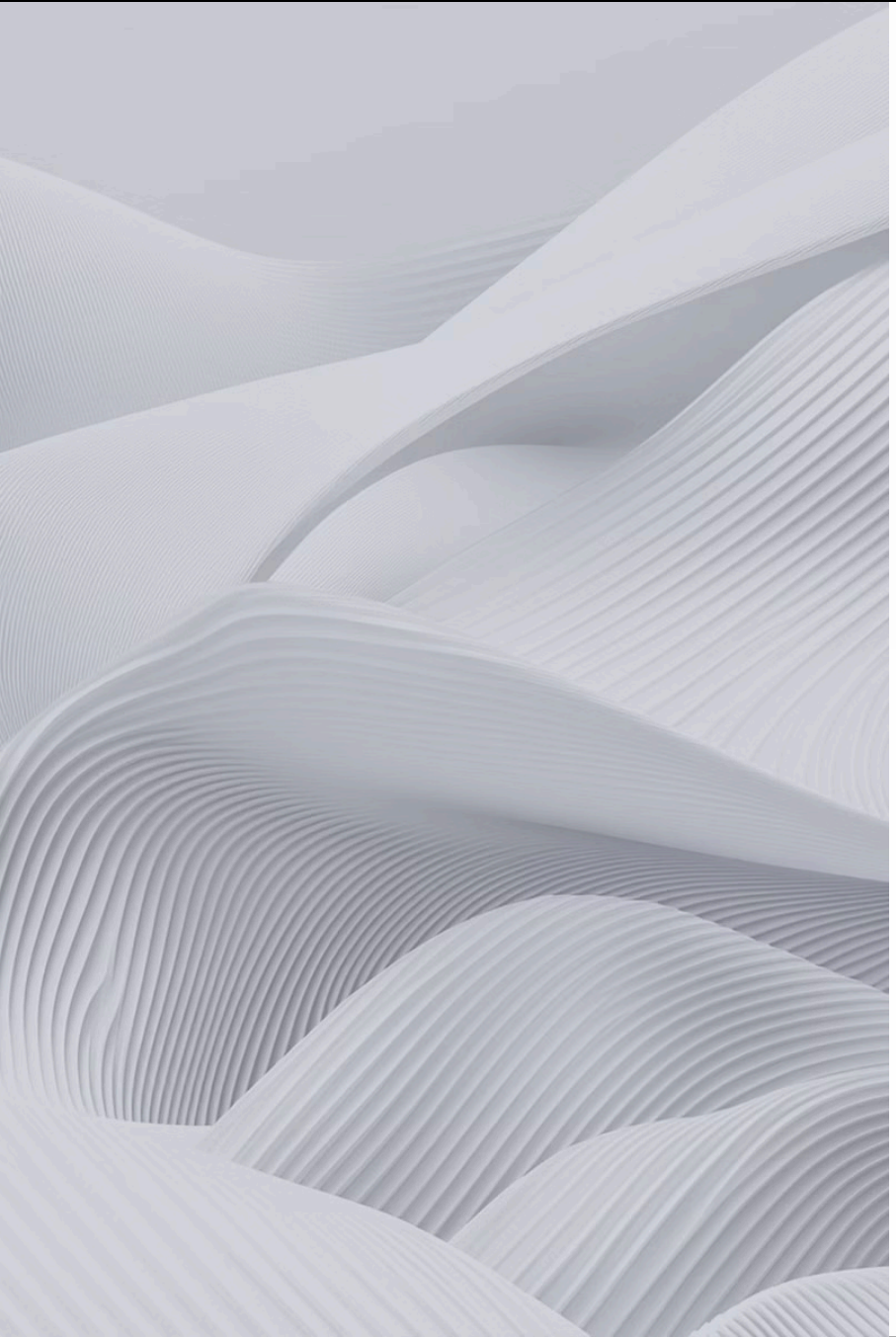
Open-source distributed event streaming platform

