CLOUDERA
Educational Services

Apache Kafka Basics

CLOUDERA

Apache Kafka Basics

Chapter 1

Course Chapters

- **Apache Kafka Basics**

Message Processing with Apache Kafka

After completing this chapter, you will be able to

- **Explain what Apache Kafka is and what advantages it offers**
- **Describe the high-level architecture of Kafka**
- **Create topics, publish messages, and read messages from the command line**

Chapter Topics

Apache Kafka Basics

- **Apache Kafka Overview**
- Messages and Topics
- Producers and Consumers
- Kafka in Context
- Command Line Tools
- Essential Points
- Hands-On Exercise: Using the Apache Kafka Command-Line Tools

What Is Apache Kafka?

- **Apache Kafka is a fast, scalable, distributed publish-subscribe messaging system**
  - Widely used for data ingest
  - Offers scalability, performance, reliability, and flexibility
- **Originally created at LinkedIn, now an open source Apache project**
  - Donated to the Apache Software Foundation in 2011
  - Graduated from the Apache Incubator in 2012

Characteristics of Kafka

- **Scalable**
  - Kafka is a distributed system that supports multiple nodes
- **Fault-tolerant**
  - Data is persisted to disk and can be replicated throughout the cluster
- **High throughput**
  - Each broker can process hundreds of thousands of messages per second*
- **Low latency**
  - Data is delivered in a fraction of a second
- **Flexible**
  - Decouples the production of data from its consumption

*Using modest hardware, with messages of a typical size

Kafka Use Cases

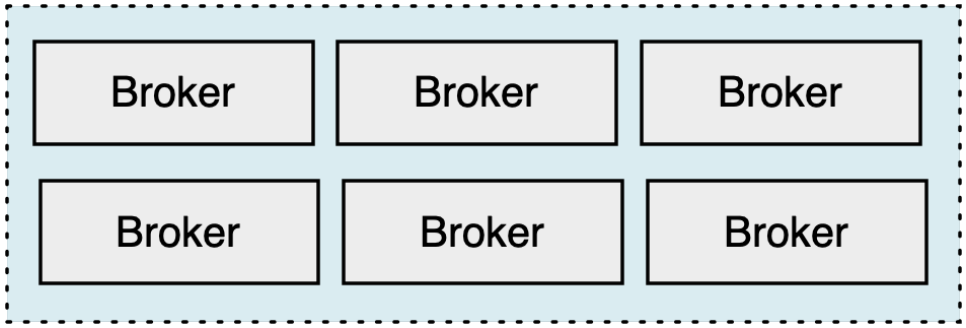
- **Kafka is used for a variety of use cases, such as**
  - Log aggregation
  - Messaging
  - Web site activity tracking
  - Stream processing
  - Event sourcing

Key Terminology

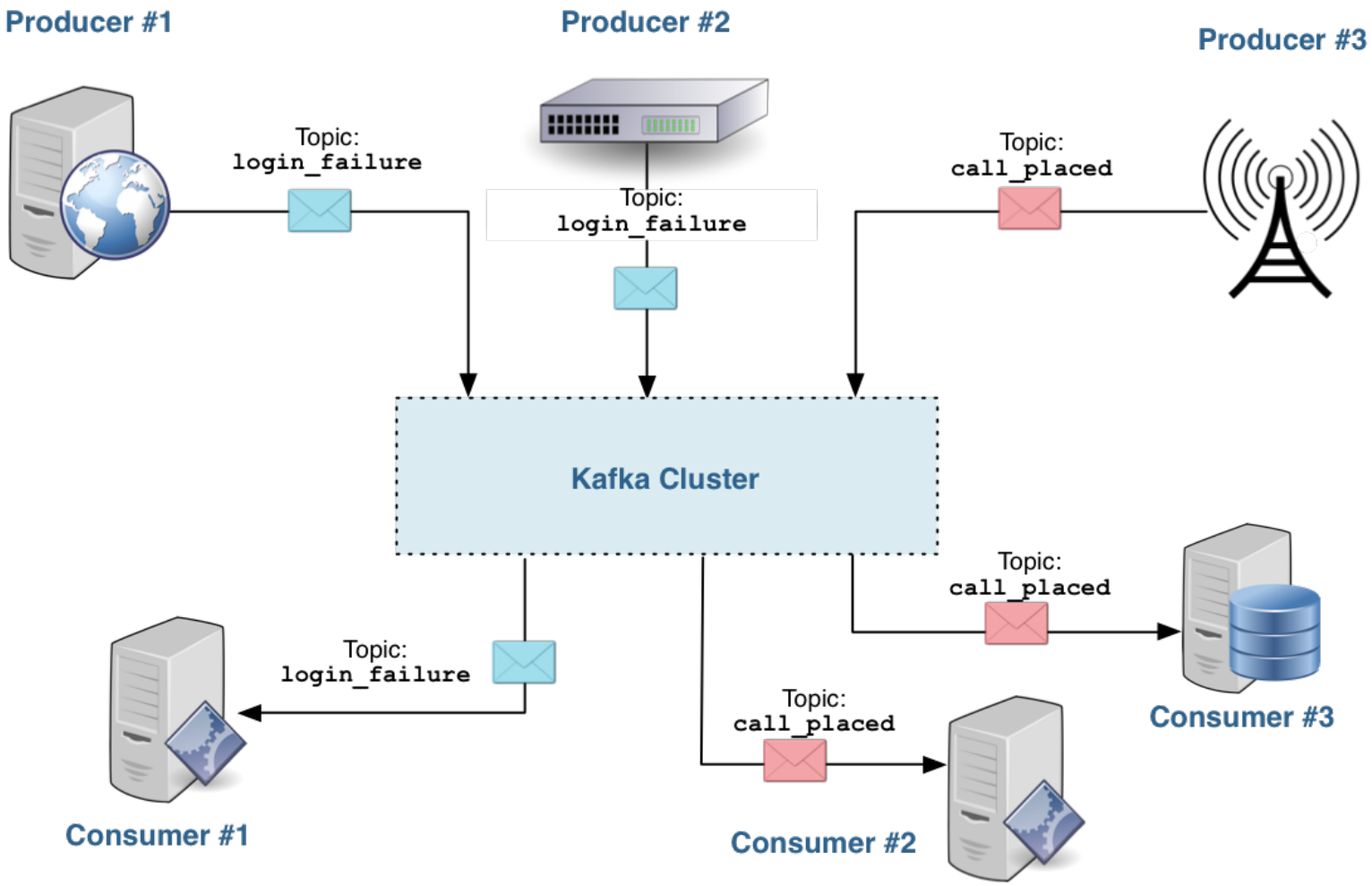
- **Message**
  - A single data record passed by Kafka
- **Topic**
  - A named log or feed of messages within Kafka
- **Producer**
  - A program that writes messages to Kafka
- **Consumer**
  - A program that reads messages from Kafka

Kafka Clusters

- **Clusters are composed of interconnected nodes running Kafka software**
  - A *broker* is the Kafka service that listens for client connections



Example: High-Level Architecture



Chapter Topics

Apache Kafka Basics

- Apache Kafka Overview
- **Messages and Topics**
- Producers and Consumers
- Kafka in Context
- Command Line Tools
- Essential Points
- Hands-On Exercise: Using the Apache Kafka Command-Line Tools

