Admissions & Support:
support@mockattempt.com
Phone / WhatsApp: +91 98765 43210
Course Code: KAFK • **Updated:** 03 Nov 2026
https://mockattempt.com/courses/kafka-training

KAFK • DISTRIBUTED EVENT STREAMING & MESSAGING | **BEGINNER TO PRINCIPAL APACHE KAFKA ARCHITECT**

100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER

Kafka Training

Master Apache Kafka from Zero-Copy OS PageCache, KRaft Consensus & ISR Replication to Idempotent Java Producers, CooperativeSticky Rebalancing, MirrorMaker 2, Debezium Kafka Connect, RocksDB Kafka Streams & JMX Lag Monitoring — with 200 Mock Tests & Voice MockAttempt AI Interviews at ₹4,299 (Incl. 18% GST).

Duration: 15–17 Weeks • Live + Hands-On Projects (208 Hrs) • **Modules:** 16 • **Live Topics:** 42 • **Subtopics:** 336+ • **Labs & Tasks:** 84 • **Mocks / AI Vivas:** 244 / 244 • **Projects:** 16 Module + 1 Live Client Prod

All-Inclusive Direct Fee (Incl. 18% GST • Zero Hidden Charges): Total Payable: ₹4,299 (Net Tuition ₹3,643 + Included 18% GST ₹656) • **Split Option:** 2 × ₹2,150 Split • **Effective ROI:** ~₹21/Guided Hr (~₹102/Live Topic)

Faculty: 100% Active Working IT Professionals • **Live Batch Tracks (7:00 AM – 8:00 PM IST Flexible Slots):** Weekday Live Online Batch (Mon – Fri • 1 Hr/Day • 7:00 AM – 8:00 PM) (Monday to Friday (1 Hour Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 1-Hour Slot Based on Trainer Availability) • 17–19 Weeks Fast-Track) | Weekend Live Online Batch (Sat – Sun • 2 Hrs/Day • 7:00 AM – 8:00 PM) (Saturday & Sunday (2 Hours Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 2-Hour Slot Based on Trainer Availability) • 19–21 Weeks Weekend Track)

✓ **Trainer LeetCode & HackerRank Mentorship (Java 21 LTS (Kafka Clients & Streams)): Live trainer support to crack LeetCode & HackerRank problems, optimize complexity, and handle edge cases.**

Core Technologies & Engineering Stack: Apache Kafka 3.8 (KRaft Mode) • Java 21 Kafka Clients API • Idempotent & Transactional Producers • Cooperative Consumer Groups • Kafka Connect & Debezium CDC • Kafka Streams & RocksDB State • MirrorMaker 2 & Prometheus JMX

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Architect fault-tolerant KRaft clusters with tuned partition counts, ISR replication (`acks=all` + `min.insync.replicas=2`), and log compaction
2. Build high-throughput idempotent Java producers and cooperative-rebalance consumer groups with deterministic offset commit guarantees
3. Engineer zero-code CDC pipelines with Kafka Connect, multi-region active-active replication via MirrorMaker 2, and stateful Kafka Streams topologies
4. Clear 244 Timed Mock Exams (10,000 Apache Kafka MCQs) and 200 Spoken Voice + MockAttempt AI Event-Streaming Architecture Interviews
5. KRaft broker architecture, topic partitioning, replication factors, ISR (`min.insync.replicas`), and leader election.
6. Crack LeetCode and HackerRank problems in Java 21 LTS (Kafka Clients & Streams) with live trainer mentorship on algorithmic/query patterns, edge cases, and optimal complexity.
7. Idempotent & transactional producers (`acks=all`), Transactional Outbox pattern, and Avro/JSON Schema Registry contracts.
8. Consumer Group rebalancing, manual offset commits, idempotent consumers, retry topics, and Dead-Letter Queues (DLQ).

Eligibility & Prerequisites: Basic familiarity with Java classes/interfaces and Linux command-line basics — distributed log fundamentals start from scratch • Any laptop/PC with JDK 21 and Docker Desktop or local Apache Kafka 3.8+ KRaft binary archive • No prior Kafka industry experience required — Module 1 starts from environment setup, core mental models, and plain-English fundamentals. • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards. • Commitment to attend live Weekday or Weekend classes, complete daily hands-on lab checklists, and practice topic-wise Mock Tests & AI Interviews.

2. DETAILED ZERO-TO-EXPERT MODULE, TOPIC & SUBTOPIC SYLLABUS (16 MODULES • 42 LIVE TOPICS • 336+ SUBTOPICS)

Progressive Zero-to-Expert Mastery Stages: Modules 1–4: Stage 1: Day-1 Foundations & Core Mental Models (8 Topics / 64 Subtopics) • Modules 5–8: Stage 2: Applied Engineering, Data & API Workflows (8 Topics / 64 Subtopics) • Modules 9–12: Stage 3: Internals, Concurrency, Security & Performance (12 Topics / 96 Subtopics) • Modules 13–16: Stage 4: Production Architecture, Testing, CI/CD & Expert Mastery (14 Topics / 112 Subtopics)

MODULE 1 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 1: Distributed Commit Logs, Zero-Copy PageCache & Apache Kafka KRaft Architecture

Understand why append-only immutable commit logs outperform traditional AMQP queues at scale, how sendfile() zero-copy I/O saturates NICs, and how KRaft eliminates ZooKeeper.

Objectives: ✓ Apache Kafka Fundamentals — Immutable Commit Log, Sequential Disk I/O & Zero-Copy `sendfile()` — Core abstractions of events (key, value, timestamp, headers), why sequential append-only disk writes rival RAM speeds via OS PageCache, and how multiple independent consumers replay the same log. • ✓ Apache Kafka Architecture — Brokers, KRaft Controller Quorum, Partitions & `__cluster_metadata` — Inside a modern ZooKeeper-less Kafka cluster: KRaft Raft consensus voters, active controller metadata propagation, broker request/I/O thread pools, and partition segment files (`.log`, `.index`, `.timeindex`). • ✓ Build, verify, and ship Module Capstone: Event-Driven Order Backbone Throughput & Retention Sizing Blueprint.

1.1 Apache Kafka Fundamentals — Immutable Commit Log, Sequential Disk I/O & Zero-Copy `sendfile()`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Immutable append-only commit log & OS PageCache sequential I/O • Core Syntax, CLI & Production APIs: ProducerRecord<K, V>, ConsumerRecord<K, V>, FileChannel.transferTo() zero-copy • Deep-Dive Mechanics: Linux sendfile() zero-copy & Event key-value-timestamp-headers • Advanced Patterns: Multi-subscriber log replay & Sparse offset memory-mapped index • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Constructing an Immutable Kafka Event Envelope in Java • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Apache Kafka Fundamentals

Purpose: To understand and apply Apache Kafka Fundamentals — Immutable Commit Log, Sequential Disk I/O & Zero-Copy `sendfile()` in real Kafka systems — specifically: Core abstract...

What Topic Is Next → Next Topic 1.2: Apache Kafka Architecture — Brokers, KRaft Controller Quorum, Partitions & `__cluster_metadata`

1.2 Apache Kafka Architecture — Brokers, KRaft Controller Quorum, Partitions & `__cluster_metadata`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: KRaft Raft consensus quorum & __cluster_metadata internal topic • Core Syntax, CLI & Production APIs: AdminClient.describeCluster(), DescribeClusterResult, process.roles=broker,controller • Deep-Dive Mechanics: Network vs I/O broker thread pools & Segment .log and .index files • Advanced Patterns: Odd-numbered controller voters (3 or 5) & Broker.rack fault-domain placement • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Inspecting KRaft Quorum & Cluster Topology via `AdminClient` • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Apache Kafka Architecture

Purpose: To understand and apply Apache Kafka Architecture — Brokers, KRaft Controller Quorum, Partitions & `__cluster_metadata` in real Kafka systems — specifically: Inside a mod...

What Topic Is Next → Module 1 Capstone (Event-Driven Order Backbone Throughput & Retention Sizing Blueprint) → Next Topic 2.1: Apache Kafka Use Cases — Transactional Outbox, Event Sourcing, CQRS & Real-Time Telemetry

Module 1 Hands-On Capstone Project (100 Marks): Event-Driven Order Backbone Throughput & Retention Sizing Blueprint — Design a 6-broker KRaft cluster blueprint capable of ingesting 250,000 order events/sec (1 KB payload) with 7-day retention, 3x replication, and zero ZooKeeper dependencies.