Ngoc My Nguyen

June 2025



Agenda

1. Distributed Message Brokers

A distributed message broker

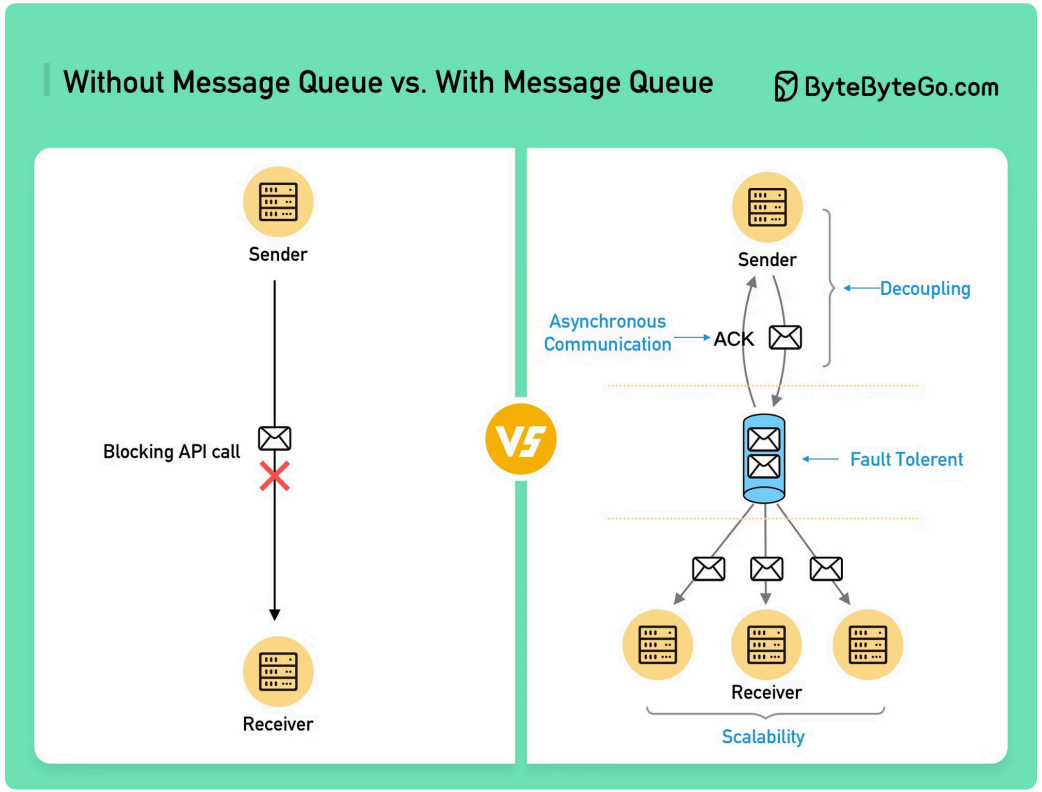
Serves as the communication backbone between different software systems, enabling asynchronous exchange of information without direct connections.

Decoupling producers and consumers

Create flexible, scalable architectures where components can be added, removed, or modified with minimal impact on the overall system.

Distributed nature advantages

Provides horizontal scalability, fault tolerance, and geographic distribution for optimized performance.



2. Log Systems

Append-only ledger

Records every event or message in a durable, ordered manner.

Replayability and fault recovery

Foundation for replayability and fault recovery.

Auditing and data flow tracking

Maintains a persistent record of all data flows.