Messages

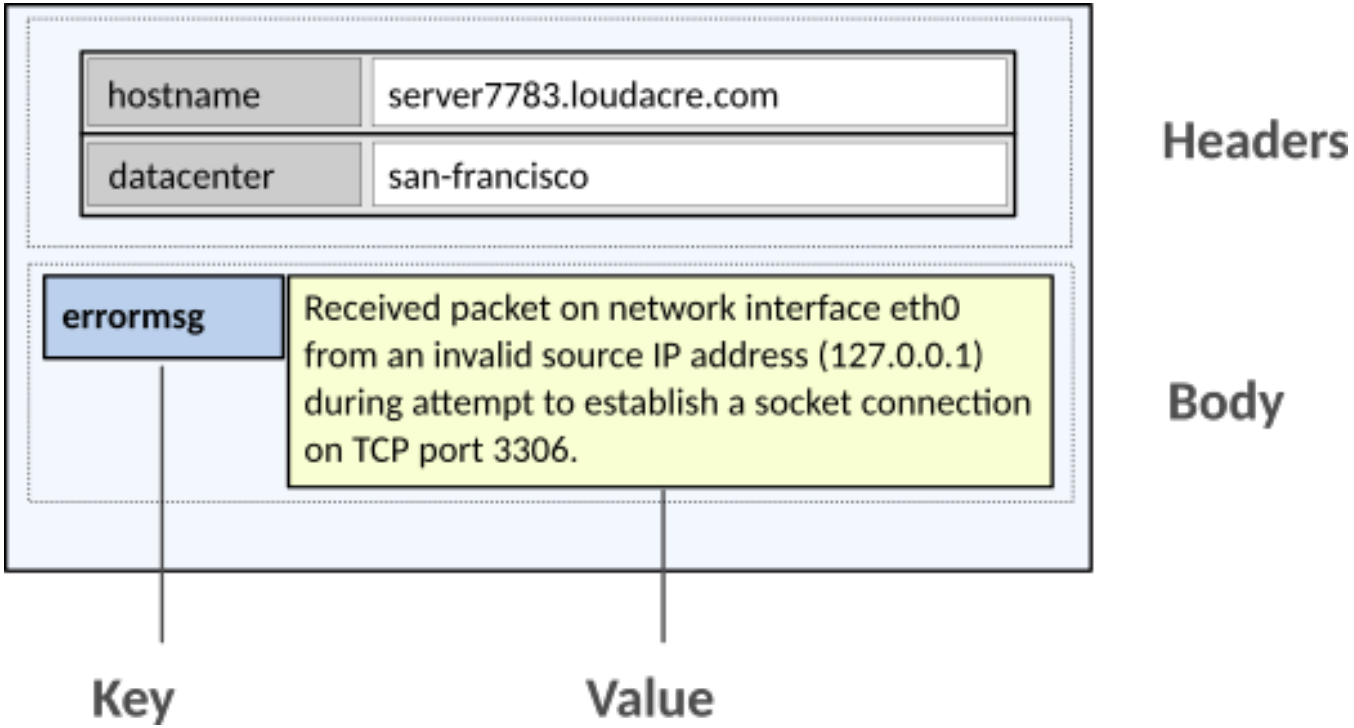
- **Kafka messages (also known as records) carry application data**
- **Messages are variable-size byte arrays**
  - Represent arbitrary user-defined content
  - Uses any format your application requires
  - Common formats include free-form text, JSON, and Avro
- **There is no explicit limit on message size**
  - Optimal performance at a few KB per message
  - Practical limit of 1MB per message

Messages Retention

- **Kafka retains all messages for a defined time period and/or total size**
  - Administrators can specify retention on global or per-topic basis
  - Kafka will retain messages regardless of whether they were read
  - Kafka discards messages automatically after the retention period or total size is exceeded (whichever limit is reached first)
  - Default retention is one week
  - Retention can reasonably be one year or longer

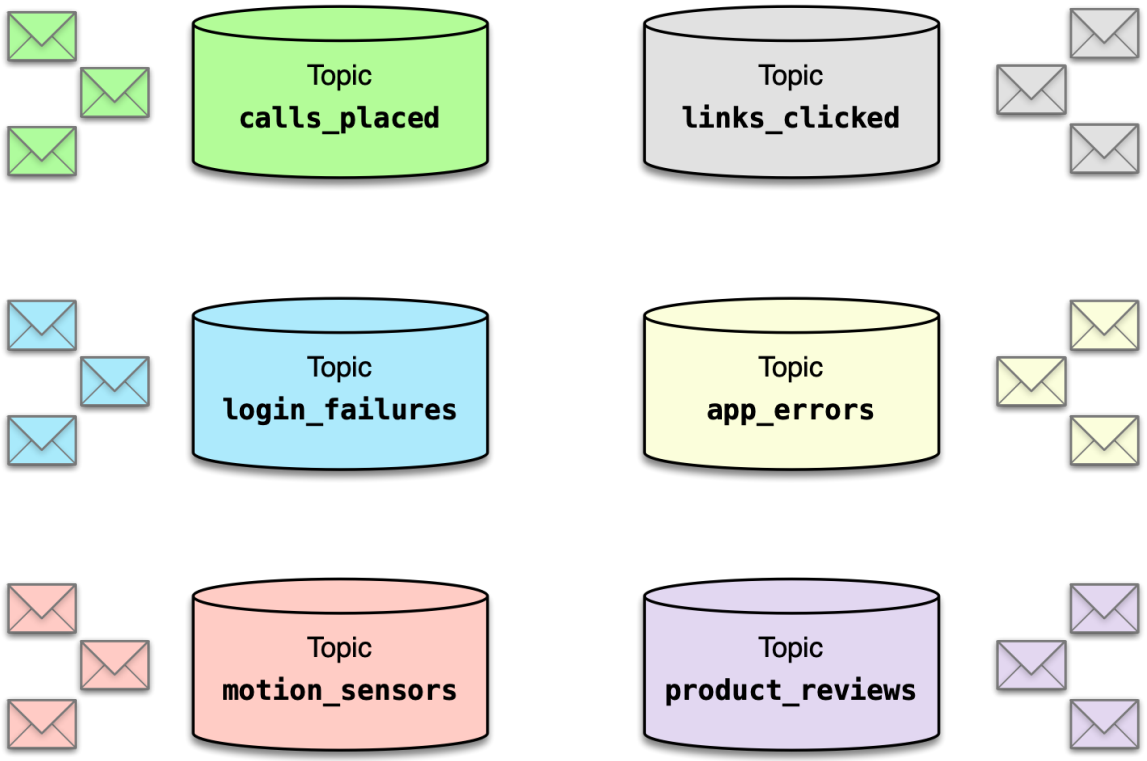
Anatomy of a Kafka Record

- **Headers are an optional set of name-value pairs**
  - Typically used for metadata and message processing
  - Name is of type `String`, value is a `byte[]`
- **The payload is sent in the record's body**
  - Consists of a key (often empty) and value
  - Both can be of any type, but each is ultimately converted to/from `byte[]`



Topics (1)

- **Messages are classified by categories, known as *topics***
  - Kafka clients specify the topic when sending and receiving messages
  - Kafka brokers use topics to organize and store data