MODULE 2 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 2: Enterprise Event-Streaming Patterns & Production KRaft Cluster Bootstrap

Map real-world event-sourcing, CDC, log aggregation, and CQRS workloads onto Kafka, then format storage UUIDs and launch production KRaft brokers.

Objectives: ✓ Apache Kafka Use Cases — Transactional Outbox, Event Sourcing, CQRS & Real-Time Telemetry — Architectural patterns where Kafka shines (decoupled microservice event buses, Outbox + CDC avoiding dual writes, IoT telemetry ingestion, fraud stream scoring) and anti-use-cases (millions of ephemeral per-user topics). • ✓ Installing & Bootstrapping Apache Kafka in KRaft Mode (`kafka-storage.sh` & Listener Security) — Step-by-step KRaft setup: generating a cluster UUID, formatting `log.dirs` (`meta.properties`), understanding `listeners` vs `advertised.listeners`, and tuning Linux `vm.swappiness=1` & `ulimit -n 100000`. • ✓ Build, verify, and ship Module Capstone: 3-Node Local KRaft Cluster Bootstrap & OS Kernel Readiness Verifier.

2.1 Apache Kafka Use Cases — Transactional Outbox, Event Sourcing, CQRS & Real-Time Telemetry

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Transactional Outbox pattern & Dual-write anti-pattern prevention • Core Syntax, CLI & Production APIs: Transactional Outbox Table -> CDC -> Compacted & Time-Retained Kafka Topics • Deep-Dive Mechanics: Event Sourcing & CQRS & Per-entity partition key ordering • Advanced Patterns: High-volume telemetry buffering & Backpressure decoupling • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Deterministic Partition Routing by Domain Aggregate Key (`Murmur2`) • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Apache Kafka Use Cases

Purpose: To understand and apply Apache Kafka Use Cases — Transactional Outbox, Event Sourcing, CQRS & Real-Time Telemetry in real Kafka systems — specifically: Architectural patt...

What Topic Is Next → Next Topic 2.2: Installing & Bootstrapping Apache Kafka in KRaft Mode (`kafka-storage.sh` & Listener Security)

2.2 Installing & Bootstrapping Apache Kafka in KRaft Mode (`kafka-storage.sh` & Listener Security)

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Kafka-storage.sh random-uuid & format & Meta.properties cluster.id guard • Core Syntax, CLI & Production APIs: kafka-storage.sh random-uuid, kafka-storage.sh format -t <UUID> -c server.properties, kafka-... • Deep-Dive Mechanics: Listeners vs advertised.listeners & Vm.swappiness=1 PageCache protection • Advanced Patterns: XFS / ext4 noatime mount flag & Modest 4-6GB JVM G1GC heap sizing • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Validating Production KRaft Broker Properties Programmatically • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Installing

Purpose: To understand and apply Installing & Bootstrapping Apache Kafka in KRaft Mode (`kafka-storage.sh` & Listener Security) in real Kafka systems — specifically: Step-by-step...

What Topic Is Next → Module 2 Capstone (3-Node Local KRaft Cluster Bootstrap & OS Kernel Readiness Verifier) → Next Topic 3.1: Kafka Topics Deep Dive — Partitions, Log Segments, Retention Policies & Key Compaction

Module 2 Hands-On Capstone Project (100 Marks): 3-Node Local KRaft Cluster Bootstrap & OS Kernel Readiness Verifier — Automate formatting and validating a 3-node combined/isolated KRaft cluster configuration with deterministic cluster UUIDs, advertised listeners, and JVM G1GC tuning.

MODULE 3 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 3: Kafka Topics, Segment Lifecycle, Log Compaction & Administrative Provisioning

Master partition parallelism math, active vs closed log segments, delete vs compact cleanup policies, and declarative topic creation via CLI and AdminClient.

Objectives: ✓ Kafka Topics Deep Dive — Partitions, Log Segments, Retention Policies & Key Compaction — How topics split into ordered immutable partitions, how active segments roll at `segment.bytes` / `segment.ms`, and how log compaction retains the latest value per key while purging keys marked with a `null`-payload tombstone. • ✓ Creating & Managing Kafka Topics (`kafka-topics.sh` & Idempotent Java `AdminClient`) — Creating, describing, and altering topics safely in production using `kafka-topics.sh --bootstrap-server` and `Admin.createTopics()`, plus why `auto.create.topics.enable=false` is mandatory in enterprise clusters. • ✓ Build, verify, and ship Module Capstone: Automated Topic Governance & Schema-Safe Provisioning Engine.

3.1 Kafka Topics Deep Dive — Partitions, Log Segments, Retention Policies & Key Compaction

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Partition-level strict ordering & Active vs rolled log segments • Core Syntax, CLI & Production APIs: TopicConfig.CLEANUP_POLICY_CONFIG, CLEANUP_POLICY_COMPACT, RETENTION_MS_CONFIG, SEGMENT_BYTE... • Deep-Dive Mechanics: Cleanup.policy=delete vs compact & Null-value compaction tombstones • Advanced Patterns: Delete.retention.ms tombstone window & Irreversible partition count increase • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Building a Compacted + Time-Bounded Hybrid Topic Spec in Java • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Kafka Topics Deep Dive

Purpose: To understand and apply Kafka Topics Deep Dive — Partitions, Log Segments, Retention Policies & Key Compaction in real Kafka systems — specifically: How topics split into...

What Topic Is Next → Next Topic 3.2: Creating & Managing Kafka Topics (`kafka-topics.sh` & Idempotent Java `AdminClient`)

3.2 Creating & Managing Kafka Topics (`kafka-topics.sh` & Idempotent Java `AdminClient`)

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Kafka-topics.sh --bootstrap-server & Admin.createTopics() idempotency • Core Syntax, CLI & Production APIs: kafka-topics.sh --create --topic <name> --partitions 12 --replication-factor 3, Admin.create... • Deep-Dive Mechanics: Auto.create.topics.enable=false rule & TopicExistsException safe handling • Advanced Patterns: Kafka-configs.sh dynamic overrides & Under-replicated partition inspection • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Idempotent Topic Provisioner Handling `TopicExistsException` • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Creating

Purpose: To understand and apply Creating & Managing Kafka Topics (`kafka-topics.sh` & Idempotent Java `AdminClient`) in real Kafka systems — specifically: Creating, describing, a...

What Topic Is Next → Module 3 Capstone (Automated Topic Governance & Schema-Safe Provisioning Engine) → Next Topic 4.1: Kafka Topic Replication — Leaders, Followers, ISR, High Watermark & Leader Epoch

Module 3 Hands-On Capstone Project (100 Marks): Automated Topic Governance & Schema-Safe Provisioning Engine — Build a Java `AdminClient` provisioning tool that creates time-retained event topics and compacted entity-snapshot topics while rejecting unsafe replication factors or partition reductions.

MODULE 4 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 4: ISR Replication, High Watermark Durability & Producer Internals

Guarantee zero data loss with In-Sync Replicas (ISR), Leader Epochs, and High Watermarks, and trace the Java Producer RecordAccumulator, batching, compression, and acks pipeline.

Objectives: ✓ Kafka Topic Replication — Leaders, Followers, ISR, High Watermark & Leader Epoch — How follower brokers pull-replicate from partition leaders, when slow followers drop out of the In-Sync Replica (`ISR`) set (`replica.lag.time.max.ms`), why consumers only read up to the High Watermark (`HW`), and why Leader Epoch prevents log truncation bugs. • ✓ Apache Kafka Producer Architecture — Serializer, Partitioner, `RecordAccumulator` & Idempotence — Tracing `KafkaProducer.send()` across Interceptor -> Serializer -> Partitioner (Sticky Partitioner for keyless records!) -> `RecordAccumulator` `ProducerBatch` deque -> `Sender` network I/O thread, plus PID + Sequence Number idempotence. • ✓ Build, verify, and ship Module Capstone: Zero-Data-Loss Financial Ledger Producer & Durability Simulator.

4.1 Kafka Topic Replication — Leaders, Followers, ISR, High Watermark & Leader Epoch

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Leader & Follower pull replication & In-Sync Replicas (ISR) membership • Core Syntax, CLI & Production APIs: replication.factor=3, min.insync.replicas=2, unclean.leader.election.enable=false, replica.l... • Deep-Dive Mechanics: High Watermark (HW) visibility & Leader Epoch log-divergence safety • Advanced Patterns: RF=3 + min.insync.replicas=2 golden rule & Unclean.leader.election.enable=false • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Verifying Golden Durability Quorum (`RF=3`, `minISR=2`, `acks=all`) • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Kafka Topic Replication

Purpose: To understand and apply Kafka Topic Replication — Leaders, Followers, ISR, High Watermark & Leader Epoch in real Kafka systems — specifically: How follower brokers pull-r...