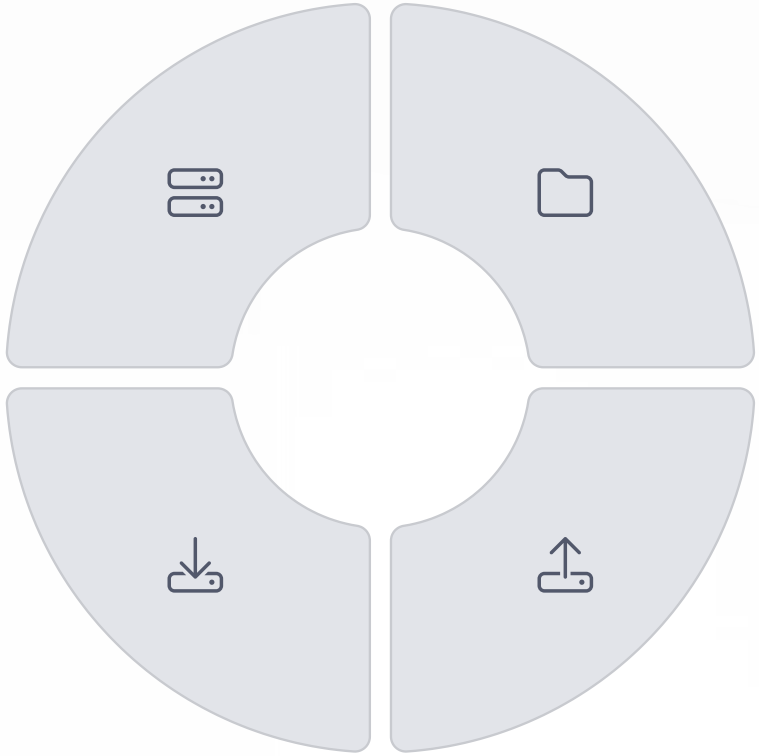
3. Kafka Architecture Overview

Brokers & Clusters

Server instances that store and manage published data. Multiple brokers form a cluster.

Topics & Partitions

Logical channels for message categories. Each topic can be split into multiple partitions.



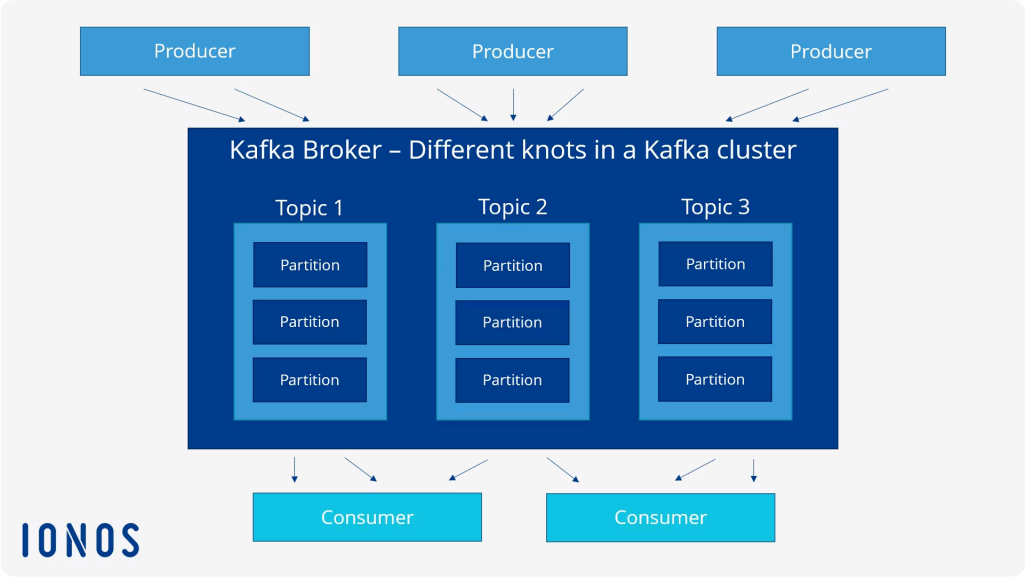
Consumers

Applications that subscribe to topics and process published messages. Often organized into groups.

Producers

Applications that publish messages to Kafka topics. They can choose partition assignment strategies.

3.1. Brokers and Clusters



Broker Responsibilities

- Handle read and write requests from clients
- Manage partition replicas for fault tolerance
- Store messages on disk with configurable retention

Cluster Controller

- One broker serves as the controller
- Manages administrative operations
- Handles broker failures and partition reassignment

