Data Stream Processing

Day 2.
Kafka Basics

+ Diane Woodbridge, PH.D

UNIVERSITY OF SAN FRANCISCO

CHANGE THE WORLD FROM HERE

Announcement

Individual Assignment - Due on Oct 26th

- Project Ideation
- Deliverable : **Submit a 2-4 page slide**
    - Project Idea and Goal
    - Data Source and Stream Design
    - Producers
    - Consumers and Analysis Goals

Office Hours

- Tue - 9 - 10 AM (Virtual)
- Thu - 5 - 5:20 PM (In Person)

Review

Intro of Data Stream Processing

Batch Processing vs. Stream Processing

Stream Processing Applications

- Spotify Data Stream
- Surveillance Camera Data Stream
- Basketball Play Data Stream

Contents

Messaging System and Pub/Sub Paradigm

Kafka Architecture

- Producer
- Consumer
- Topic
  - Messages
- Kafka Cluster
  - Partitions
  - Replications

Confluent

- Create a cluster, topic, and produce/subscribe messages.

Contents

Messaging System and Pub/Sub Paradigm

Kafka Architecture

- Producer
- Consumer
- Topic
  - Messages
- Kafka Cluster
  - Partitions
  - Replications

Confluent

- Create a cluster, topic, and produce/subscribe messages.

Messaging System

Why messaging systems?

- Decouple data producers and consumers.
- Handle bursts of data gracefully.
- Enable scalable, fault-tolerant pipelines.

Example:
Web servers → Kafka → Analytics service → Database

Pub–Sub Paradigm in Messaging System

Pub-Sub Paradigm

- A communication pattern where producers (publishers) send messages to topics, and consumers (subscribers) receive messages from those topics — decoupled in time and logic.
- **Benefits**
  - **Decoupling**: Publishers and subscribers operate independently.
  - **Scalability**: Multiple producers/consumers can produce/consume in parallel.
  - **Asynchronicity**: Real-time data flow without direct request–response.
  - **Fault tolerance**: Messages can persist even if consumers are temporarily offline.

Apache Kafka

A high-throughput, fault-tolerant, distributed messaging system designed for real-time data pipelines and stream processing.

- In a pub/sub system, Kafka acts as a broker, that stores data and handles requests.
- Written in Scala, and Java.

"I thought that since Kafka was a system optimized for writing using, a writer's name would make sense. I had taken a lot of lit classes in college and liked Franz Kafka."

- Jay Kreps

Contents

Messaging System and Pub/Sub Paradigm

Kafka Architecture

- Producer
- Consumer
- Topic
  - Messages
- Kafka Cluster
  - Partitions
  - Replications

Confluent

- Create a cluster, topic, and produce/subscribe messages.

Contents

Messaging System and Pub/Sub Paradigm

Kafka Architecture

- **Producer**
- **Consumer**
- Topic
    - Messages
- Kafka Cluster
    - Partitions
    - Replications

