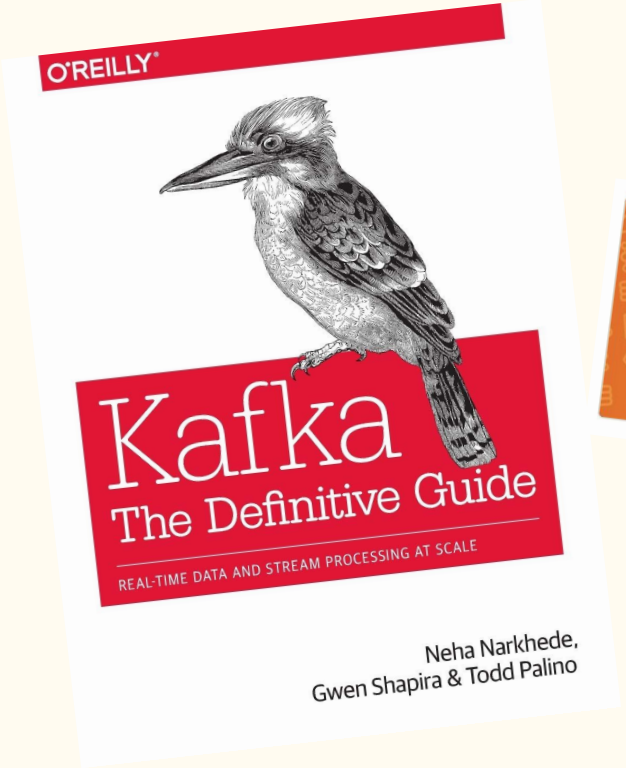
Kafka

Part 1

Resources


O'REILLY
Kafka
The Definitive Guide
REAL-TIME DATA AND STREAM PROCESSING AT SCALE
Neha Narkhede,
Gwen Shapira & Todd Palino


Apache Kafka Series - Learn Apache Kafka for Beginners v2
START HERE: Learn Apache Kafka 2.0 Ecosystem, Core Concepts, Real World Java Producers/Consumers & Big Data Architecture
Stephane Maarek | AWS Certified Solutions Architect & Developer Associate
4.7 (18,290)
7.5 total hours • 123 lectures • All Levels
Bestseller

