Apache Kafka Fundamentals

The Complete Guide to Distributed Event Streaming

Agenda

1. The Birth of Kafka
2. What is Apache Kafka?
3. Core Kafka Architecture
4. Topics & Partitions
5. Replication & Fault Tolerance
6. ZooKeeper vs KRaft
7. How Kafka Stores Data
8. Producers
9. Consumers & Consumer Groups
10. Kafka Connect
11. Kafka Streams
12. Kafka Use Cases
13. Monitoring & Metrics
14. Security in Kafka
15. Hands-on Demo
16. Wrap-Up

The Birth of Kafka

2010–2011 at LinkedIn

The Problem

- Billions of events/day
- Real-time analytics needed
- Traditional brokers couldn’t scale

The Solution Team

- Jay Kreps
- Neha Narkhede
- Jun Rao

Kafka Timeline

Timeline

Year	Milestone
2011	Open-sourced Kafka
2013	Apache Top-Level Project
2014	Confluent founded
2016	Kafka Streams introduced
2020	KRaft mode (no ZooKeeper)

What is Apache Kafka?

Definition

Apache Kafka is a *distributed, fault-tolerant event streaming platform* that stores, processes, and transports real-time data at scale.

Core Roles

Publish/subscribe • Durable storage • Stream processing

Analogy

Like a high-speed, persistent *data conveyor belt* where many apps can place and pick up packages.

Core Kafka Architecture

```
[Producer Apps]  --->  [Kafka Brokers Cluster]  --->  [Consumer Apps]
                          (Topics & Partitions)
```

Key Components

- **Producers** send messages to topics
- **Brokers** store and serve messages
- **Topics** are named categories of messages
- **Partitions** are parallel units of a topic
- **Consumers** read messages from topics
- **Cluster Coordination** via ZooKeeper or KRaft

Topics & Partitions

```
Topic: user-events
 ├─ Partition 0: Msg1 → Msg2 → Msg3
 ├─ Partition 1: Msg4 → Msg5 → Msg6
 └─ Partition 2: Msg7 → Msg8 → Msg9
```

- **Topics**: Logical streams of data
- **Partitions**: Scale & parallelize consumption
- **Keys**: Route to partitions; ordering within a partition

Replication & Fault Tolerance

```
Partition 0 (Replication Factor = 3)
  ┌──────────┐ Leader   (Broker 1)
  ├──────────┤ Follower (Broker 2)
  └──────────┘ Follower (Broker 3)
```

- **Leader** handles reads/writes; **followers** replicate
- **ISR**: In-Sync Replicas
- Prevent data loss: `unclean.leader.election.enable=false`
- Best practice: `min.insync.replicas=2` with RF=3

ZooKeeper vs KRaft

ZooKeeper
External service for metadata & leader election.
More components to run and manage.

KRaft (Kafka Raft)
Built-in consensus; simpler ops, faster startup, better scalability.
Default in Kafka 3.5+.

How Kafka Stores Data

Commit Log
Append-only files on disk. Retain by *time* or *size*; optional *log compaction* keeps latest record per key.

Segments
Partitions split into segments. Old segments are deleted or compacted based on policy.

Producers

What is a Producer?

An application that publishes messages to Kafka topics.

Partitioning

- **Key-based** → same key → same partition
- **No key** → round-robin distribution

Key Settings

```
acks=all
enable.idempotence=true
compression.type=snappy
```

Consumers & Consumer Groups

```
Topic: orders (6 partitions)
Group: analytics
  ┌──────────┬──────────┬──────────┐
  │Consumer1 │Consumer2 │Consumer3 │
  │ [P0, P1] │ [P2, P3] │ [P4, P5] │
  └──────────┴──────────┴──────────┘
```

- Each **group** sees all messages
- Within a group, a **partition** is consumed by one member
- **Offsets** track read position

Kafka Connect

Purpose
Integrate Kafka with external systems *without custom code*.

Types
Source connectors (import) • Sink connectors (export)
Runs in standalone or distributed mode.

Kafka Streams

Purpose
Build real-time applications directly on Kafka.

Highlights
Java/Scala library • Stateful ops (aggregations, joins, windows) • Exactly-once semantics • Scales with partitions.

Kafka Use Cases

Great Fit

Event streaming (user activity, IoT)

Real-time analytics (fraud, dashboards)

Microservice communication (EDA)

Not Ideal

Ultra low latency trading

Small simple queues

OLTP DB replacement

Monitoring & Metrics

- Kafka CLI tools
- Prometheus + Grafana
- Confluent Control Center

Watch: Broker health • Topic/partition status • Consumer lag

Security in Kafka

AuthN
SASL/PLAIN • SASL/SCRAM • Kerberos • OAuth

Encryption
TLS in transit

AuthZ
ACLs per topic, group, and cluster resources

Hands-On Demo

We'll cover:

- Creating topics
- Producing and consuming messages
- Observing replication
- Checking consumer lag

Reference Handouts:

- `01-basic-commands.md`
- `02-kcat-examples.md`
- `03-python-examples.md`
- `04-load-test-commands.md`

We'll cover:

- Creating topics
- Producing and consuming messages
- Observing replication
- Checking consumer lag

Wrap-Up

We covered:

- Kafka basics & architecture
- Topics, partitions, replication
- Producers & consumers
- Storage model, Connect, Streams
- Use cases, monitoring, security

Next: Hands-on practice & advanced configs

Appendix: Config Highlights

Producer

```
acks=all
enable.idempotence=true
compression.type=snappy
```

Consumer

```
group.id=my-group
auto.offset.reset=earliest
```

Broker

```
default.replication.factor=3
min.insync.replicas=2
```