Confluent

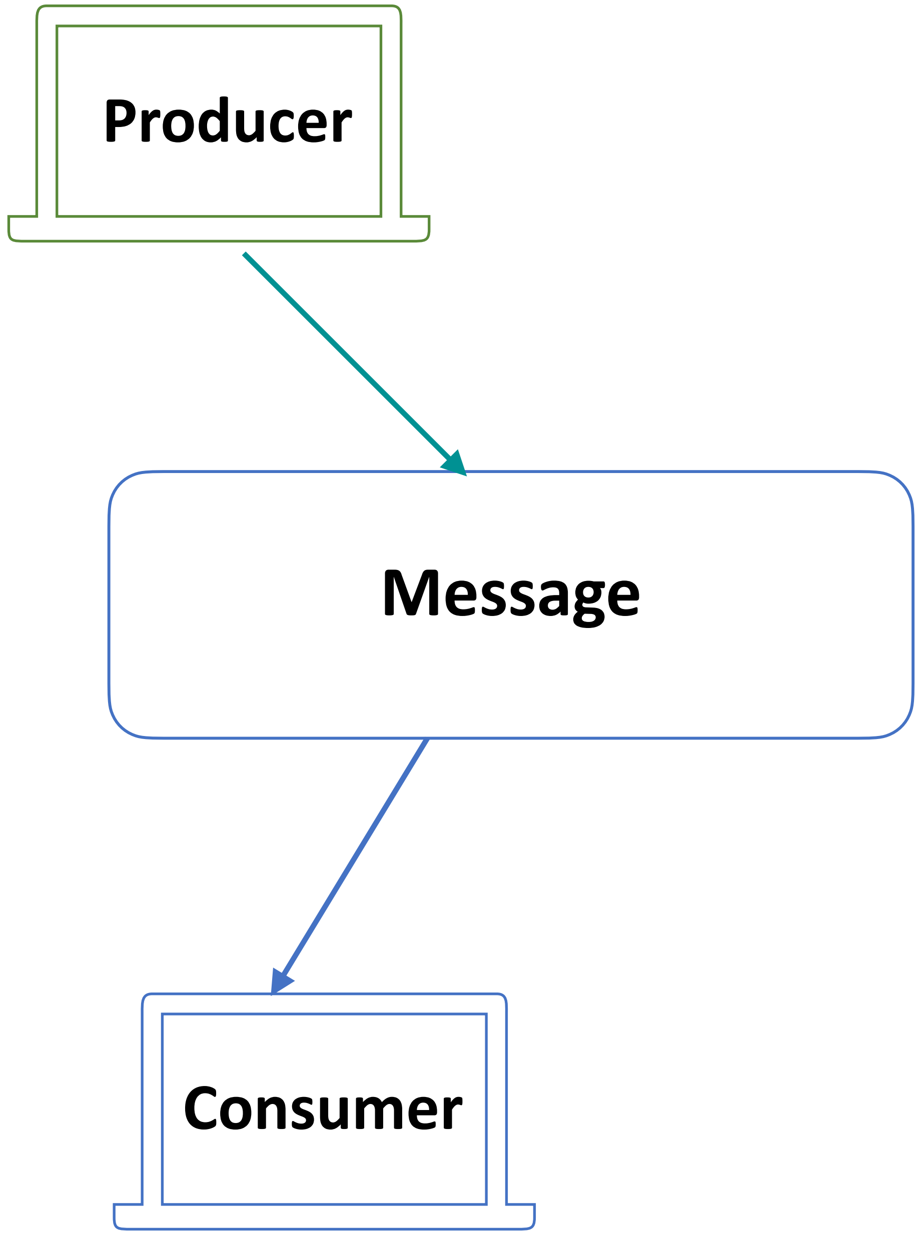
- Create a cluster, topic, and produce/subscribe messages.

Kafka

Architecture

- Message
  - The fundamental unit of data in Kafka, also known as a record.
  - Represents a single piece of data using a key, a value, and a timestamp, and optionally, headers.
- Producer
  - Client applications that publish (write) messages
- Consumer
  - Applications or systems that subscribe to (read and process) these messages

- In Kafka, producers and consumers are fully decoupled and agnostic of each other, which is a key design element to achieve the high scalability.



Contents

Messaging System and Pub/Sub Paradigm

Kafka Architecture

- Producer
- Consumer
- **Topic**
  - **Messages**
- Kafka Cluster
  - Partitions
  - Replications