What Topic Is Next → Next Topic 4.2: Apache Kafka Producer Architecture — Serializer, Partitioner, `RecordAccumulator` & Idempotence

4.2 Apache Kafka Producer Architecture — Serializer, Partitioner, `RecordAccumulator` & Idempotence

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Thread-safe singleton KafkaProducer & Sticky Partitioner for null keys • Core Syntax, CLI & Production APIs: ProducerConfig.ACKS_CONFIG="all", ENABLE_IDEMPOTENCE_CONFIG=true, LINGER_MS_CONFIG, BATCH_SI... • Deep-Dive Mechanics: RecordAccumulator ProducerBatch memory & Linger.ms & batch.size trade-off • Advanced Patterns: Enable.idempotence PID + SeqNum & Max.in.flight.requests.per.connection<=5 • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Production-Hardened Idempotent `KafkaProducer` Configuration • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Apache Kafka Producer Architecture

Purpose: To understand and apply Apache Kafka Producer Architecture — Serializer, Partitioner, `RecordAccumulator` & Idempotence in real Kafka systems — specifically: Tracing `Kaf...

What Topic Is Next → Module 4 Capstone (Zero-Data-Loss Financial Ledger Producer & Durability Simulator) → Next Topic 5.1: Sending Data to Kafka Topics — CLI Key Parsing, Sync vs Async Delivery & `delivery.timeout.ms`

Module 4 Hands-On Capstone Project (100 Marks): Zero-Data-Loss Financial Ledger Producer & Durability Simulator — Configure and benchmark a financial ledger producer pipeline that guarantees durability across single-broker crashes while maximizing batch compression efficiency.

MODULE 5 OF 16 • STAGE 2 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 5: CLI Ingestion, Delivery Semantics & Java Producer Engineering

Compare fire-and-forget, synchronous, and asynchronous send modes, test ingestion via `kafka-console-producer.sh`, and write clean, leak-free Java 21 producer clients.

Objectives: ✓ Sending Data to Kafka Topics — CLI Key Parsing, Sync vs Async Delivery & `delivery.timeout.ms` — Using `kafka-console-producer.sh` with `--property parse.key=true --property key.separator=:`, understanding why calling `producer.send(rec).get()` on every message kills batching throughput by 100x, and bounding end-to-end latency with `delivery.timeout.ms`. • ✓ Creating a Kafka Producer in Java — Try-With-Resources, `StringSerializer` & Transactional Sends — Writing idiomatic Java 21 `KafkaProducer<String, String>` code with explicit `StringSerializer` classes, deterministic keys, atomic multi-partition transactions (`transactional.id`, `initTransactions`, `commitTransaction`), and clean resource closure. • ✓ Build, verify, and ship Module Capstone: High-Throughput Async Telemetry Producer with Graceful SIGTERM Drain.

5.1 Sending Data to Kafka Topics — CLI Key Parsing, Sync vs Async Delivery & `delivery.timeout.ms`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Parse.key=true & key.separator CLI & Sync .get() batch-killing anti-pattern • Core Syntax, CLI & Production APIs: kafka-console-producer.sh --property parse.key=true --property key.separator=:, producer.sen... • Deep-Dive Mechanics: Async RecordAccumulator enqueue & Delivery.timeout.ms end-to-end bound • Advanced Patterns: Max.block.ms backpressure guard & SerializationException non-retriable • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Validating Producer Timeout Invariant (`delivery.timeout.ms >= linger.ms + request.timeout.ms`) • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Sending Data to Kafka Topics

Purpose: To understand and apply Sending Data to Kafka Topics — CLI Key Parsing, Sync vs Async Delivery & `delivery.timeout.ms` in real Kafka systems — specifically: Using `kafka-...

What Topic Is Next → Next Topic 5.2: Creating a Kafka Producer in Java — Try-With-Resources, `StringSerializer` & Transactional Sends

5.2 Creating a Kafka Producer in Java — Try-With-Resources, `StringSerializer` & Transactional Sends

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: KafkaProducer<K, V> generic client & StringSerializer / Avro / Protobuf • Core Syntax, CLI & Production APIs: new KafkaProducer<>(props), StringSerializer.class.getName(), initTransactions(), beginTrans... • Deep-Dive Mechanics: Try-with-resources close() flush & Transactional.id zombie fencing • Advanced Patterns: Producer.epoch monotonic bump & AbortTransaction() atomic rollback • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Java 21 `KafkaProducer` with `MockProducer` Deterministic Verification • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Creating a Kafka Producer in Java

Purpose: To understand and apply Creating a Kafka Producer in Java — Try-With-Resources, `StringSerializer` & Transactional Sends in real Kafka systems — specifically: Writing idi...

What Topic Is Next → Module 5 Capstone (High-Throughput Async Telemetry Producer with Graceful SIGTERM Drain) → Next Topic 6.1: Kafka Producer Callbacks — `RecordMetadata`, Retriable vs Non-Retriable Errors & I/O Thread Safety

Module 5 Hands-On Capstone Project (100 Marks): High-Throughput Async Telemetry Producer with Graceful SIGTERM Drain — Build a Java 21 telemetry producer service that ingests 50,000 keyed JSON/Avro events asynchronously, enforces bounded memory backpressure (`max.block.ms`), and flushes all in-flight batches...

MODULE 6 OF 16 • STAGE 2 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 6: Async Producer Callbacks, Consumer Groups & Cooperative Rebalancing

Capture `RecordMetadata` offsets and non-retriable exceptions in non-blocking callbacks, and master Consumer Group partition assignment and `CooperativeStickyAssignor` zero-stop-the-world rebalancing.

Objectives: ✓ Kafka Producer Callbacks — `RecordMetadata`, Retriable vs Non-Retriable Errors & I/O Thread Safety — Inspecting partition, offset, and timestamp via `RecordMetadata` inside `Callback`, distinguishing retriable errors (`LeaderNotAvailableException`) from fatal ones (`RecordTooLargeException`), and keeping callbacks non-blocking. • ✓ Kafka Consumers & Consumer Groups — Partition Ownership, Heartbeats & `CooperativeStickyAssignor` — How `group.id` turns competing consumers into a scalable horizontal worker pool while distinct `group.id`s act as independent pub-sub subscribers, `GroupCoordinator` & `__consumer_offsets`, and `session.timeout.ms` vs `max.poll.interval.ms`. • ✓ Build, verify, and ship Module Capstone: Dead-Letter Routing Producer Callback & Zero-Pause Cooperative Consumer Fleet.

6.1 Kafka Producer Callbacks — `RecordMetadata`, Retriable vs Non-Retriable Errors & I/O Thread Safety [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Callback(RecordMetadata, Exception) & Sender I/O thread non-blocking rule • Core Syntax, CLI & Production APIs: producer.send(record, (RecordMetadata meta, Exception ex) -> ...), meta.partition(), meta.of... • Deep-Dive Mechanics: RetriableException auto-retry & RecordTooLargeException DLQ routing • Advanced Patterns: Per-partition callback ordering & SerializedKeySize & serializedValueSize • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Asynchronous Producer Callback Capturing Partition & Offset • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Kafka Producer Callbacks

Purpose: To understand and apply Kafka Producer Callbacks — `RecordMetadata`, Retriable vs Non-Retriable Errors & I/O Thread Safety in real Kafka systems — specifically: Inspectin...

What Topic Is Next → Next Topic 6.2: Kafka Consumers & Consumer Groups — Partition Ownership, Heartbeats & `CooperativeStickyAssignor`

6.2 Kafka Consumers & Consumer Groups — Partition Ownership, Heartbeats & `CooperativeStickyAssignor` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: 1-partition-to-at-most-1-group-member & Idle consumers when C > P • Core Syntax, CLI & Production APIs: ConsumerConfig.GROUP_ID_CONFIG, CooperativeStickyAssignor, GROUP_INSTANCE_ID_CONFIG, MAX_POL... • Deep-Dive Mechanics: Heartbeat thread vs max.poll.interval.ms & CooperativeStickyAssignor incremental • Advanced Patterns: Group.instance.id static membership & __consumer_offsets internal topic (50p) • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Simulating Consumer Group Partition Allocation (`C <= P` vs Idle `C > P`) • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Kafka Consumers

Purpose: To understand and apply Kafka Consumers & Consumer Groups — Partition Ownership, Heartbeats & `CooperativeStickyAssignor` in real Kafka systems — specifically: How `group...