$$7\ 000\ 000\ 000\ 000\ 000\ 000 \Leftrightarrow 7 * 10^{18}$$

Kafka at LinkedIn - 2019

7 000 000 000 000 000 000 ⇔ 7 * 10^18

7 trillions messages handled / 24h

100 Kafka clusters

4 000 brokers

100 000 topics

7 000 000 partitions

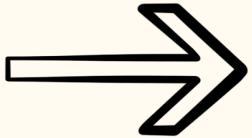
Why Kafka - LinkedIn original use case

User activity tracking

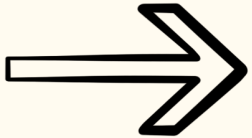
Page views

Click tracking

Profile updates



Topics

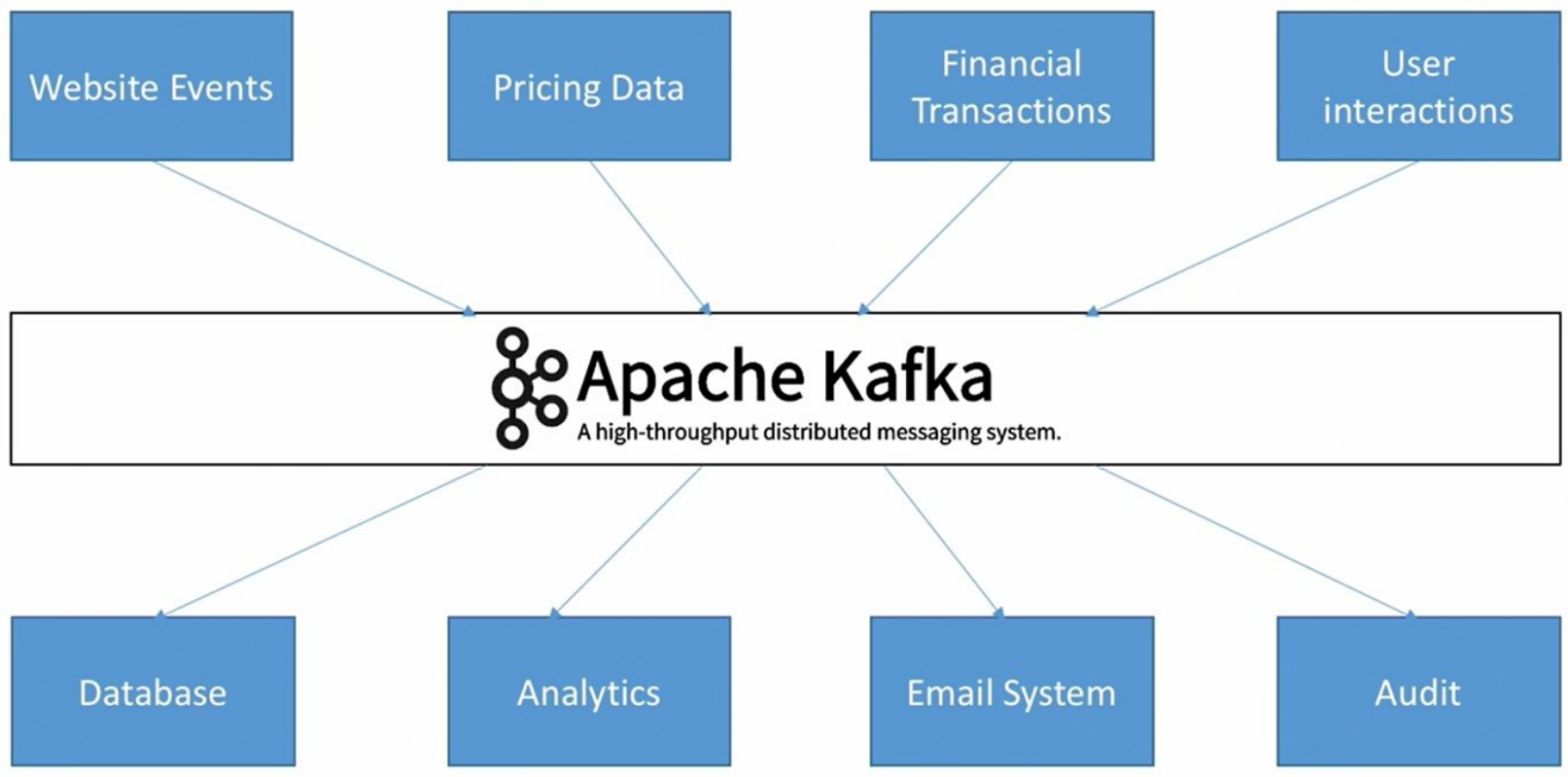


Report generators

Machine learning systems

Search results updater

Why Kafka


Website Events
Pricing Data
Financial Transactions
User interactions
Apache Kafka
A high-throughput distributed messaging system.
Database
Analytics
Email System
Audit