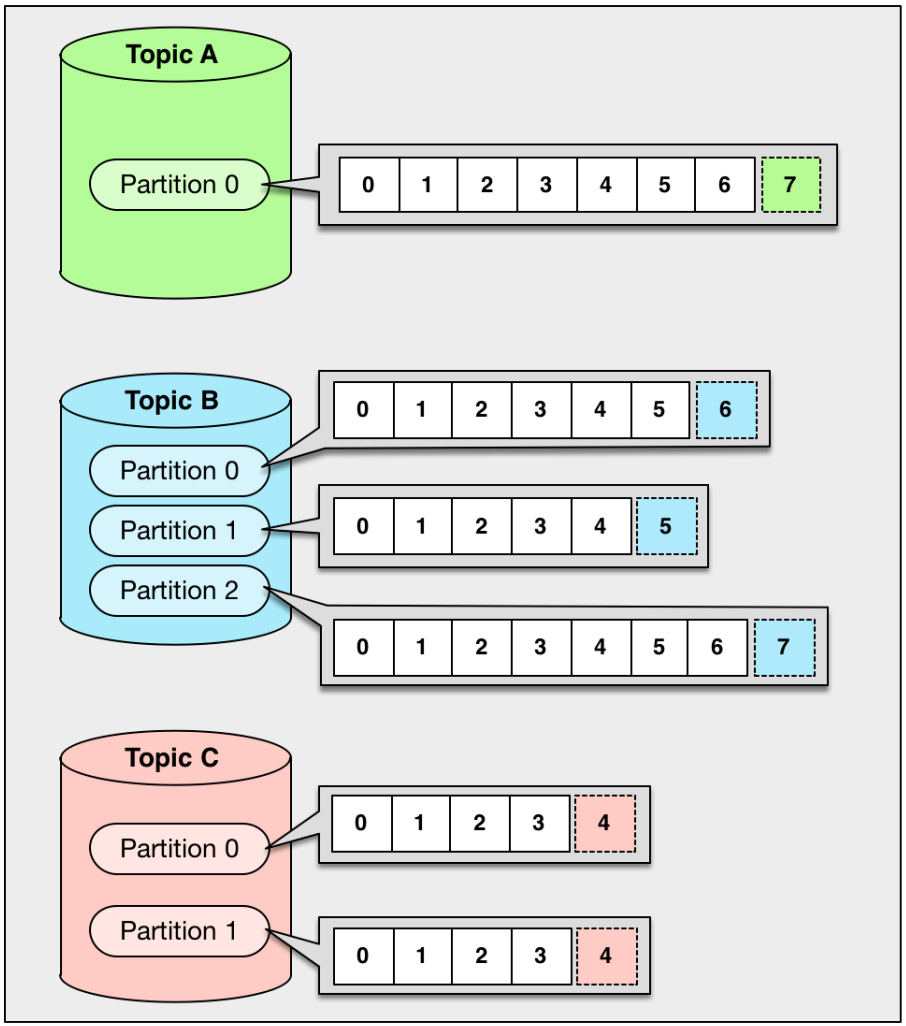


Topics (2)

- **There is no explicit limit on the number of topics**
  - However, Kafka works better with a few large topics than many small ones
- **A topic can be created explicitly or simply by publishing to the topic**
  - This behavior is configurable
  - Cloudera recommends that administrators disable auto-creation of topics to avoid accidental creation of large numbers of topics

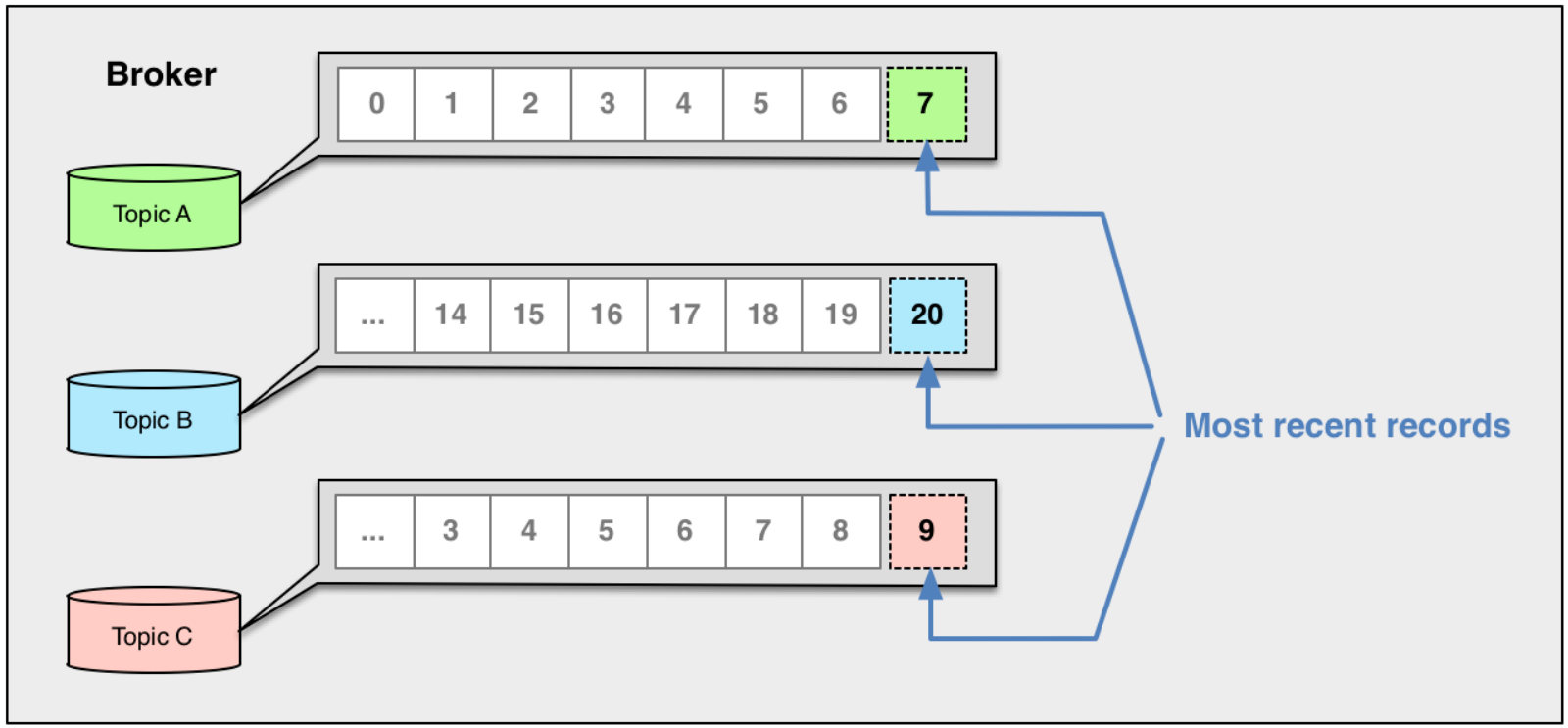
Partitioning

- **For better scalability, each topic can be divided into multiple *partitions***



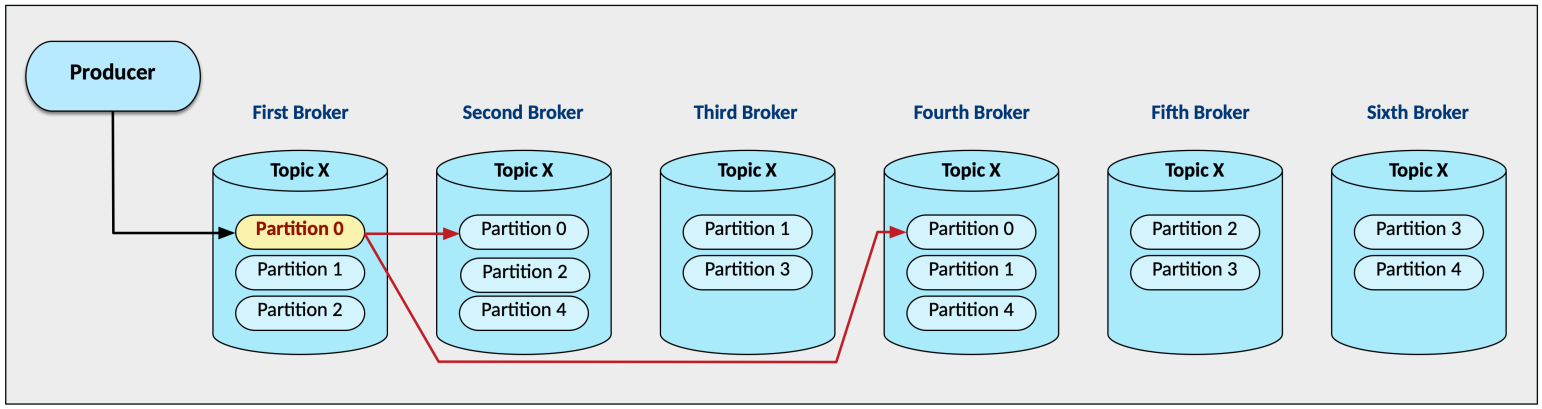
Message Storage and Retention

- **Messages are immutable and stored in the order sent**
  - May be deleted after the specified retention period has elapsed
  - Retention period is configurable on a per-topic basis



Replication

- ***Replication* distributes copies of a partition's data across multiple brokers**
  - A topic's replication factor is set at creation
  - Provides fault tolerance
- **Example: a six-node cluster with five partitions and three replicas**
  - Partition 0 of Topic X is replicated on the first, second, and fourth brokers
  - The copy on the first broker is the *leader*.