What Topic Is Next → Module 6 Capstone (Dead-Letter Routing Producer Callback & Zero-Pause Cooperative Consumer Fleet) → Next Topic 7.1: Kafka Console Consumer — Key/Header Formatting, `--from-beginning` & `read_committed` Isolation

Module 6 Hands-On Capstone Project (100 Marks): Dead-Letter Routing Producer Callback & Zero-Pause Cooperative Consumer Fleet — Build an async producer callback handler that routes failed records to a DLQ without blocking `Sender` I/O, paired with a static-member `CooperativeStickyAssignor` consumer fleet.

MODULE 7 OF 16 • STAGE 2 • 11 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 7: CLI Stream Debugging, Consumer Lag Forensics & Offset Reset Operations

Inspect live headers, keys, partitions, and uncommitted/committed records with `kafka-console-consumer.sh`, and diagnose lag or safely rewind offsets using `kafka-consumer-groups.sh`.

Objectives: ✓ Kafka Console Consumer — Key/Header Formatting, `--from-beginning` & `read_committed` Isolation — Inspecting live topics cleanly with `DefaultMessageFormatter` properties (`print.key=true`, `print.offset=true`, `print.timestamp=true`, `print.headers=true`), understanding `auto.offset.reset=earliest|latest`, and filtering aborted transactions via `--isolation-level read_committed`. • ✓ Kafka Consumer Group CLI — `LAG` Diagnostics, Hot-Key Skew & Safe `--reset-offsets --dry-run` — Using `kafka-consumer-groups.sh --describe` to inspect `CURRENT-OFFSET`, `LOG-END-OFFSET`, `LAG`, and `CONSUMER-ID`, spotting hot-partition key skew when 1 partition lags by millions while 11 sit at `0`, and safely rewinding offsets. • ✓ Build, verify, and ship Module Capstone: Incident Incident-Response Consumer Lag Auditor & Dry-Run Offset Rewind Runbook.

7.1 Kafka Console Consumer — Key/Header Formatting, `--from-beginning` & `read_committed` Isolation [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Print.key & print.headers formatter & --from-beginning vs committed offsets • Core Syntax, CLI & Production APIs: kafka-console-consumer.sh --bootstrap-server ... --property print.key=true --isolation-level... • Deep-Dive Mechanics: Auto.offset.reset trigger condition & --isolation-level read_committed • Advanced Patterns: Last Stable Offset (LSO) visibility & --partition & --offset targeted seek • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Building Exact `kafka-console-consumer.sh` Forensic Flags • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Kafka Console Consumer

Purpose: To understand and apply Kafka Console Consumer — Key/Header Formatting, `--from-beginning` & `read_committed` Isolation in real Kafka systems — specifically: Inspecting l...

What Topic Is Next → Next Topic 7.2: Kafka Consumer Group CLI — `LAG` Diagnostics, Hot-Key Skew & Safe `--reset-offsets --dry-run`

7.2 Kafka Consumer Group CLI — `LAG` Diagnostics, Hot-Key Skew & Safe `--reset-offsets --dry-run` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: LAG = LOG-END-OFFSET - CURRENT-OFFSET & Hot-key single-partition skew diagnosis • Core Syntax, CLI & Production APIs: kafka-consumer-groups.sh --describe --group <id>, --reset-offsets --to-datetime <ISO> --dry-... • Deep-Dive Mechanics: Empty group state prerequisite for reset & --dry-run before --execute discipline • Advanced Patterns: --to-datetime ISO-8601 time travel & Offsets.retention.minutes guard • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Detecting Hot-Partition Lag Skew Across Consumer Group Partitions • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Kafka Consumer Group CLI

Purpose: To understand and apply Kafka Consumer Group CLI — `LAG` Diagnostics, Hot-Key Skew & Safe `--reset-offsets --dry-run` in real Kafka systems — specifically: Using `kafka-c...

What Topic Is Next → Module 7 Capstone (Incident Incident-Response Consumer Lag Auditor & Dry-Run Offset Rewind Runbook) → Next Topic 8.1: Creating a Kafka Consumer in Java — Poll Loop, `enable.auto.commit=false` & `WakeupException`

Module 7 Hands-On Capstone Project (100 Marks): Incident Incident-Response Consumer Lag Auditor & Dry-Run Offset Rewind Runbook — Create an operational CLI + Java AdminClient diagnostic utility that detects skewed consumer-group partition lag (hot keys) and validates safe `--dry-run` offset rewinds

after a bad downstre...

MODULE 8 OF 16 • STAGE 2 • 11 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 8: Java Consumer Poll Loops, Manual Offset Commits & Multi-Cluster MirrorMaker 2

Write thread-safe Java consumer poll loops with manual at-least-once offset commits and `ConsumerRebalanceListener`, and architect multi-region disaster recovery with MirrorMaker 2.

Objectives: ✓ Creating a Kafka Consumer in Java — Poll Loop, `enable.auto.commit=false` & `WakeupException` — Why `KafkaConsumer` is deliberately **NOT thread-safe**, why `enable.auto.commit=true` loses messages if a worker crashes after `poll()` auto-commits an unprocessed batch, and how `commitAsync` + `onPartitionsRevoked` `commitSync` delivers clean at-least-once processing. • ✓ Multi-Cluster Kafka Architectures — MirrorMaker 2, Active-Active Replication & Stretch Clusters — Designing geo-distributed topologies: Hub-and-Spoke edge aggregation, Active-Passive DR, Active-Active bi-directional replication with MirrorMaker 2 topic prefixing (`us-east.orders`) to prevent infinite loops, and `RemoteClusterUtils` offset translation. • ✓ Build, verify, and ship Module Capstone: At-Least-Once Idempotent Java Consumer & Active-Passive DR Failover Blueprint.

8.1 Creating a Kafka Consumer in Java — Poll Loop, `enable.auto.commit=false` & `WakeupException`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Single-threaded KafkaConsumer rule & Enable.auto.commit=false safety • Core Syntax, CLI & Production APIs: KafkaConsumer.poll(Duration), ENABLE_AUTO_COMMIT_CONFIG=false, commitSync(), ConsumerRebalan... • Deep-Dive Mechanics: Commit offset = lastOffset + 1 & ConsumerRebalanceListener.onPartitionsRevoked • Advanced Patterns: Consumer.wakeup() clean shutdown & Downstream idempotent deduplication • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Deterministic Java `MockConsumer` Poll & Manual `commitSync` Loop • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Creating a Kafka Consumer in Java

Purpose: To understand and apply Creating a Kafka Consumer in Java — Poll Loop, `enable.auto.commit=false` & `WakeupException` in real Kafka systems — specifically: Why `KafkaCons...

What Topic Is Next → Next Topic 8.2: Multi-Cluster Kafka Architectures — MirrorMaker 2, Active-Active Replication & Stretch Clusters

8.2 Multi-Cluster Kafka Architectures — MirrorMaker 2, Active-Active Replication & Stretch Clusters

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: MirrorMaker 2 (MM2) on Kafka Connect & Active-Passive vs Active-Active WAN • Core Syntax, CLI & Production APIs: MirrorSourceConnector, MirrorCheckpointConnector, MirrorHeartbeatConnector, RemoteClusterUti... • Deep-Dive Mechanics: Source-alias topic loop prevention & MirrorCheckpointConnector offset sync • Advanced Patterns: Stretch Cluster synchronous RTT cost & RPO / RTO disaster recovery SLAs • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: MirrorMaker 2 Replication Policy Loop-Prevention Verification • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Multi-Cluster Kafka Architectures

Purpose: To understand and apply Multi-Cluster Kafka Architectures — MirrorMaker 2, Active-Active Replication & Stretch Clusters in real Kafka systems — specifically: Designing ge...

What Topic Is Next → Module 8 Capstone (At-Least-Once Idempotent Java Consumer & Active-Passive DR Failover Blueprint) → Next Topic 9.1: Kafka Connect & Debezium CDC — Distributed Workers, Internal Compacted Topics, SMTs & DLQ

Module 8 Hands-On Capstone Project (100 Marks): At-Least-Once Idempotent Java Consumer & Active-Passive DR Failover Blueprint — Implement a production Java `KafkaConsumer` poll loop with `enable.auto.commit=false`, rebalance-safe offset commits, `wakeup()` shutdown, and a MirrorMaker 2 cross-region DR config.