3.2. Topics and Partitions

Topic Creation

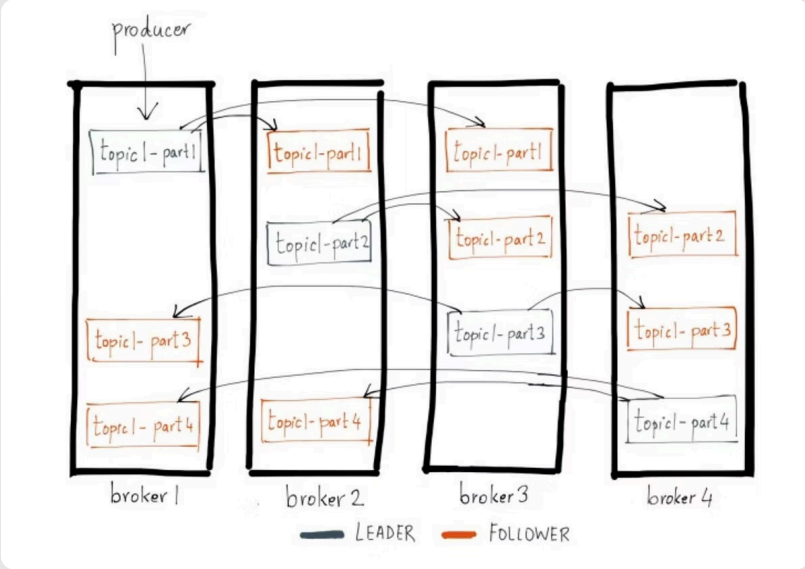
Topics are created with a name and partition count. They act as feed names to categorize messages.

Partition Distribution

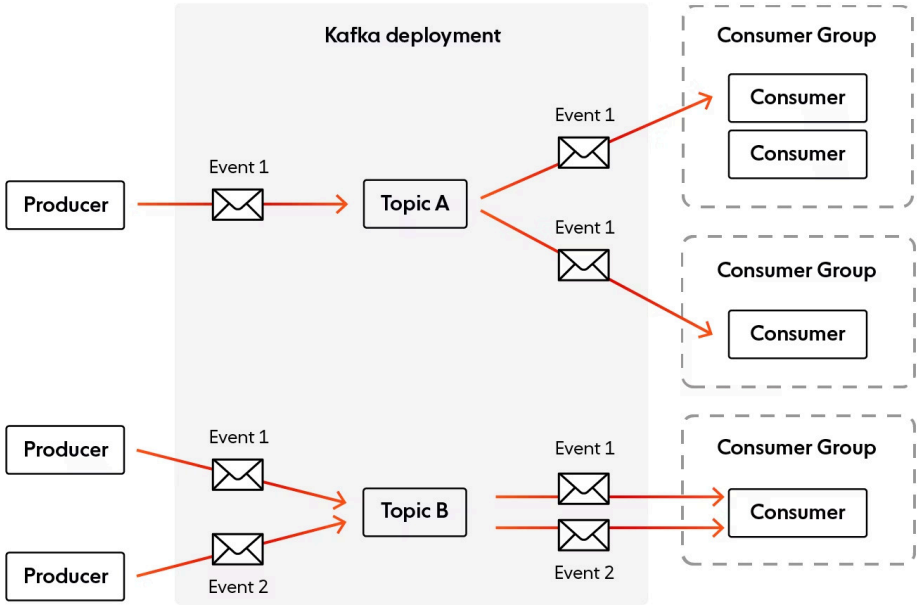
Each partition is distributed across brokers. This allows horizontal scaling and parallel processing.

Replication

Partitions are replicated to multiple brokers. One replica serves as the leader, handling all reads and writes.



3.3. Producers and Consumers



Producers

Send messages to topics with configurable delivery guarantees. Can specify keys to control partition routing.

Consumers

Read messages from topics, tracking their position via offsets. Can read from multiple partitions simultaneously.

Consumer Groups

Multiple consumers that work together. Each partition is read by exactly one consumer in the group.

4. Client Libraries

Kafka supports multiple programming languages through official client libraries

1 Java Client

Docs | Tutorial | API | GitHub

2 C/C++ Client

Docs | Tutorial | API | GitHub

3 Python Client

Docs | Tutorial | API | GitHub

4 Go Client

Docs | Tutorial | API | GitHub

5 .NET Client

Docs | Tutorial | API | GitHub

6 JavaScript Client

Docs | Tutorial | API | GitHub

5. Programming with Kafka

1 Producer API

Send messages to topics with configurable reliability guarantees

- Synchronous/asynchronous delivery
- Custom partitioning strategies
- Automatic batching and compression

2 Consumer API

Subscribe to topics and process messages

- Manual/automatic offset management
- Consumer groups for scalability
- Fault-tolerant rebalancing

3 Streams

Build real-time processing applications with Streams DSL

- Stateful/stateless transformations
- Time-based windowing operations
- Exactly-once processing

4 Monitoring

Track system health and performance

- JMX metrics for brokers/clients
- Monitoring tool integration
- Consumer group lag tracking

All APIs include robust error handling and secure authentication.