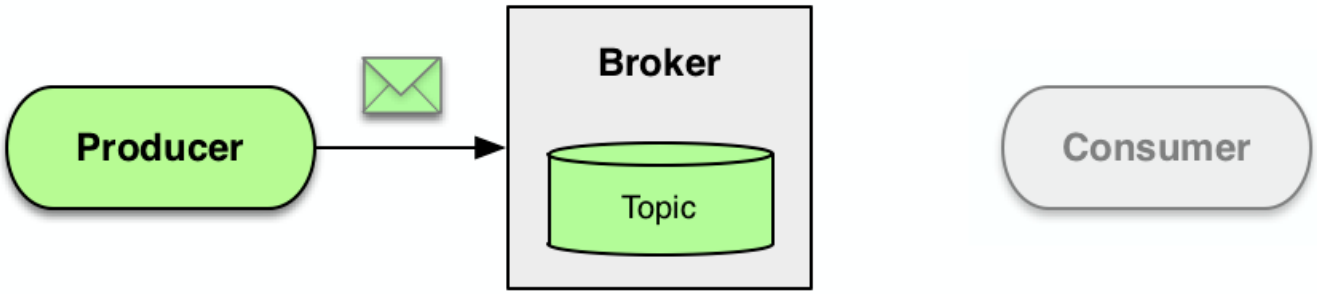
Chapter Topics

Apache Kafka Basics

- Apache Kafka Overview
- Messages and Topics
- **Producers and Consumers**
- Kafka in Context
- Command Line Tools
- Essential Points
- Hands-On Exercise: Using the Apache Kafka Command-Line Tools

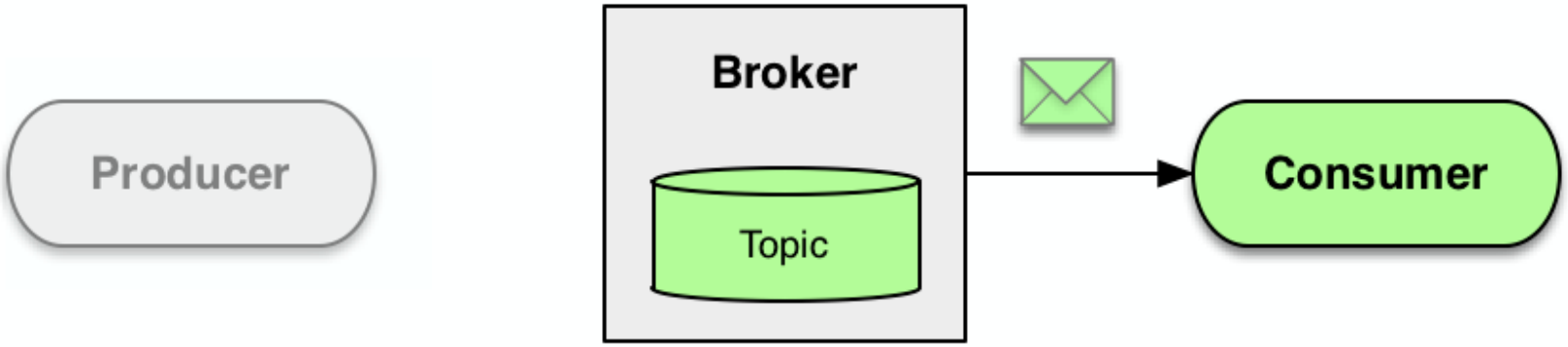
Producers

- **Producers are Kafka clients that publish messages to a topic**
  - Kafka persists messages to disk on receipt
- **Can be configured to retry if sending fails**
- **For efficiency, producers can send a batch of messages to a broker**



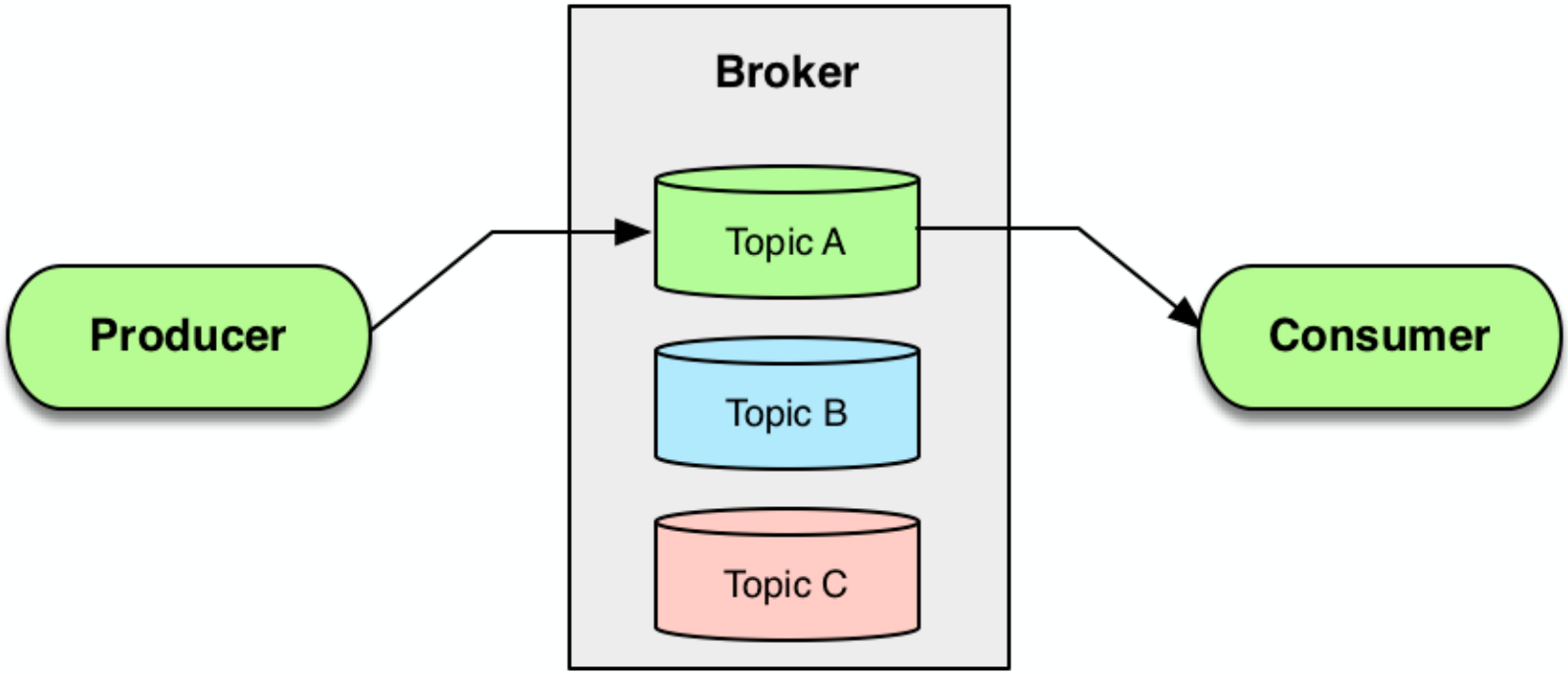
Consumers

- **Consumers are Kafka clients that fetch messages from a topic**
  - They poll the broker for messages
  - Producers and consumers are decoupled from one another
    - Each communicates only with brokers, not each other