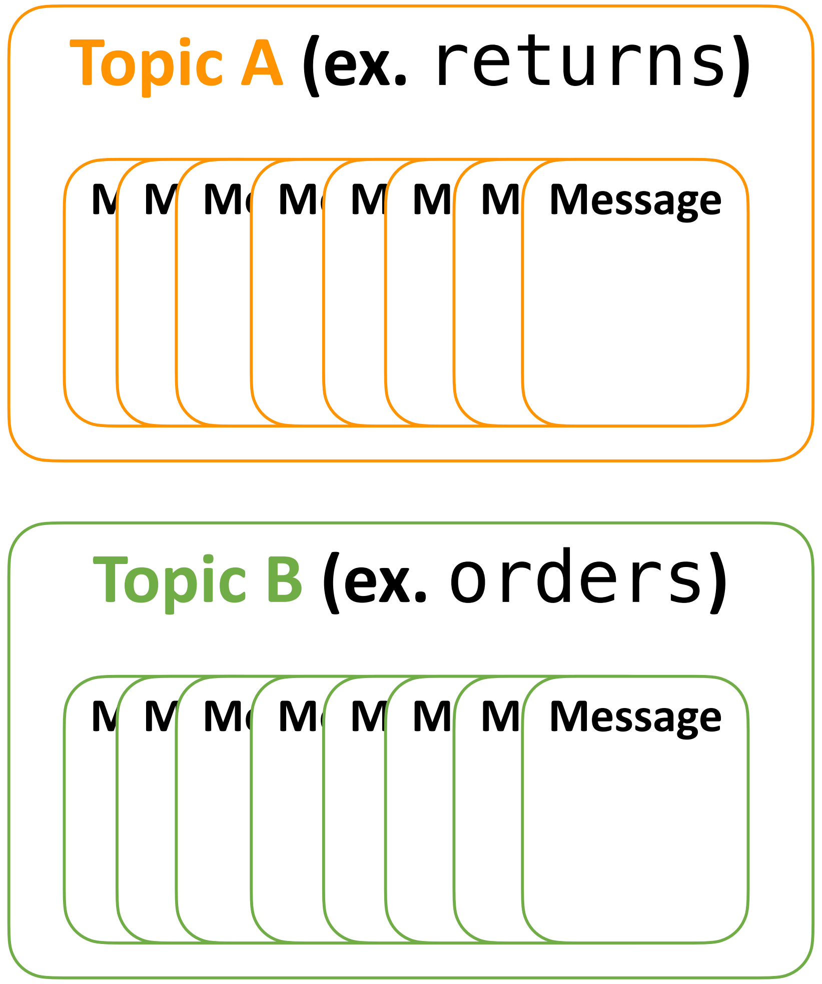
Confluent

- Create a cluster, topic, and produce/subscribe messages.

Kafka

Architecture

- **Messages**
  - The fundamental unit of data in Kafka.
  - Represents a single piece of data using a key, a value, and a timestamp, and optionally, headers.
- **Topics**
  - A category or feed name for organizing and storing messages in Kafka.
  - Mechanism for organizing and storing messages in Kafka.
    - Messages are the actual data units that are contained within topics.



Kafka Topics – Naming Conventions

Recommended Practices

- **Lowercase only**: Use lowercase letters for consistency.
- **Descriptive**: Name should reflect what the topic contains.
  - Example: `sales.orders.transactions`
- **Hierarchical naming**: Use dot notation to indicate structure.
  - Example: `department.service.entity`, `sales.orders.transactions`
- **Environment prefix/suffix**: Differentiate dev/test/prod if sharing clusters.
  - Example: `dev.sales.orders.transactions`
- **Versioning**: Add version numbers when schemas evolve.
  - Example: `sales.orders.transactions.v1`

https://www.confluent.io/blog/how-choose-number-topics-partitions-kafka-cluster/

In CAP theorem, what are C, A and P?

Consistency, Atomic, Partition Tolerance

Concurrency, Availability, Persistency

Consistency, Availability, Partition Tolerance

Concurrency, Atomic, Partition Tolerance

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

In CAP theorem, what are C, A and P?

Consistency, Atomic, Partition Tolerance	0%
Concurrency, Availability, Persistency	0%
Consistency, Availability, Partition Tolerance	0%
Concurrency, Atomic, Partition Tolerance	0%

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

In CAP theorem, what are C, A and P?

