java library for building stream processing apps.
Control Center	Monitoring and management UI for clusters.

3. Kafka Streams (KStreams / KTables)

Kafka Streams is a lightweight Java library for real-time stream transformations directly on Kafka topics.

Q: What are KStreams and KTables?

A: KStream represents a continuous sequence of records; KTable represents a changelog with latest state per key.

Q: Example: Word Count with KStreams

A: Processing pipeline example in Java.

```
StreamsBuilder builder = new StreamsBuilder();
KStream<String, String> textLines = builder.stream("input-topic");
KTable<String, Long> counts = textLines
    .flatMapValues(value -> Arrays.asList(value.toLowerCase().split(" ")))
    .groupBy((key, word) -> word)
    .count();
counts.toStream().to("output-topic");
```

Q: What are windowed aggregations?

A: Aggregations performed on events grouped by time windows (tumbling, hopping, sliding).

Q: How are joins handled?

A: Supports stream-stream, stream-table, and table-table joins within defined windows.

4. KSQL / ksqlDB

KSQL provides SQL syntax for Kafka Streams operations—allowing you to query, aggregate, and join Kafka topics using SQL.

Q: Example: Aggregate purchases per user

A: Summarize user purchases using SQL.

```
CREATE STREAM purchases (user_id VARCHAR, amount DOUBLE)
  WITH (KAFKA_TOPIC='purchases', VALUE_FORMAT='JSON');
CREATE TABLE totals AS
  SELECT user_id, SUM(amount) AS total
  FROM purchases
  GROUP BY user_id
  EMIT CHANGES;
```

Q: Stream–Stream Join Example

A: Joins two streams within a defined time window.

```
CREATE STREAM orders_enriched AS
SELECT p.order_id, p.amount, s.ship_status
FROM payments p
JOIN shipments s
WITHIN 1 HOUR
ON p.order_id = s.order_id
EMIT CHANGES;
```

5. Schema Registry

Schema Registry ensures producers and consumers exchange data using compatible schemas, enabling strong typing.

Q: Why use Schema Registry?

A: It enforces schema evolution and prevents breaking changes in data contracts.

Q: Schema Evolution Modes

A: Modes include BACKWARD, FORWARD, FULL, and NONE.

Q: AvroProducer Example

A: Producer example with Avro serialization.

```
from confluent_kafka import avro
from confluent_kafka.avro import AvroProducer

value_schema = avro.load("user.avsc")
key_schema = avro.load("user_key.avsc")

producer = AvroProducer({
  'bootstrap.servers': 'localhost:9092',
  'schema.registry.url': 'http://localhost:8081'
}, default_key_schema=key_schema, default_value_schema=value_schema)
```

Q: How are schemas versioned?

A: Each topic-key/value schema is tracked by a subject (e.g., topic-value) with incremental versions.

Q: Integration with Kafka Connect?

A: Connect converters use Schema Registry for serialization and validation (AvroConverter, JsonSchemaConverter).

6. Kafka & Confluent Best Practices

Key recommendations for production deployments.

• Use replication factor ≥3 for durability.

• Enable idempotent producers and transactions for exactly-once delivery.

• Leverage Schema Registry and Avro for backward compatibility.

• Monitor consumer lag via Control Center.

• Tune partitions and retention for throughput optimization.