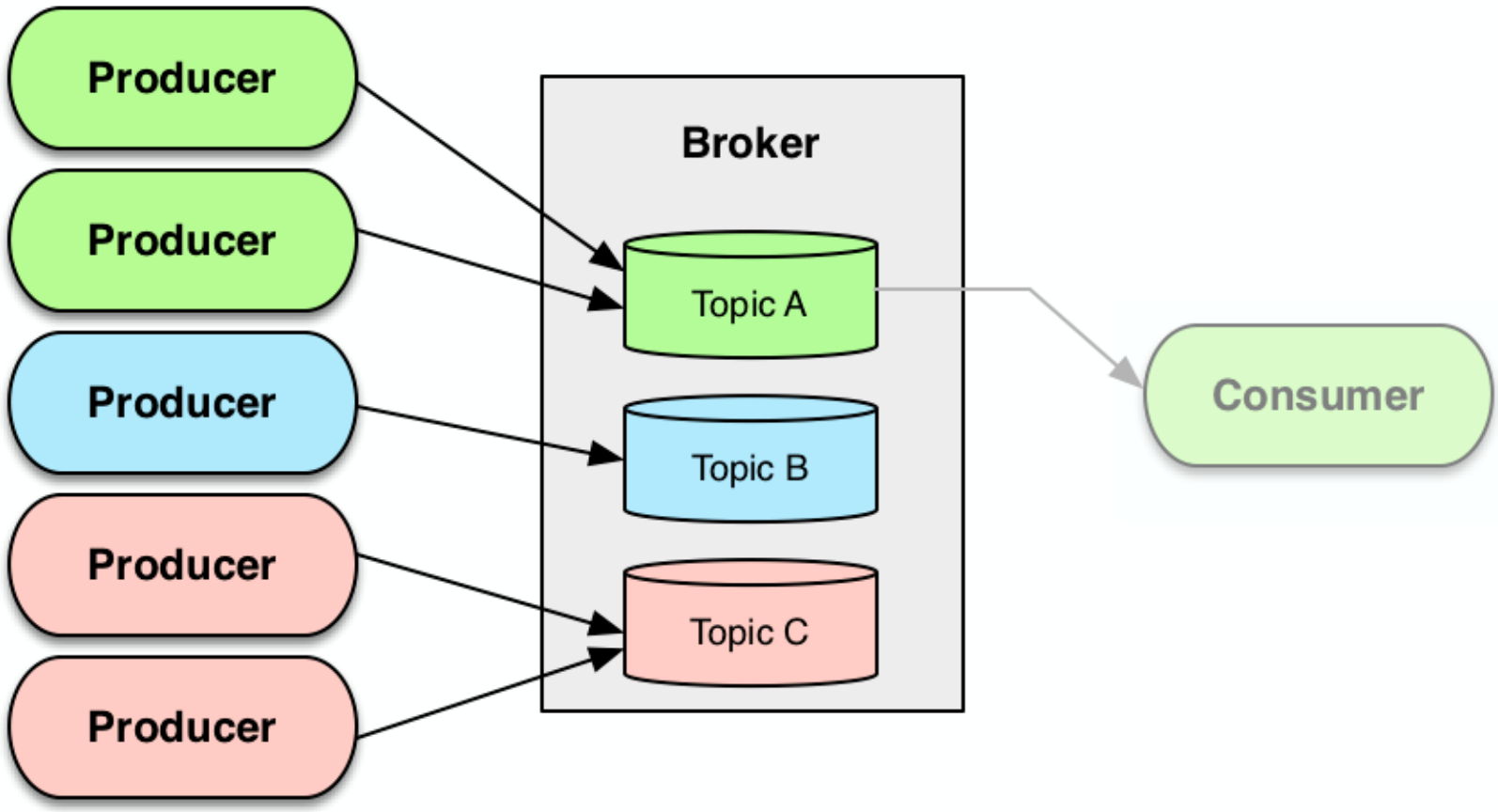
Multiple Topics

- **Each broker may have responsibility for multiple topics**



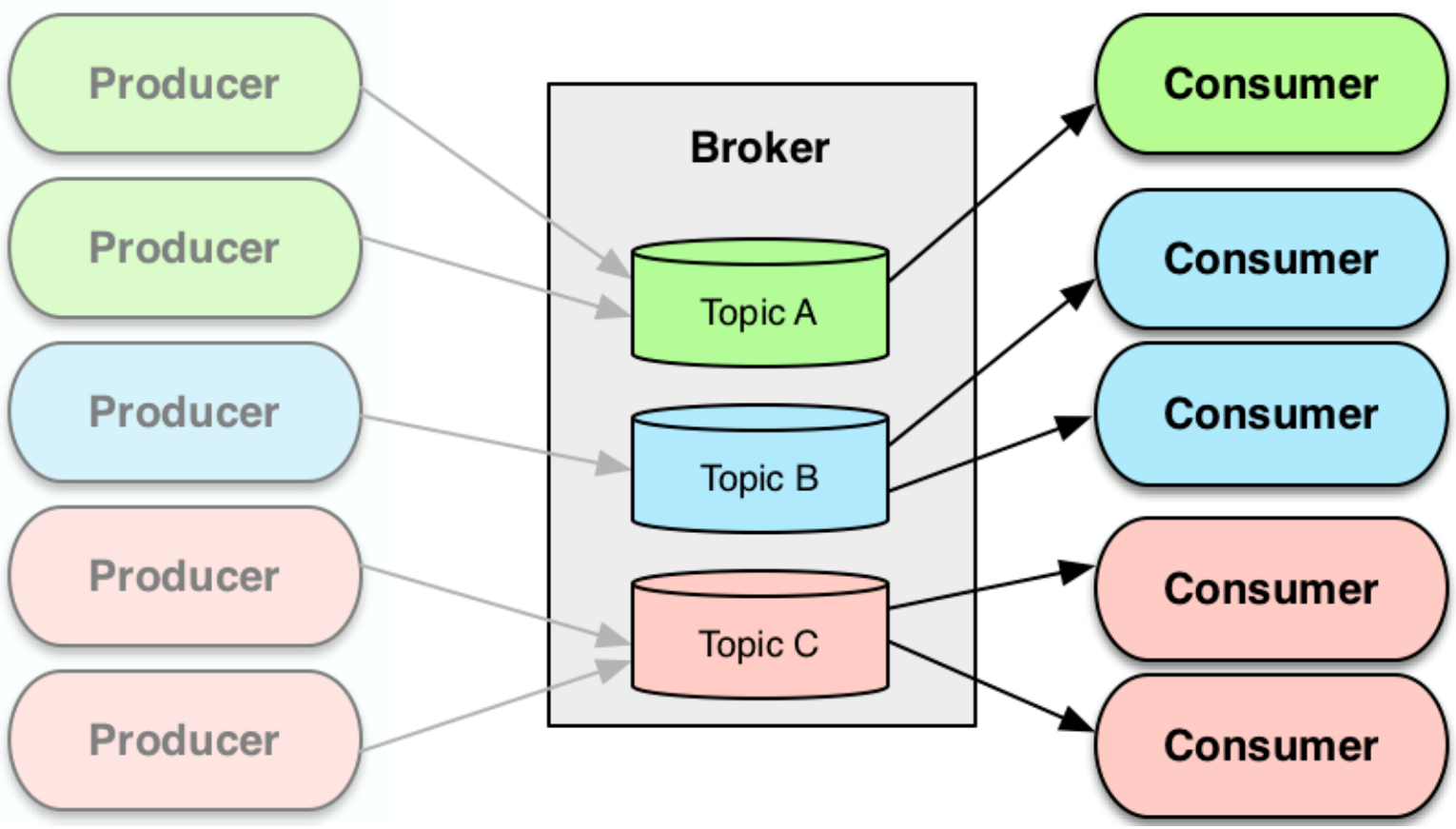
Multiple Producers

- **Each topic may have many producers publishing messages to it**



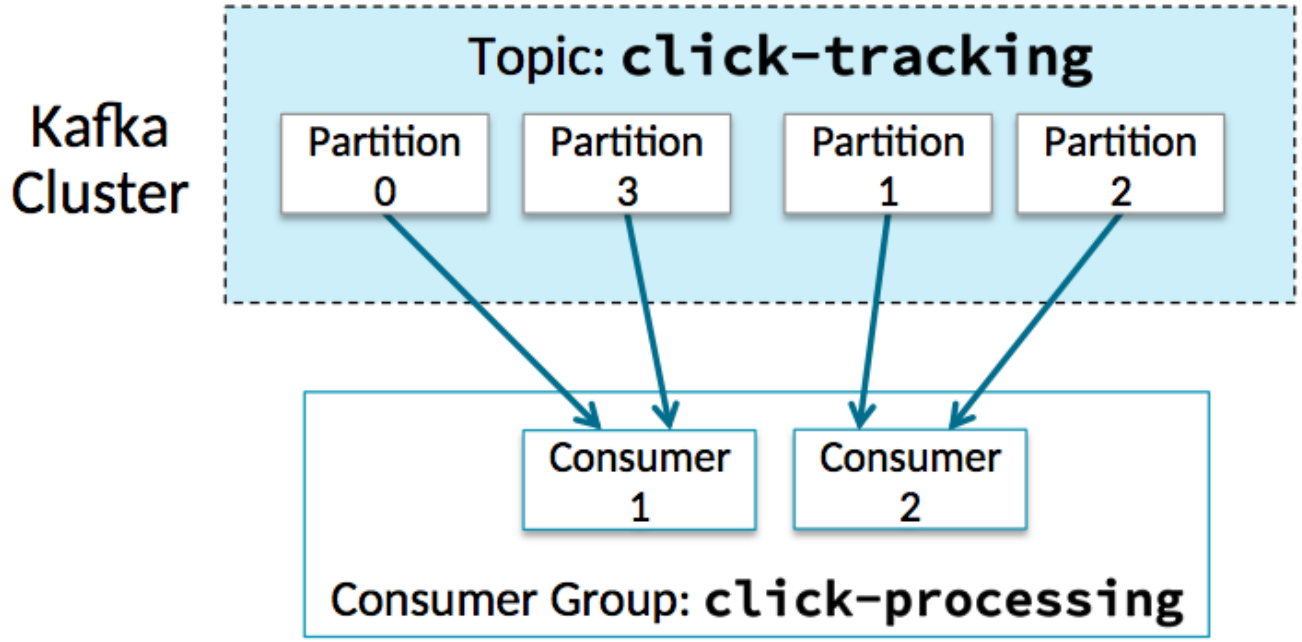
Multiple Consumers

- **Likewise, multiple consumers can read from each topic**
  - Consuming a message does not cause it to be removed



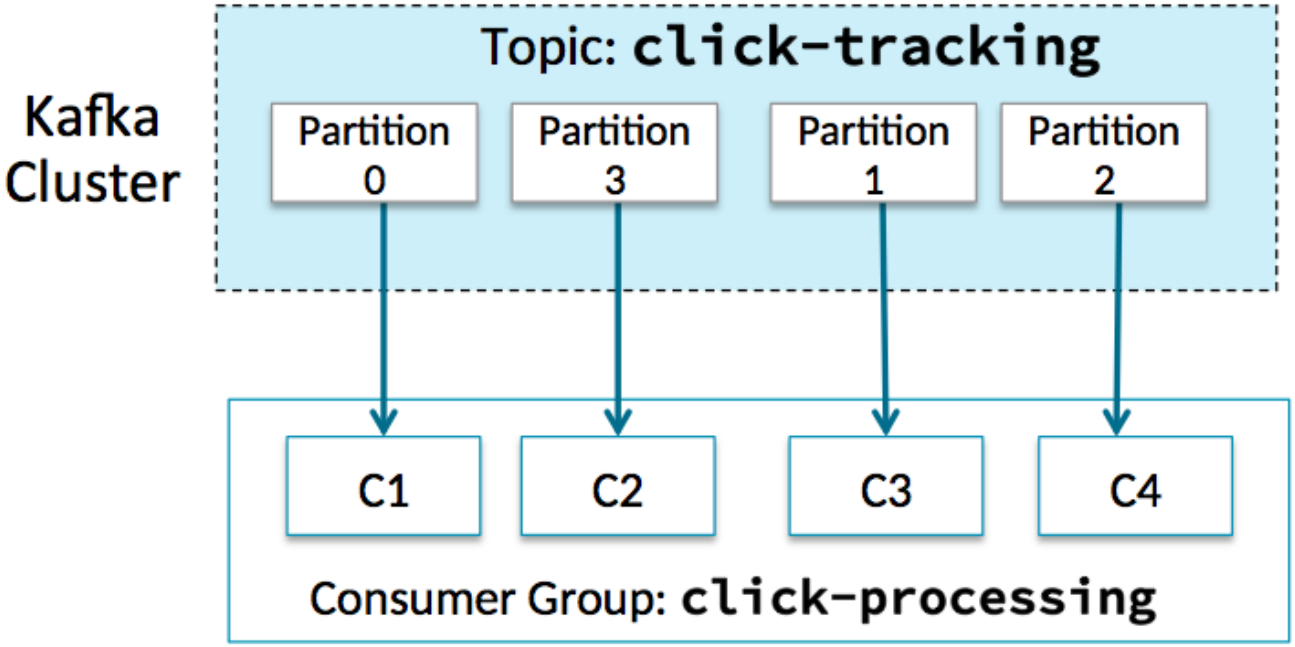
Consumer Groups