Consistency, Atomic, Partition Tolerance

0%

Concurrency, Availability, Persistency

0%

Consistency, Availability, Partition Tolerance

0%

Concurrency, Atomic, Partition Tolerance

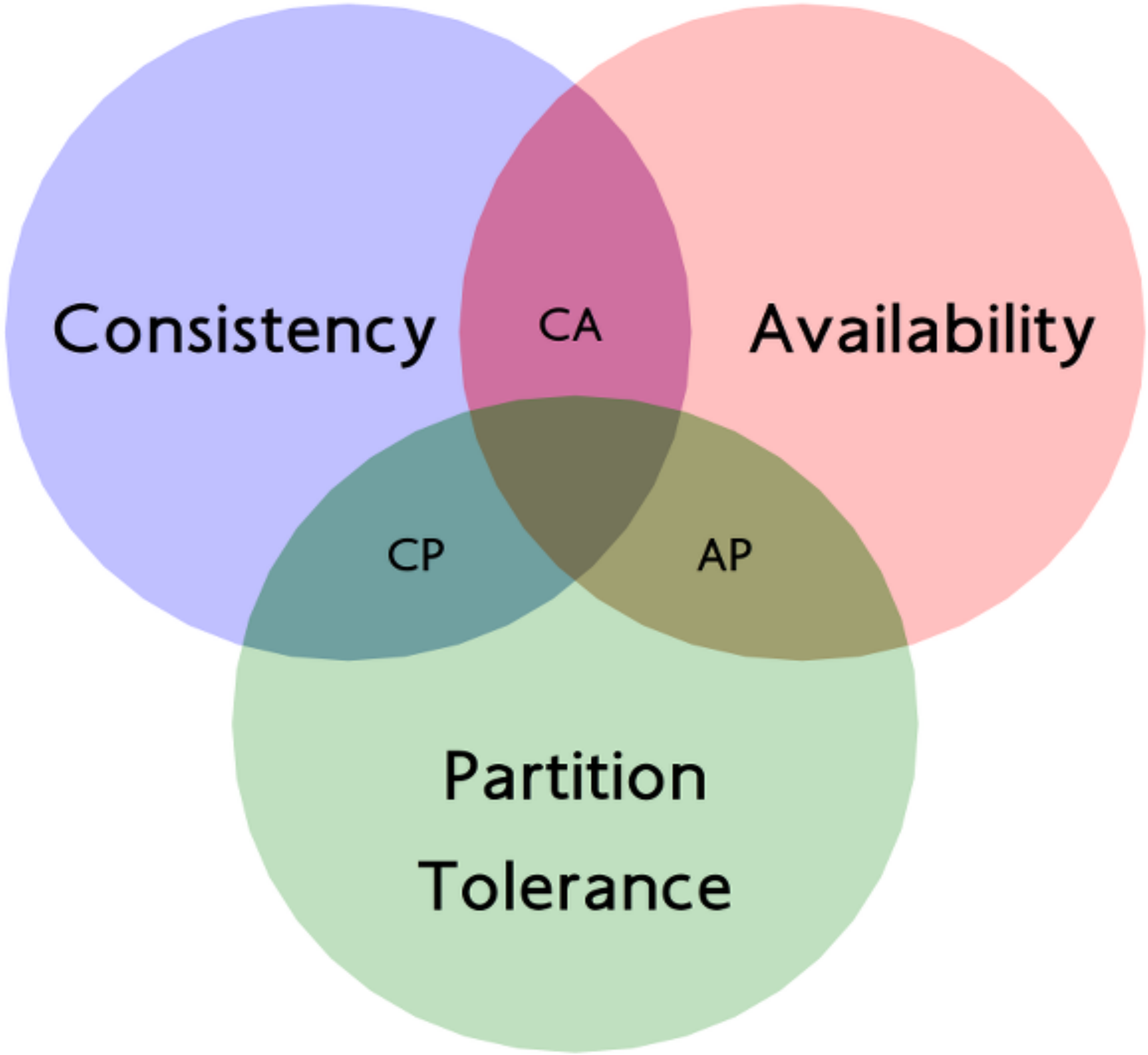
0%

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Recap : CAP Theorem

- CAP Theorem
  1. Consistency
     - All nodes have the most recent data.
  2. Availability
     - Every request received by a non-failing node must return a response.
  3. Partition Tolerance
     - Clusters can survive from communication breakages in the cluster.

➔ You can only get two.



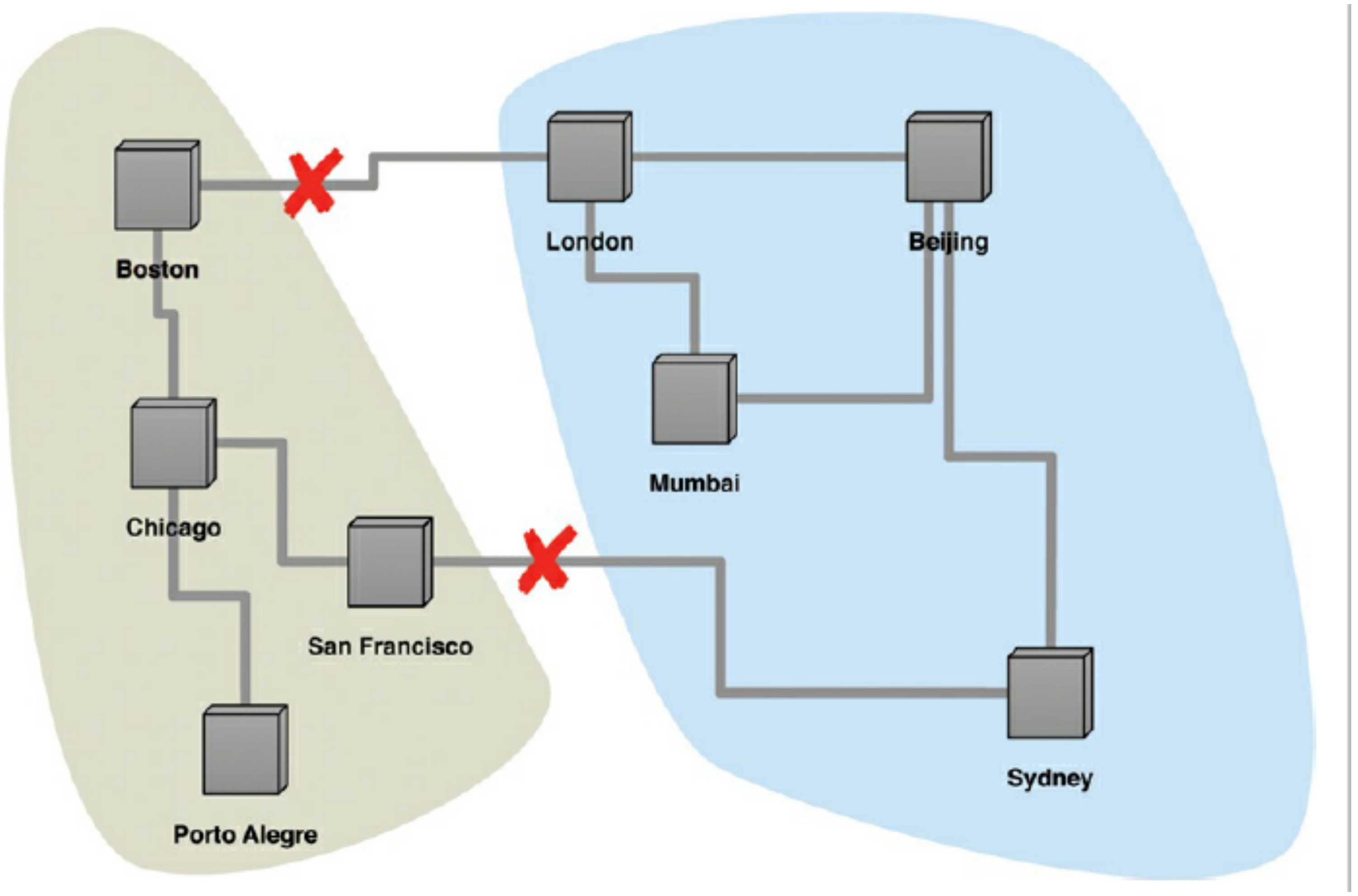
http://blingtechs.blogspot.com/2016/02/cap-theorem.html