MODULE 9 OF 16 • STAGE 3 • 11 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 9: Kafka Connect CDC Pipelines & Stateful Kafka Streams Topologies

Stream database binlogs/WALs code-free with Kafka Connect + Debezium Single Message Transforms, and build embedded stateful stream-processing microservices using the Kafka Streams DSL.

Objectives: ✓ Kafka Connect & Debezium CDC — Distributed Workers, Internal Compacted Topics, SMTs & DLQ — Declarative Source & Sink connectors running in fault-tolerant Distributed Mode (`group.id`), the 3 compacted internal topics (`config.storage.topic`, `offset.storage.topic`, `status.storage.topic`), Debezium WAL/binlog Change Data Capture, Single Message Transforms (SMTs), and Dead Letter Queues. • ✓ Kafka Stream Processing — Embedded `KafkaStreams` Topologies, `exactly_once_v2` & `TopologyTestDriver` — Why Kafka Streams runs as an embedded Java library inside standard microservices (zero separate Spark/Flink processing cluster required!), DAG stream topologies (`StreamsBuilder`), StreamTasks mapped 1:1 to input partitions, and `exactly_once_v2` atomic read-state-write commits. • ✓ Build, verify, and ship Module Capstone: Debezium Outbox CDC Connector + Real-Time Fraud Aggregation Kafka Streams Service.

9.1 Kafka Connect & Debezium CDC — Distributed Workers, Internal Compacted Topics, SMTs & DLQ

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Source vs Sink Connectors & Tasks & Distributed Mode 3 compacted topics • Core Syntax, CLI & Production APIs: REST POST /connectors, SourceConnector / SinkConnector, Single Message Transforms (transform... • Deep-Dive Mechanics: Debezium WAL/binlog log-based CDC & Single Message Transforms (SMTs) • Advanced Patterns: Errors.tolerance=all + Connect DLQ & Schema Registry Avro/Protobuf • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Validating Distributed Kafka Connect Debezium + DLQ JSON Spec • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Kafka Connect

Purpose: To understand and apply Kafka Connect & Debezium CDC — Distributed Workers, Internal Compacted Topics, SMTs & DLQ in real Kafka systems — specifically: Declarative Source...

What Topic Is Next → Next Topic 9.2: Kafka Stream Processing — Embedded `KafkaStreams` Topologies, `exactly_once_v2` & `TopologyTestDriver`

9.2 Kafka Stream Processing — Embedded `KafkaStreams` Topologies, `exactly_once_v2` & `TopologyTestDriver` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Embedded library (no external cluster) & StreamTask 1:1 input partition unit • Core Syntax, CLI & Production APIs: StreamsBuilder, KStream<K, V>, KafkaStreams, StreamsConfig.PROCESSING_GUARANTEE_CONFIG="exac... • Deep-Dive Mechanics: MapValues vs key-changing repartition & Processing.guarantee=exactly_once_v2 • Advanced Patterns: TopologyTestDriver fast unit testing & Num.stream.threads vertical scaling • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Kafka Streams Topology Filtering High-Value Payments (`mapValues` + `filter`) • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Kafka Stream Processing

Purpose: To understand and apply Kafka Stream Processing — Embedded `KafkaStreams` Topologies, `exactly_once_v2` & `TopologyTestDriver` in real Kafka systems — specifically: Why K...

What Topic Is Next → Module 9 Capstone (Debezium Outbox CDC Connector + Real-Time Fraud Aggregation Kafka Streams Service) → Next Topic 10.1: Stream Processing Concepts — `KStream` vs `KTable` Duality, Event-Time Windows & RocksDB State

Module 9 Hands-On Capstone Project (100 Marks): Debezium Outbox CDC Connector + Real-Time Fraud Aggregation Kafka Streams Service — Design a Distributed Kafka Connect Debezium PostgreSQL Outbox connector config with DLQ error tolerance, feeding a Java `KafkaStreams` topology that flags accounts exceeding 5 high-value tra...

MODULE 10 OF 16 • STAGE 3 • 11 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 10: KStream/KTable Duality, Event-Time Windowing & Production JMX Observability

Master stream-table duality (`KStream`, `KTable`, `GlobalKTable`), event-time windows with grace periods, RocksDB standby replicas, and zero-blindspot JMX/Prometheus broker & consumer monitoring.

Objectives: ✓ Stream Processing Concepts — `KStream` vs `KTable` Duality, Event-Time Windows & RocksDB State — Stream-Table Duality (`KStream` unbounded facts vs `KTable` latest-value mutable state vs `GlobalKTable` full-replica dimension joins), Event Time vs Ingestion/Processing Time, Tumbling/Hopping/Sliding/Session windows with `.grace()`, and RocksDB off-heap state stores backed by compacted changelog topics. • ✓ Kafka Monitoring & SRE Observability — JMX Exporter, Critical Broker MBeans & Lag Alerting — Exposing Kafka JMX MBeans to Prometheus & Grafana, monitoring P0 page-worthy alerts (`OfflinePartitionsCount > 0`, `ActiveControllerCount != 1`, sustained `UnderReplicatedPartitions > 0`), tracking `RequestHandlerAvgIdlePercent` / `NetworkProcessorAvgIdlePercent`, and time-based consumer lag SLAs. • ✓ Build, verify, and ship Module Capstone: Windowed Revenue Aggregator (`RocksDB` + `StandbyReplicas`) & SRE JMX Alerting Suite.

10.1 Stream Processing Concepts — `KStream` vs `KTable` Duality, Event-Time Windows & RocksDB State [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: KStream vs KTable stream-table duality & GlobalKTable non-co-partitioned join • Core Syntax, CLI & Production APIs: KStream, KTable, GlobalKTable, TimeWindows.ofSizeAndGrace(), Materialized.as(), num.standby.... • Deep-Dive Mechanics: Event Time vs Processing Time & Tumbling / Hopping / Session windows • Advanced Patterns: Window grace period for late events & RocksDB + compacted changelog + standby • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Stateful `KTable` Word/Event Counter Verified with `TopologyTestDriver` • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Stream Processing Concepts

Purpose: To understand and apply Stream Processing Concepts — `KStream` vs `KTable` Duality, Event-Time Windows & RocksDB State in real Kafka systems — specifically: Stream-Table...

What Topic Is Next → Next Topic 10.2: Kafka Monitoring & SRE Observability — JMX Exporter, Critical Broker MBeans & Lag Alerting

10.2 Kafka Monitoring & SRE Observability — JMX Exporter, Critical Broker MBeans & Lag Alerting [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: OfflinePartitionsCount == 0 P0 alert & UnderReplicatedPartitions ISR health • Core Syntax, CLI & Production APIs: kafka.controller:type=KafkaController,name=OfflinePartitionsCount, UnderReplicatedPartitions... • Deep-Dive Mechanics: ActiveControllerCount == 1 invariant & RequestHandlerAvgIdlePercent > 0.3 • Advanced Patterns: ISR shrink/expand churn detection & Time-lag + commit-rate consumer SLAs • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Evaluating Kafka Cluster Golden JMX Health Signals in Java • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Kafka Monitoring

Purpose: To understand and apply Kafka Monitoring & SRE Observability — JMX Exporter, Critical Broker MBeans & Lag Alerting in real Kafka systems — specifically: Exposing Kafka JM...

What Topic Is Next → Module 10 Capstone (Windowed Revenue Aggregator (`RocksDB` + `StandbyReplicas`) & SRE JMX Alerting Suite) → Next Topic 11.1: KRaft Consensus Metadata Quorum (Zero ZooKeeper) & ISR Replication

Module 10 Hands-On Capstone Project (100 Marks): Windowed Revenue Aggregator (`RocksDB` + `StandbyReplicas`) & SRE JMX Alerting Suite — Build an event-time tumbling-window revenue aggregator backed by RocksDB changelog state and `num.standby.replicas=1`, plus a Prometheus/JMX health evaluator alerting on cluster degradation.

MODULE 11 OF 16 • STAGE 3 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

KRaft Internals, Idempotent/Transactional Producers & Partition Strategy

Production-grade Kafka Training engineering module covering KRaft Internals, Idempotent/Transactional Producers & Partition Strategy with hands-on labs and capstone build.