- **One or more consumers can form their own consumer group**
  - Working together to consume the messages in a topic
- **Each partition is consumed by only one member of a consumer group**
- **Message ordering is preserved per partition, but not across the topic**



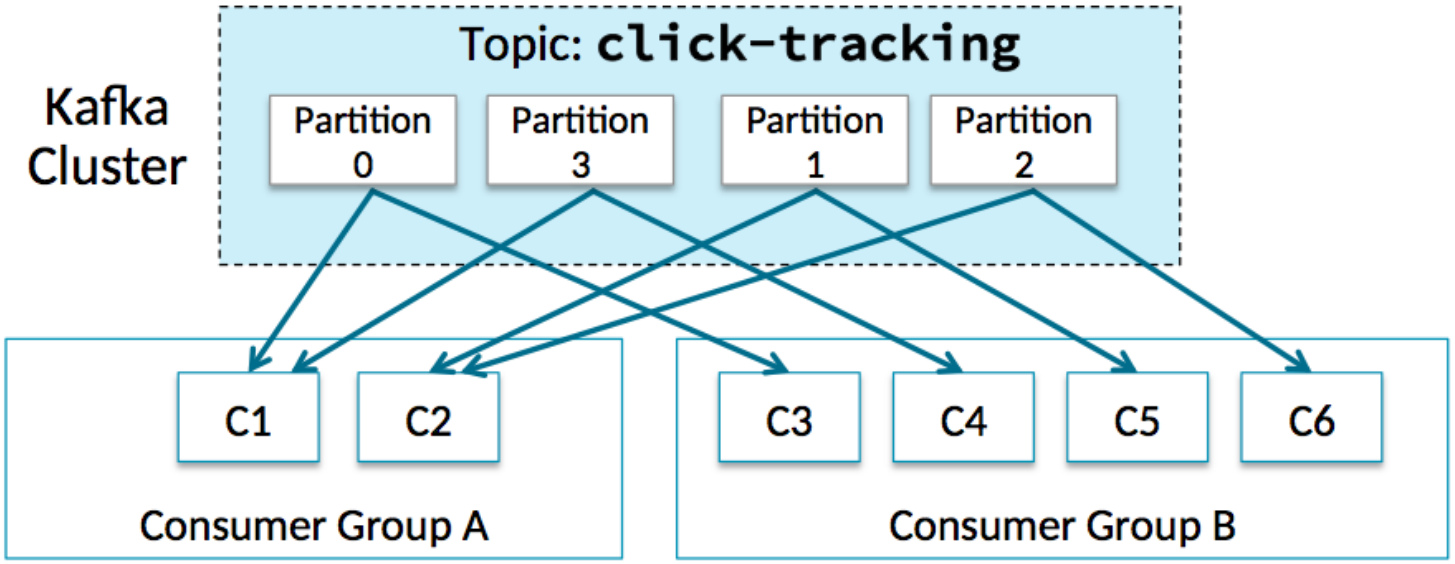
Increasing Consumer Throughput

- Additional consumers can be added to scale consumer group processing
- Consumer instances that belong to the same consumer group can be in separate processes or on separate machines