Why Kafka - other use cases

Activity tracking

Messaging

Metrics and logging

Stream processing

Kafka and its benefits

Created at LinkedIn, now open source

Decouple data streams from services

Horizontal scalability - can scale to millions of messages per second

High performance - less than 10ms latency

Durable message retention

Vocabulary/concepts

Message

Topic

Partition

Broker

Replication factor

Producer

Consumer

Message

Unit of data - similar to a row or a record in a DB

An array of bytes - it has no meaning/format to Kafka

Can have an optional key - impact the writing strategy

Topic

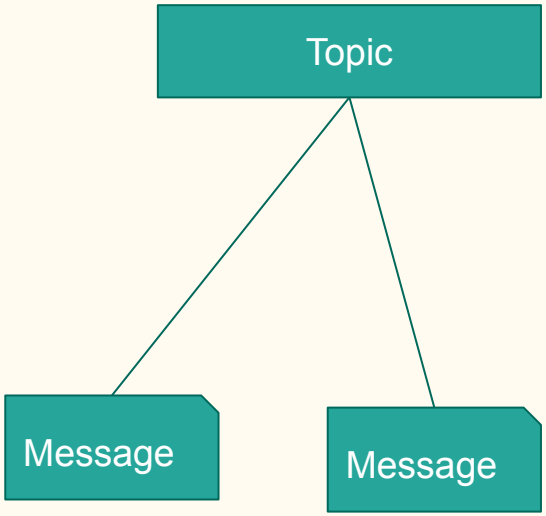
A stream of data

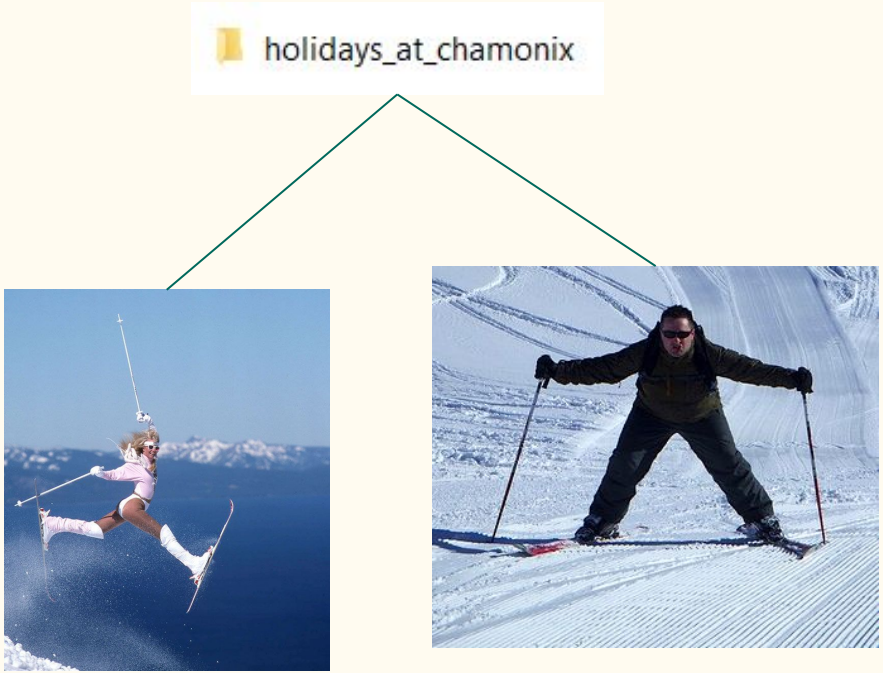
Similar to a folder in a file system or a table in a DB

Identified by its name

Splits in partitions

Topic example





Partition

Subset of data for a topic

Each message within a partition gets an incremental id called offset

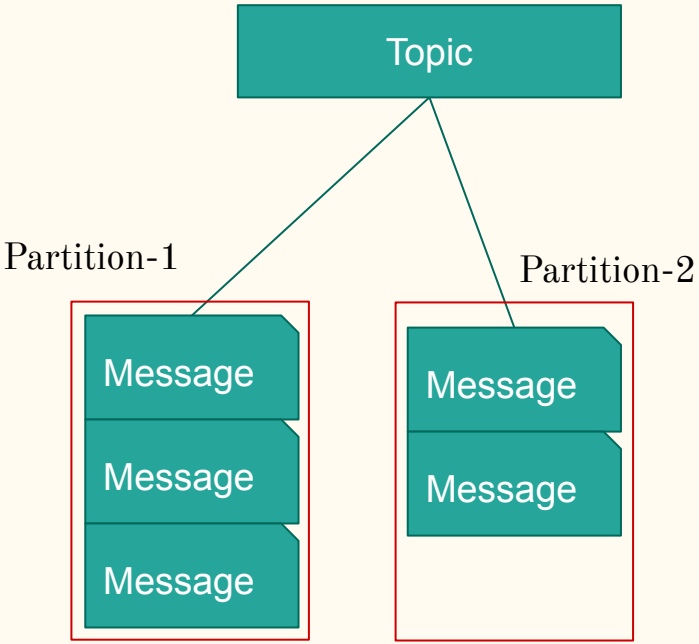
Order is guaranteed only within a particular partition