Objectives: ✓ KRaft Consensus Metadata Quorum (Zero ZooKeeper) & ISR Replication — Purpose: Understand leader election, high watermarks, and why acks=all + min.insync.replicas=2 prevents data loss. • ✓ Idempotent Producers (PID + Sequence Numbers) & EOS Transactions — Purpose: Eliminate duplicate messages during network retries and atomic consume-transform-produce flows. • ✓ Partition Key Sizing, Hot-Key Skew Mitigation & Log Compaction — Purpose:

Balance throughput across brokers while preserving strict per-customer event order. • ✓ Batching (linger.ms, batch.size) & LZ4/Zstd Compression Tuning — Purpose: Multiply Kafka throughput 5x while slashing cross-AZ network bandwidth bills.

11.1 KRaft Consensus Metadata Quorum (Zero ZooKeeper) & ISR Replication [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of ISR Replication • Integration with Idempotent Producers (PID + Sequence Numbers) & EOS Transactions • Live Debugging, Telemetry & Technical Interview Trade • What Is KRaft Consensus Metadata Quorum (Zero ZooKeeper) & ISR Replication & Why Industry Teams Use It • In Kafka Training, KRaft Consensus Metadata Quorum (Zero ZooKeeper) & ISR Replication exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply KRaft Consensus Metadata Quorum (Zero ZooKeeper) & ISR Replication in real Kafka systems — specifically: Understand leader election, high watermar...

What Topic Is Next → Next Topic 11.2: Idempotent Producers (PID + Sequence Numbers) & EOS Transactions

11.2 Idempotent Producers (PID + Sequence Numbers) & EOS Transactions [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of EOS Transactions • Integration with Partition Key Sizing, Hot • Live Debugging, Telemetry & Technical Interview Trade • What Is Idempotent Producers (PID + Sequence Numbers) & EOS Transactions & Why Industry Teams Use It • In Kafka Training, Idempotent Producers (PID + Sequence Numbers) & EOS Transactions exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Idempotent Producers (PID + Sequence Numbers) & EOS Transactions in real Kafka systems — specifically: Eliminate duplicate messages during network...

What Topic Is Next → Next Topic 11.3: Partition Key Sizing, Hot-Key Skew Mitigation & Log Compaction

11.3 Partition Key Sizing, Hot-Key Skew Mitigation & Log Compaction [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Hot • Integration with Batching (linger.ms, batch.size) & LZ4/Zstd Compression Tuning • Live Debugging, Telemetry & Technical Interview Trade • What Is Partition Key Sizing, Hot • In Kafka Training, Partition Key Sizing, Hot • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Partition Key Sizing, Hot-Key Skew Mitigation & Log Compaction in real Kafka systems — specifically: Balance throughput across brokers while prese...

What Topic Is Next → Next Topic 11.4: Batching (linger.ms, batch.size) & LZ4/Zstd Compression Tuning

11.4 Batching (linger.ms, batch.size) & LZ4/Zstd Compression Tuning [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of batch.size) • Integration with KRaft Consensus Metadata Quorum (Zero ZooKeeper) & ISR Replication • Live Debugging, Telemetry & Technical Interview Trade • What Is Batching (linger.ms, batch.size) & LZ4/Zstd Compression Tuning & Why Industry Teams Use It • In Kafka Training, Batching (linger.ms, batch.size) & LZ4/Zstd Compression Tuning exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Batching (linger.ms, batch.size) & LZ4/Zstd Compression Tuning in real Kafka systems — specifically: Multiply Kafka throughput 5x while slashing c...

What Topic Is Next → Module 11 Capstone (Zero-Data-Loss Financial Event Backbone on Apache Kafka) → Next Topic 12.1: CooperativeStickyAssignor vs Stop-the-World Eager Rebalances

Module 11 Hands-On Capstone Project (100 Marks): Zero-Data-Loss Financial Event Backbone on Apache Kafka — Design, implement, test, and document 'Zero-Data-Loss Financial Event Backbone on Apache Kafka' applying every production concept from Module 11 (KRaft Internals, Idempotent/Transactional Pr...

MODULE 12 OF 16 • STAGE 3 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Consumer Groups, Cooperative Rebalancing, DLQ Retries & Schema Registry

Production-grade Kafka Training engineering module covering Consumer Groups, Cooperative Rebalancing, DLQ Retries & Schema Registry with hands-on labs and capstone build.

Objectives: ✓ CooperativeStickyAssignor vs Stop-the-World Eager Rebalances — Purpose: Scale consumer pods up and down without freezing event processing across all partitions. • ✓ Manual Offset Commits, At-Least-Once Idempotency & Poison-Pill Isolation — Purpose: Ensure offsets commit only after durable DB writes and route bad payloads to DLQs. • ✓ Non-Blocking Retry Topics (retry-5s -> retry-1m -> DLT) — Purpose: Retry transient downstream API timeouts without blocking healthy partition traffic. • ✓ Confluent/Apicurio Schema Registry (Avro/Protobuf) Compatibility Modes — Purpose: Enforce BACKWARD/FULL schema rules in CI so producer deploys never crash consumers.

12.1 CooperativeStickyAssignor vs Stop-the-World Eager Rebalances [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Edge • Integration with Manual Offset Commits, At • Live Debugging, Telemetry & Technical Interview Trade • What Is CooperativeStickyAssignor vs Stop • In Kafka Training, CooperativeStickyAssignor vs Stop • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply CooperativeStickyAssignor vs Stop-the-World Eager Rebalances in real Kafka systems — specifically: Scale consumer pods up and down without freezin...

What Topic Is Next → Next Topic 12.2: Manual Offset Commits, At-Least-Once Idempotency & Poison-Pill Isolation

12.2 Manual Offset Commits, At-Least-Once Idempotency & Poison-Pill Isolation [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of At • Integration with Non • Live Debugging, Telemetry & Technical Interview Trade • What Is Manual Offset Commits, At • In Kafka Training, Manual Offset Commits, At • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Manual Offset Commits, At-Least-Once Idempotency & Poison-Pill Isolation in real Kafka systems — specifically: Ensure offsets commit only after du...

What Topic Is Next → Next Topic 12.3: Non-Blocking Retry Topics (retry-5s -> retry-1m -> DLT)

12.3 Non-Blocking Retry Topics (retry-5s -> retry-1m -> DLT) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Edge • Integration with Confluent/Apicurio Schema Registry (Avro/Protobuf) Compatibility Modes • Live Debugging, Telemetry & Technical Interview Trade • What Is Non • In Kafka Training, Non • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Non-Blocking Retry Topics (retry-5s -> retry-1m -> DLT) in real Kafka systems — specifically: Retry transient downstream API timeouts without bloc...

What Topic Is Next → Next Topic 12.4: Confluent/Apicurio Schema Registry (Avro/Protobuf) Compatibility Modes

12.4 Confluent/Apicurio Schema Registry (Avro/Protobuf) Compatibility Modes [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Edge • Integration with CooperativeStickyAssignor vs Stop • Live Debugging, Telemetry & Technical Interview Trade • What Is Confluent/Apicurio Schema Registry (Avro/Protobuf) Compatibility Modes & Why Industry Teams Use It • In Kafka Training, Confluent/Apicurio Schema Registry (Avro/Protobuf) Compatibility Modes exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Confluent/Apicurio Schema Registry (Avro/Protobuf) Compatibility Modes in real Kafka systems — specifically: Enforce BACKWARD/FULL schema rules in...

What Topic Is Next → Module 12 Capstone (Resilient Multi-Stage Retry & Schema-Governed Consumer Fleet) → Next Topic 13.1: Kafka Connect Distributed Workers, Source/Sink Connectors & SMTs

Module 12 Hands-On Capstone Project (100 Marks): Resilient Multi-Stage Retry & Schema-Governed Consumer Fleet — Design, implement, test, and document 'Resilient Multi-Stage Retry & Schema-Governed Consumer Fleet' applying every production concept from Module 12 (Consumer Groups, Cooperative Rebalancin...

MODULE 13 OF 16 • STAGE 4 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Kafka Connect, Debezium Change Data Capture (CDC) & Outbox Streaming

Production-grade Kafka Training engineering module covering Kafka Connect, Debezium Change Data Capture (CDC) & Outbox Streaming with hands-on labs and capstone build.