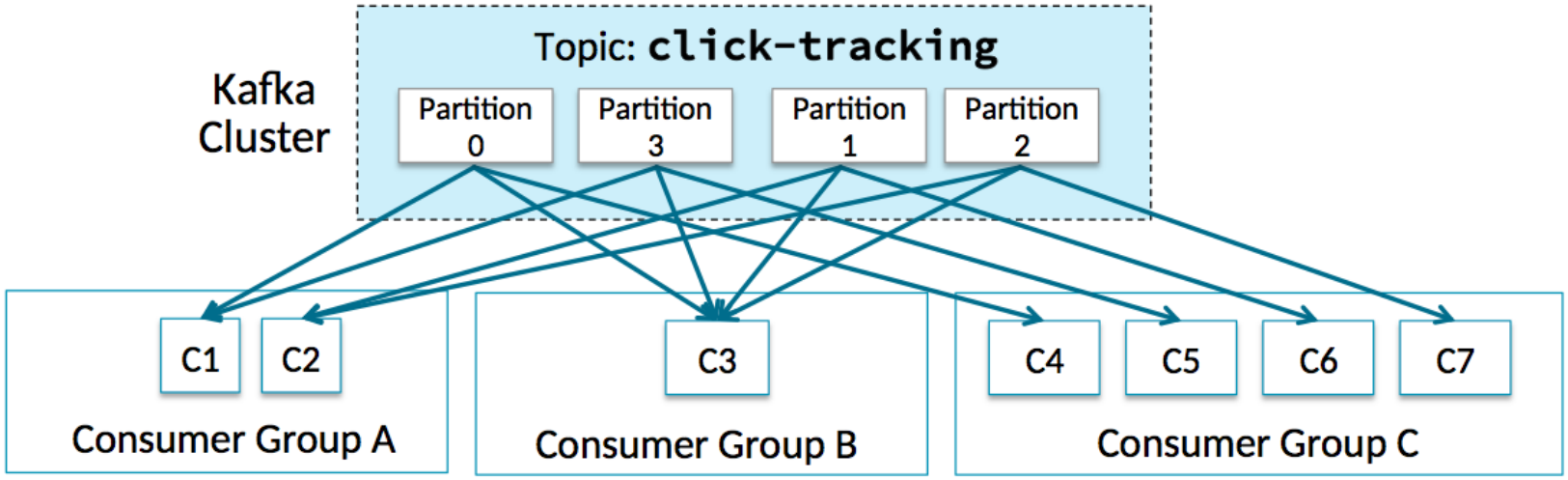
Multiple Consumer Groups

- **Each message published to a topic is delivered to one consumer instance within each subscribing consumer group**
- **Kafka scales to large numbers of consumer groups and consumers**



Publish and Subscribe to Topic

- **Kafka functions like a traditional queue when all consumer instances belong to the same consumer group**
  - In this case, a given message is received by one consumer
- **Kafka functions like traditional publish-subscribe when each consumer instance belongs to a different consumer group**
  - In this case, all messages are broadcast to all consumer groups



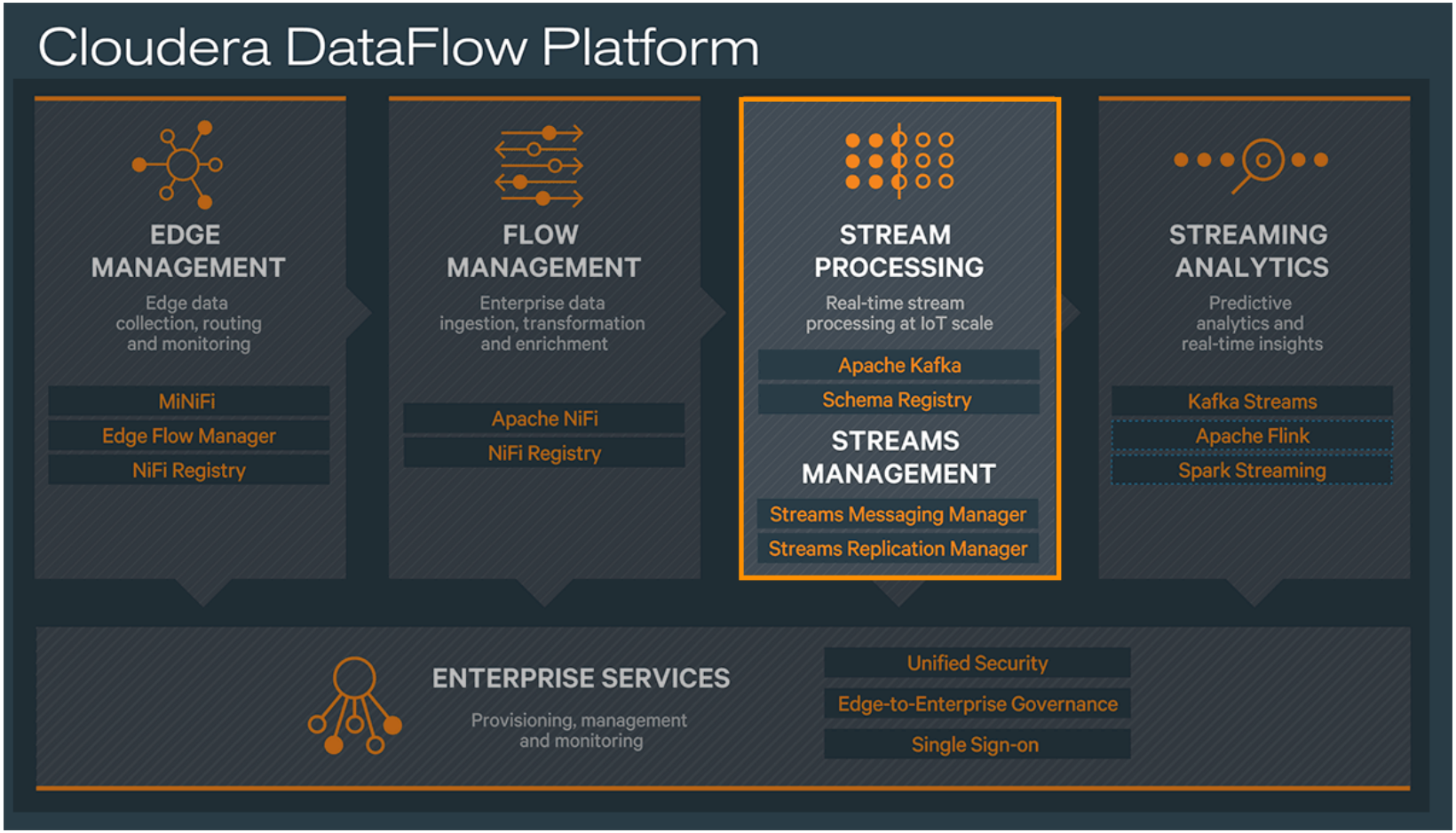
Chapter Topics

Apache Kafka Basics

- Apache Kafka Overview
- Messages and Topics
- Producers and Consumers
- **Kafka in Context**
- Command Line Tools
- Essential Points
- Hands-On Exercise: Using the Apache Kafka Command-Line Tools

Kafka with Cloudera

- **Kafka is part of Cloudera Stream Processing (CSP)**
  - A key part of Cloudera Data Flow—Cloudera's "Data in Motion" solution