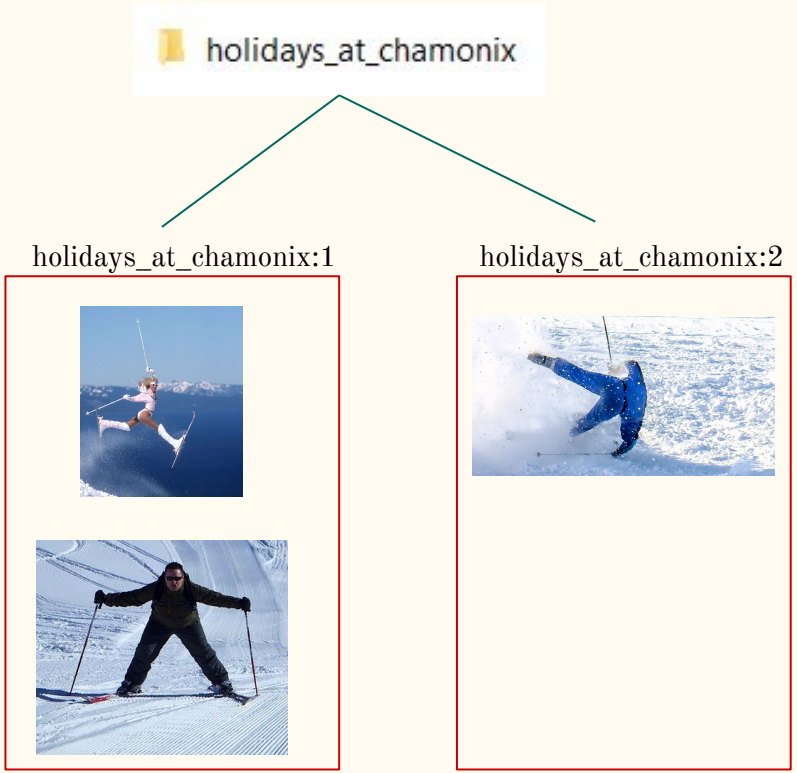
Data written to a partition is immutable, can’t be changed

Data is assigned randomly to a partition unless a key is provided

Partitions are evenly distributed across brokers in a Kafka cluster

Partition example - subdfolders





Broker

A Kafka server

Identified by an Integer

A Kafka cluster is composed of multiple brokers

Once connected to a broker, you are connected to the entire cluster

What happens to the associated partition(s) if a broker goes down?

Broker example

Personal laptop

holidays_at_chamonix

holidays_at_chamonix:1

holidays_at_chamonix:2

Work laptop

holidays_at_chamonix

holidays_at_chamonix:3

Replication factor

Number of copies of partitions for a topic

Duplicate data to avoid losing it if a broker goes down

A replication factor of 2 means that we would have 2 copies of data in different brokers

Replication factor example

Personal laptop

holidays_at_chamonix

holidays_at_chamonix:1

holidays_at_chamonix:2 copy

Work laptop

holidays_at_chamonix

holidays_at_chamonix:2

holidays_at_chamonix:1 copy

Producer

Write data to topic

A key can be sent along with the message

If the key is sent, all message with the same key will be written to the same partition - Allowing to maintain message ordering

If the key is not sent then data is written in partitions in a round robin manner