Objectives: ✓ Kafka Connect Distributed Workers, Source/Sink Connectors & SMTs — Purpose: Stream data between databases, S3, and Elasticsearch with config instead of custom glue code. • ✓ Log-Based Debezium CDC (WAL/Binlog) vs Polling Anti-Patterns — Purpose: Capture every row insert/update/delete in milliseconds with near-zero primary DB load. • ✓ Transactional Outbox Event Router Pattern — Purpose: Publish clean domain events atomically alongside microservice SQL commits. • ✓ Connector Secret Providers, Dead-Letter Sink Routing & Offset Resets — Purpose: Operate Connect clusters securely and recover broken connectors cleanly.

13.1 Kafka Connect Distributed Workers, Source/Sink Connectors & SMTs [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Source/Sink Connectors • Integration with Log • Live Debugging, Telemetry & Technical Interview Trade • What Is Kafka Connect Distributed Workers, Source/Sink Connectors & SMTs & Why Industry Teams Use It • In Kafka Training, Kafka Connect Distributed Workers, Source/Sink Connectors & SMTs exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Kafka Connect Distributed Workers, Source/Sink Connectors & SMTs in real Kafka systems — specifically: Stream data between databases, S3, and Elas...

What Topic Is Next → Next Topic 13.2: Log-Based Debezium CDC (WAL/Binlog) vs Polling Anti-Patterns

13.2 Log-Based Debezium CDC (WAL/Binlog) vs Polling Anti-Patterns [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Edge • Integration with Transactional Outbox Event Router Pattern • Live Debugging, Telemetry & Technical Interview Trade • What Is Log • In Kafka Training, Log • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Log-Based Debezium CDC (WAL/Binlog) vs Polling Anti-Patterns in real Kafka systems — specifically: Capture every row insert/update/delete in milli...

What Topic Is Next → Next Topic 13.3: Transactional Outbox Event Router Pattern

13.3 Transactional Outbox Event Router Pattern [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Edge • Integration with Connector Secret Providers, Dead • Live Debugging, Telemetry & Technical Interview Trade • What Is Transactional Outbox Event Router Pattern & Why Industry Teams Use It • In Kafka Training, Transactional Outbox Event Router Pattern exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Transactional Outbox Event Router Pattern in real Kafka systems — specifically: Publish clean domain events atomically alongside microservice SQL...

What Topic Is Next → Next Topic 13.4: Connector Secret Providers, Dead-Letter Sink Routing & Offset Resets

13.4 Connector Secret Providers, Dead-Letter Sink Routing & Offset Resets [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Dead • Integration with Kafka Connect Distributed Workers, Source/Sink Connectors & SMTs • Live Debugging, Telemetry & Technical Interview Trade • What Is Connector Secret Providers, Dead • In Kafka Training, Connector Secret Providers, Dead • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Connector Secret Providers, Dead-Letter Sink Routing & Offset Resets in real Kafka systems — specifically: Operate Connect clusters securely and r...

What Topic Is Next → Module 13 Capstone (Real-Time PostgreSQL-to-Kafka CDC & Search Sync Pipeline) → Next Topic 14.1: KStream vs KTable Duality, GlobalKTable & Stream-Table Joins

Module 13 Hands-On Capstone Project (100 Marks): Real-Time PostgreSQL-to-Kafka CDC & Search Sync Pipeline — Design, implement, test, and document 'Real-Time PostgreSQL-to-Kafka CDC & Search Sync Pipeline' applying every production concept from Module 13 (Kafka Connect, Debezium Change Data Capture...

MODULE 14 OF 16 • STAGE 4 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Stateful Kafka Streams (RocksDB, Windows) & Production Cluster Ops

Production-grade Kafka Training engineering module covering Stateful Kafka Streams (RocksDB, Windows) & Production Cluster Ops with hands-on labs and capstone build.

Objectives: ✓ KStream vs KTable Duality, GlobalKTable & Stream-Table Joins — Purpose: Enrich high-velocity click/payment events with customer tier metadata in real time. • ✓ Tumbling, Hopping & Session Windows + Watermarks for Late Events — Purpose: Detect 5-failed-card-attempts-in-2-minutes fraud bursts accurately even on delayed mobile events. • ✓ RocksDB State Stores, Changelog Topics & Standby Replicas — Purpose: Fail over stateful stream processors in seconds instead of replaying hours of state. • ✓ Consumer Lag Alerting, Cruise Control Rebalancing, mTLS/SASL & MirrorMaker 2 — Purpose: Monitor consumer lag SLAs, enforce ACL security, and replicate cross-region DR clusters.

14.1 KStream vs KTable Duality, GlobalKTable & Stream-Table Joins [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of GlobalKTable • Integration with Tumbling, Hopping & Session Windows + Watermarks for Late Events • Live Debugging, Telemetry & Technical Interview Trade • What Is KStream vs KTable Duality, GlobalKTable & Stream • In Kafka Training, KStream vs KTable Duality, GlobalKTable & Stream • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply KStream vs KTable Duality, GlobalKTable & Stream-Table Joins in real Kafka systems — specifically: Enrich high-velocity click/payment events with...

What Topic Is Next → Next Topic 14.2: Tumbling, Hopping & Session Windows + Watermarks for Late Events

14.2 Tumbling, Hopping & Session Windows + Watermarks for Late Events [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Hopping • Integration with RocksDB State Stores, Changelog Topics & Standby Replicas • Live Debugging, Telemetry & Technical Interview Trade • What Is Tumbling, Hopping & Session Windows + Watermarks for Late Events & Why Industry Teams Use It • In Kafka Training, Tumbling, Hopping & Session Windows + Watermarks for Late Events exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Tumbling, Hopping & Session Windows + Watermarks for Late Events in real Kafka systems — specifically: Detect 5-failed-card-attempts-in-2-minutes...

What Topic Is Next → Next Topic 14.3: RocksDB State Stores, Changelog Topics & Standby Replicas

14.3 RocksDB State Stores, Changelog Topics & Standby Replicas [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Changelog Topics • Integration with Consumer Lag Alerting, Cruise Control Rebalancing, mTLS/SASL & MirrorMaker 2 • Live Debugging, Telemetry & Technical Interview Trade • What Is RocksDB State Stores, Changelog Topics & Standby Replicas & Why Industry Teams Use It • In Kafka Training, RocksDB State Stores, Changelog Topics & Standby Replicas exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply RocksDB State Stores, Changelog Topics & Standby Replicas in real Kafka systems — specifically: Fail over stateful stream processors in seconds in...

What Topic Is Next → Next Topic 14.4: Consumer Lag Alerting, Cruise Control Rebalancing, mTLS/SASL & MirrorMaker 2

14.4 Consumer Lag Alerting, Cruise Control Rebalancing, mTLS/SASL & MirrorMaker 2 [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Cruise Control Rebalancing • Integration with KStream vs KTable Duality, GlobalKTable & Stream • Live Debugging, Telemetry & Technical Interview Trade • What Is Consumer Lag Alerting, Cruise Control Rebalancing, mTLS/SASL & MirrorMaker 2 & Why Industry Teams Use It • In Kafka Training, Consumer Lag Alerting, Cruise Control Rebalancing, mTLS/SASL & MirrorMaker 2 exists for one clea... • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Consumer Lag Alerting, Cruise Control Rebalancing, mTLS/SASL & MirrorMaker 2 in real Kafka systems — specifically: Monitor consumer lag SLAs, enfo...

What Topic Is Next → Module 14 Capstone (Real-Time Fraud Velocity & Sliding-Window Aggregator) → Next Topic 15.1: Schema Registry Evolution Rules (Backward, Forward & Full Compatibility)

Module 14 Hands-On Capstone Project (100 Marks): Real-Time Fraud Velocity & Sliding-Window Aggregator — Design, implement, test, and document 'Real-Time Fraud Velocity & Sliding-Window Aggregator' applying every production concept from Module 14 (Stateful Kafka Streams (RocksDB, Windows) & Pro...