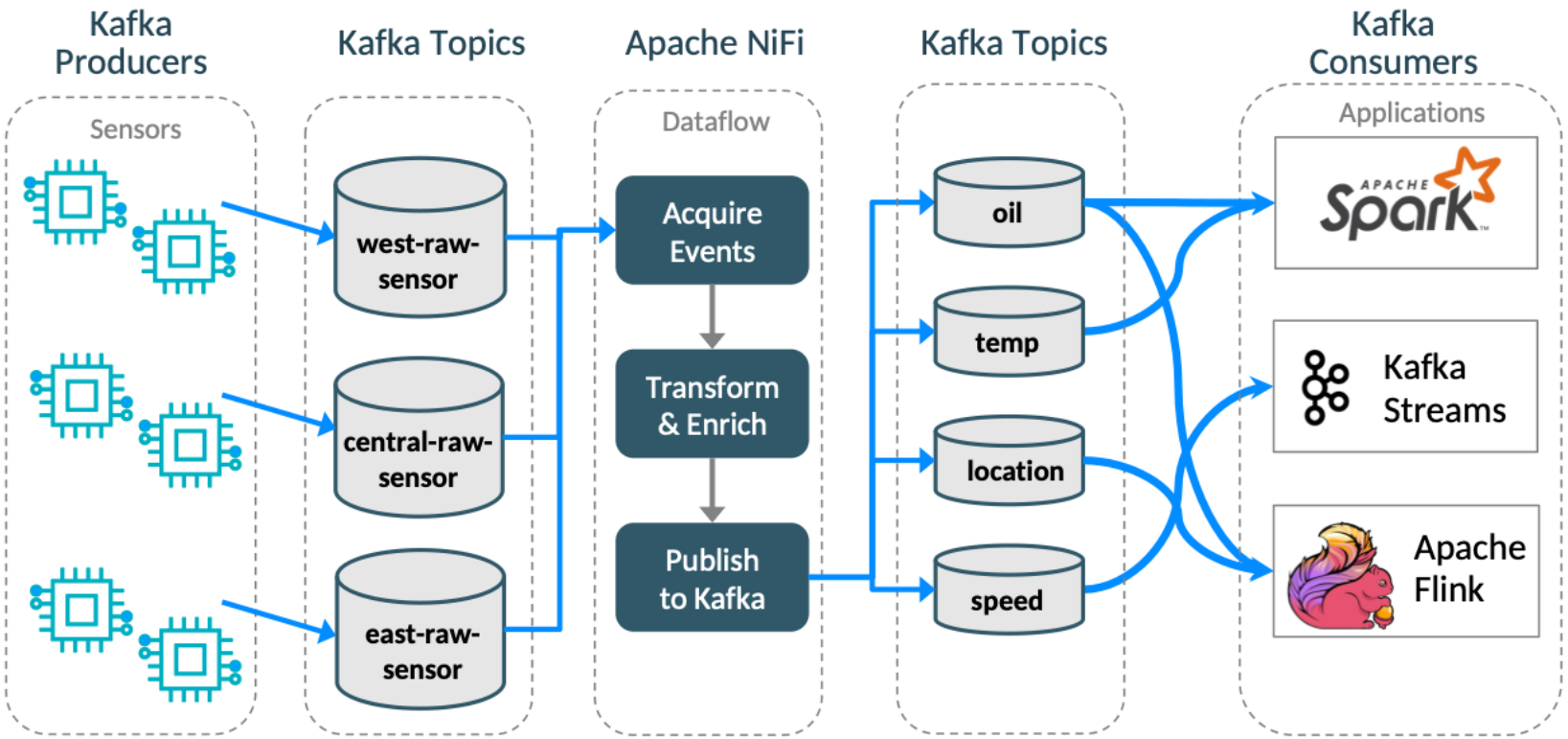
Producer and Consumer Client Applications

- **Tools available as part of Kafka**
  - Command-line producer and consumer tools
  - Kafka Client base Java APIs
  - Kafka Streams Java APIs
- **A growing number of other APIs are available from third parties**
  - Client libraries in many languages including Python, PHP, C/C++, Go, .NET, and Ruby

Kafka Ecosystem

- **Kafka is integrated with a growing number of other systems to create end-to-end data solutions, such as**
  - Apache NiFi—defines flows for data within a solution
  - Apache Spark—analytics engine for streaming and static data
  - Apache Flink—stream processing framework
  - Apache Kafka Streams—an API for creating applications that work with streaming data
- **Other integrated systems provide operational support, such as**
  - Streams Messaging Manager (SMM)—an operations monitoring and management tool
  - Streams Replication Manager (SRM)—a tool for replication of data across multiple clusters and sites

Example: Truck Fleet Sensor Data Analysis



Chapter Topics

Apache Kafka Basics

- Apache Kafka Overview
- Messages and Topics
- Producers and Consumers
- Kafka in Context
- **Command Line Tools**
- Essential Points
- Hands-On Exercise: Using the Apache Kafka Command-Line Tools

Creating Topics from the Command Line

- **Kafka includes a convenient set of command line tools**
  - These are helpful for exploring and experimentation
- **The `kafka-topics` command offers a simple way to create Kafka topics**
  - Provide the topic name of your choice, such as `device_status`
  - You must also specify one or more Kafka brokers
  - Older versions of Kafka used a list of ZooKeeper instances instead of Kafka brokers

```
$ kafka-topics --create \
  --bootstrap-server brokerhost1:9092,brokerhost2:9092 \
  --replication-factor 3 \
  --partitions 5 \
  --topic device_status
```

Displaying Topics from the Command Line

- **Use the `--list` option to list all topics**

```
$ kafka-topics --list \
  --bootstrap-server localhost:9092
```

- **Use the `--help` option to list all `kafka-topics` options**

```
$ kafka-topics --help
```

Running a Producer from the Command Line (1)

- **You can run a producer using the `kafka-console-producer` tool**
- **Specify one or more brokers in the `--broker-list` option**
  - Each broker consists of a hostname, a colon, and a port number
  - If specifying multiple brokers, separate them with commas
- **You must also provide the name of the topic**

```
$ kafka-console-producer \
  --broker-list brokerhost1:9092,brokerhost2:9092  \
  --topic device_status
```

Running a Producer from the Command Line (2)

- **You may see a few log messages in the terminal after the producer starts**
- **The producer will then accept input in the terminal window**
  - Each line you type will be a message sent to the topic
- **Until you have configured a consumer for this topic, you will see no other output from Kafka**

Writing File Contents to Topics Using the Command Line

- **Using UNIX pipes or redirection, you can read input from files**
  - The data can then be sent to a topic using the command line producer
- **This example shows how to read input from a file named `alerts.txt`**
  - Each line in this file becomes a separate message in the topic

```
$ cat alerts.txt | kafka-console-producer \
  --broker-list brokerhost1:9092,brokerhost2:9092 \
  --topic device_status
```

- **This technique can be an easy way to integrate with existing programs**

Running a Consumer from the Command Line

- **You can run a consumer with the `kafka-console-consumer` tool**
- **Requires one or more bootstrap servers (Kafka brokers)**
  - Older versions used ZooKeeper instances instead
- **The command also requires a topic name**
- **Use `--from-beginning` to read *all* available messages**
  - Otherwise, it reads only *new* messages

```
$ kafka-console-consumer \
  --bootstrap-server brokerhost:9092 \
  --topic device_status \
  --from-beginning
```

Chapter Topics

Apache Kafka Basics

- Apache Kafka Overview
- Messages and Topics
- Producers and Consumers
- Kafka in Context
- Command Line Tools
- **Essential Points**
- Hands-On Exercise: Using the Apache Kafka Command-Line Tools

Essential Points

- **Kafka is a distributed platform for streaming data**
  - Messages (records) are categorized by topic
  - Producers send messages to a topic
  - Consumers read messages from a topic
- **Nodes in a cluster running the Kafka service are called brokers**
- **Scale Kafka by using**
  - Partitions—distributed buckets for messages within a topic
  - Consumer groups—multiple consumers working together to process messages from the same topic to increase throughput
- **Partitions are replicated across multiple nodes for fault tolerance**
- **Kafka includes command-line tools for managing topics, and for starting producers and consumers**

Bibliography

The following offer more information on topics discussed in this chapter

- Apache Kafka Web Site
- Cloudera's Apache Kafka Guide
- List of Organizations Using Kafka
- Original LinkedIn Blog Article about Kafka

Chapter Topics

Apache Kafka Basics

- Apache Kafka Overview
- Messages and Topics
- Producers and Consumers
- Kafka in Context
- Command Line Tools
- Essential Points
- **Hands-On Exercise: Using the Apache Kafka Command-Line Tools**

Hands-On Exercise: Using the Apache Kafka Command-Line Tools

- In this exercise, you will use the Kafka command-line tools to pass messages from a producer to a consumer
- Please refer to the Hands-On Exercise Manual for instructions