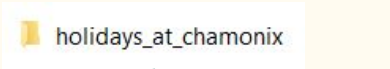
Producer example

Sends to topic

with key=”manol”

with key=”vlad”

with key=”manol”



holidays_at_chamonix:1

holidays_at_chamonix:2

Consumer

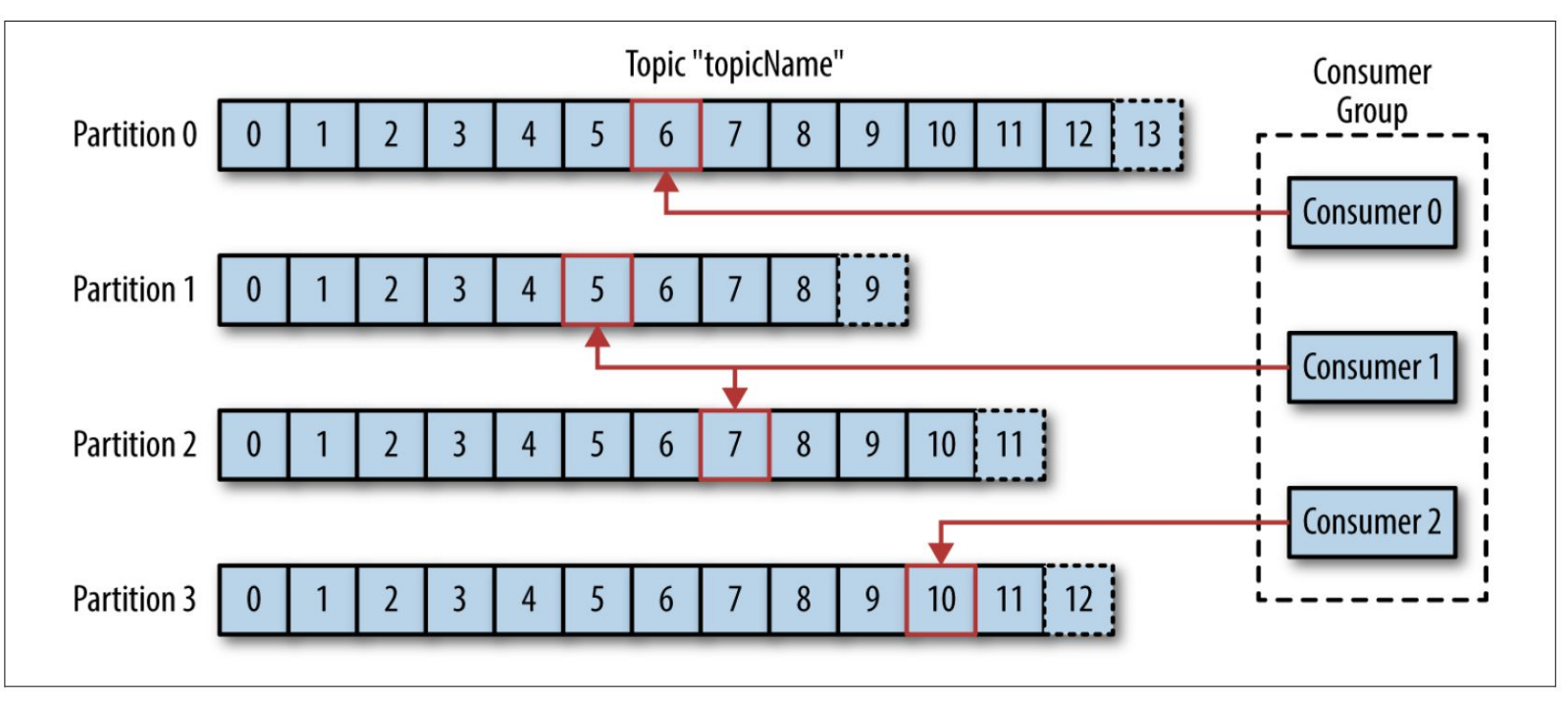
Consume data from topic

Data is read in order within each partitions

Consumers are part of a consumer group

Only 1 consumer within a consumer group can read data from a partition - If we have 3 consumers and 2 partitions then 1 consumer will be inactive

Consumer example



Demo