6. Typical Kafka Use Cases

Real-Time Analytics

Process streaming data for dashboards, monitoring systems, and business intelligence. Enables instant insights from user behavior.

Event-Driven Architecture

Facilitate communication between microservices. Enables loose coupling and asynchronous processing between components.

Data Pipeline Backbone

Connect diverse data sources and destinations. Powers ETL processes, data lakes, and real-time data integration.

Log Aggregation

Collect logs from distributed systems and microservices. Creates a centralized stream for analysis, monitoring, and alerting.

7. Installation Options

Binary tarballs from Apache Kafka website

Option for manual installation and configuration.

Docker containers for development and testing

Easy setup and portability across environments.

Kubernetes operators for cloud-native deployments

Scalable and efficient deployment on Kubernetes clusters.

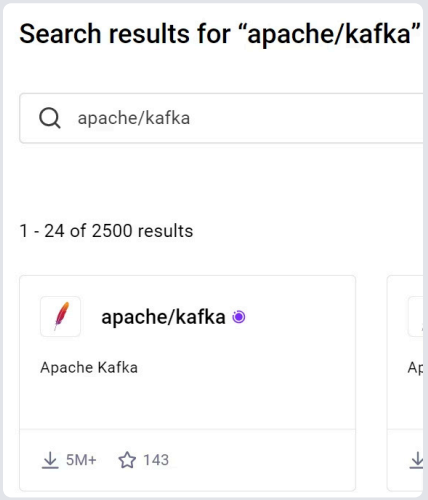
Managed services: AWS MSK, Confluent Cloud, Aiven

Outsource management and maintenance to cloud providers.

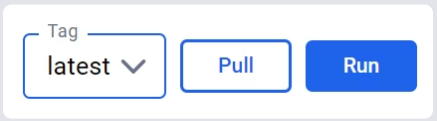
Instructions can be found **here**.

7.1. Example: Easy installation and configuration with Docker Desktop (GUI)

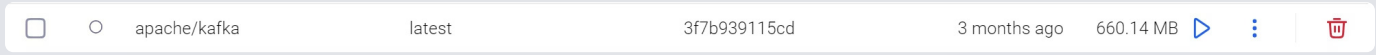
1. Search for "apache/kafka" on Docker Hub and click on the result.



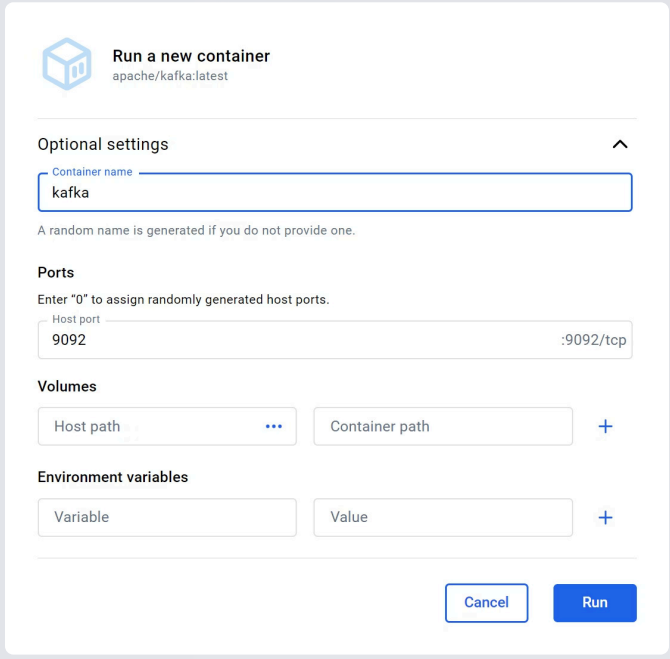
2. On the upper right side, select and pull the desired version.



3. Locate the downloaded image and run it.



4. Configure the container by editing the fields listed below.



7.2. Create a topic via the Kafka Bash

Navigate to the created container and type in this command to create a topic

docker exec -it <your-contain-name> /opt/kafka/bin/kafka-topics.sh --create --topic <your-topic-name> --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1