MODULE 15 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Schema Registry Compatibility, Transactional Outbox & Dead-Letter Topologies

Self-explanatory list-and-bullet production mastery module covering Schema Registry Compatibility, Transactional Outbox & Dead-Letter Topologies in Kafka Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ Schema Registry Evolution Rules (Backward, Forward & Full Compatibility) — Purpose: Evolve event payloads across teams without breaking downstream Kafka consumers in production. • ✓ Non-Blocking Retry Topics, Dead-Letter Topics (DLT) & Idempotent Consumer Deduplication — Purpose: Isolate malformed or transiently failing records without stalling partition consumption. • ✓ Change Data Capture (CDC) with Debezium & Transactional Outbox Architecture — Purpose: Stream committed database changes into Kafka reliably with zero dual-write race conditions. • ✓ Build, verify, and document 'Avro/Protobuf Schema-Governed Order Pipeline with Retry Tiers & DLT Replay' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

15.1 Schema Registry Evolution Rules (Backward, Forward & Full Compatibility) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Schema Registry Evolution Rules (Backward, Forward & Full Compatibility) • Performance Optimization, Complexity & Latency Impact of Schema Registry Evolution Rules (Backward, Forward & Full.... • Live Debugging, Stack • Runnable Code Implementation (JAVA) • Hands • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Schema Registry Evolution Rules (Backward, Forward &... • Production Hardening & Security: mTLS/SASL ACLs, Schema Registry compatibility & poison-pill isolation

Purpose: To understand and apply Schema Registry Evolution Rules (Backward, Forward & Full Compatibility) in real Kafka systems — specifically: Evolve event payloads across teams...

What Topic Is Next → Next Topic 15.2: Non-Blocking Retry Topics, Dead-Letter Topics (DLT) & Idempotent Consumer Deduplication

15.2 Non-Blocking Retry Topics, Dead-Letter Topics (DLT) & Idempotent Consumer Deduplication [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Non • Performance Optimization, Complexity & Latency Impact of Non • Live Debugging, Stack • Runnable Code Implementation (JAVA) • Hands • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Non-Blocking Retry Topics • Production Hardening & Security: mTLS/SASL ACLs, Schema Registry compatibility & poison-pill isolation

Purpose: To understand and apply Non-Blocking Retry Topics, Dead-Letter Topics (DLT) & Idempotent Consumer Deduplication in real Kafka systems — specifically: Isolate malformed or...

What Topic Is Next → Next Topic 15.3: Change Data Capture (CDC) with Debezium & Transactional Outbox Architecture

15.3 Change Data Capture (CDC) with Debezium & Transactional Outbox Architecture [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Change Data Capture (CDC) with Debezium & Transactional Outbox Architectur... • Performance Optimization, Complexity & Latency Impact of Change Data Capture (CDC) with Debezium & Transactional Ou... • Live Debugging, Stack • Runnable Code Implementation (JAVA) • Hands • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Change Data Capture (CDC) with Debezium • Production Hardening & Security: mTLS/SASL ACLs, Schema Registry compatibility & poison-pill isolation

Purpose: To understand and apply Change Data Capture (CDC) with Debezium & Transactional Outbox Architecture in real Kafka systems — specifically: Stream committed database change...

What Topic Is Next → Module 15 Capstone (Avro/Protobuf Schema-Governed Order Pipeline with Retry Tiers & DLT Replay) → Next Topic 16.1: CooperativeStickyAssignor, Static Group Membership & Preventing Rebalance Storms

Module 15 Hands-On Capstone Project (100 Marks): Avro/Protobuf Schema-Governed Order Pipeline with Retry Tiers & DLT Replay — Architect, implement, test, and document 'Avro/Protobuf Schema-Governed Order Pipeline with Retry Tiers & DLT Replay' applying every bullet-wise engineering concept from Module 15 (Schema Re...

MODULE 16 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

KRaft Cluster Operations, Consumer Lag SRE & Event-Driven System Design

Self-explanatory list-and-bullet production mastery module covering KRaft Cluster Operations, Consumer Lag SRE & Event-Driven System Design in Kafka Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ CooperativeStickyAssignor, Static Group Membership & Preventing Rebalance Storms — Purpose: Keep high-throughput consumer groups stable during rolling pod deployments and GC pauses. • ✓ Monitoring Under-Replicated Partitions, ISR Shrinks, Disk Retention & Compaction — Purpose: Detect broker bottlenecks early and tune min.insync.replicas + acks=all for zero data loss. • ✓ Multi-Region MirrorMaker 2 & Event-Driven Architecture Interviews — Purpose: Design ordering guarantees, partition keys, and replayable event-sourced systems in interviews. • ✓ Build, verify, and document 'Consumer-Lag Autoscaling Monitor, Rebalance Tuning & Disaster-Recovery Mirroring' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

16.1 CooperativeStickyAssignor, Static Group Membership & Preventing Rebalance Storms [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for CooperativeStickyAssignor, Static Group Membership & Preventing Rebalance.... • Performance Optimization, Complexity & Latency Impact of CooperativeStickyAssignor, Static Group Membership & Preve... • Live Debugging, Stack • Runnable Code Implementation (JAVA) • Hands • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for CooperativeStickyAssignor • Production Hardening & Security: mTLS/SASL ACLs, Schema Registry compatibility & poison-pill isolation

Purpose: To understand and apply CooperativeStickyAssignor, Static Group Membership & Preventing Rebalance Storms in real Kafka systems — specifically: Keep high-throughput consum...

What Topic Is Next → Next Topic 16.2: Monitoring Under-Replicated Partitions, ISR Shrinks, Disk Retention & Compaction

16.2 Monitoring Under-Replicated Partitions, ISR Shrinks, Disk Retention & Compaction [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Monitoring Under • Performance Optimization, Complexity & Latency Impact of Monitoring Under • Live Debugging, Stack • Runnable Code Implementation (JAVA) • Hands • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Monitoring Under-Replicated Partitions • Production Hardening & Security: mTLS/SASL ACLs, Schema Registry compatibility & poison-pill isolation

Purpose: To understand and apply Monitoring Under-Replicated Partitions, ISR Shrinks, Disk Retention & Compaction in real Kafka systems — specifically: Detect broker bottlenecks e...

What Topic Is Next → Next Topic 16.3: Multi-Region MirrorMaker 2 & Event-Driven Architecture Interviews

16.3 Multi-Region MirrorMaker 2 & Event-Driven Architecture Interviews [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Multi • Performance Optimization, Complexity & Latency Impact of Multi • Live Debugging, Stack • Runnable Code Implementation (JAVA) • Hands • Idiomatic KAFK Implementation: partition key ordering, acks=all idempotent producers, Outbox & DLQ retries • Apache Kafka KRaft Quorum, Pagecache Sequential Log Segments & Consumer Groups Execution & Memory Lifecycle for Multi-Region MirrorMaker 2 • Production Hardening & Security: mTLS/SASL ACLs, Schema Registry compatibility & poison-pill isolation

Purpose: To understand and apply Multi-Region MirrorMaker 2 & Event-Driven Architecture Interviews in real Kafka systems — specifically: Design ordering guarantees, partition keys...

What Topic Is Next → Module 16 Capstone → End-of-Course Live Client Project: StreamPulse Event-Driven Order Pipeline, Dead-Letter Recovery & Outbox Relay

Module 16 Hands-On Capstone Project (100 Marks): Consumer-Lag Autoscaling Monitor, Rebalance Tuning & Disaster-Recovery Mirroring — Architect, implement, test, and document 'Consumer-Lag Autoscaling Monitor, Rebalance Tuning & Disaster-Recovery Mirroring' applying every bullet-wise engineering concept from Module 16 (KRa...

3. LIVE CLIENT PRODUCTION PROJECT & CAREER SCOPE

End-of-Course Live Client Production Project + Official Internship Confirmation Letter

Project: StreamPulse Event-Driven Order Pipeline, Dead-Letter Recovery & Outbox Relay • **Confirmed Role:** Apache Kafka & Event-Driven Systems Intern

Domain & Production Infra: High-Volume Retail & FinTech Event Streaming Client (Apache Kafka KRaft Cluster + Schema Registry (Avro/JSON) + Kafka Connect + Consumer Groups)

- Designed partitioned Kafka topics, replication factors, and key-based ordering strategies for high-throughput events.
- Built idempotent producers, transactional outbox relays, and manual-commit consumer groups with zero message loss.
- Configured Schema Registry compatibility rules, exponential retry topics, and Dead-Letter Queue (DLQ) replay tooling.
- Monitored consumer lag, broker throughput, and partition rebalancing on a live containerized Kafka cluster.

Target Job Roles: Kafka & Event-Driven Backend Engineer, Real-Time Data Streaming Engineer, Distributed Systems Software Engineer, Platform / Messaging Infrastructure Engineer

Realistic Annual CTC Scope: **Associate Streaming / Backend Engineer (0–1 Yr):** ₹6.5 LPA – ₹12.0 LPA **Distributed Microservices / Data Engineer (1–3 Yrs):** ₹13.0 LPA – ₹25.0 LPA **Principal Event-Driven Systems Architect (3–6+ Yrs):** ₹26.0 LPA – ₹48.0+ LPA