Recap : CAP Theorem

- CAP Theorem
  1. Consistency
     - All nodes have the most recent data.
  2. Availability
     - Every request received by a non-failing node must return a response.
  3. Partition Tolerance
     - Clusters can survive from communication breakages in the cluster.

➔ **You can only get two.**

ACID addresses an individual node's data consistency.
CAP addresses cluster-wide data consistency .



http://blingtechs.blogspot.com/2016/02/cap-theorem.html

Contents

Messaging System and Pub/Sub Paradigm

Kafka Architecture

- Producer
- Consumer
- Topic
  - Messages
- **Kafka Cluster**
  - Partitions
  - Replications

Confluent

- Create a cluster, topic, and produce/subscribe messages.

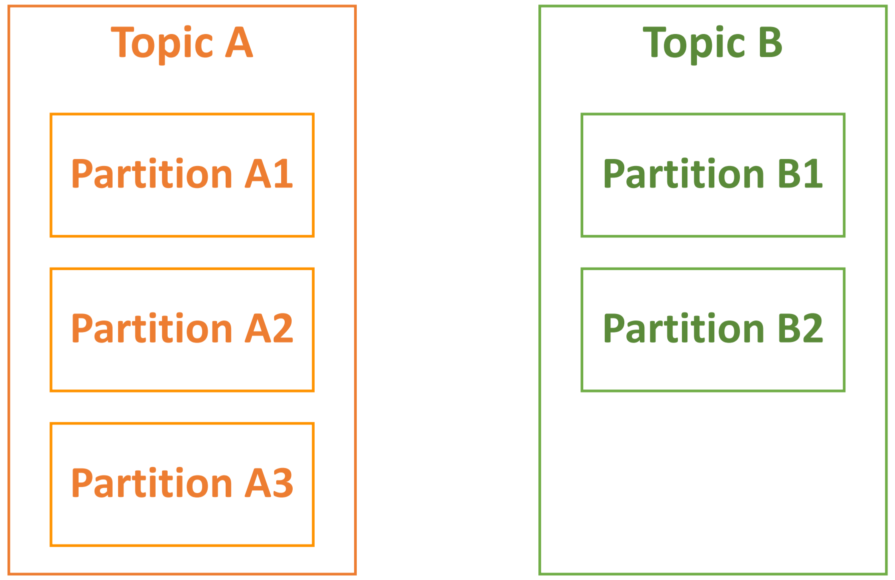
Kafka

Architecture

- Topic - A category or feed name for organizing and storing messages in Kafka.

- Partition - A partition is a fundamental unit of parallelism and scalability within a Kafka topic.
  - Each topic is divided into one or more partitions, and each partition is an ordered, immutable sequence of messages.
    - Using the hash value of the key or, if no key is available, uses a round-robin mechanism.
  - When an message is published to a topic, it is appended to one of its partitions.

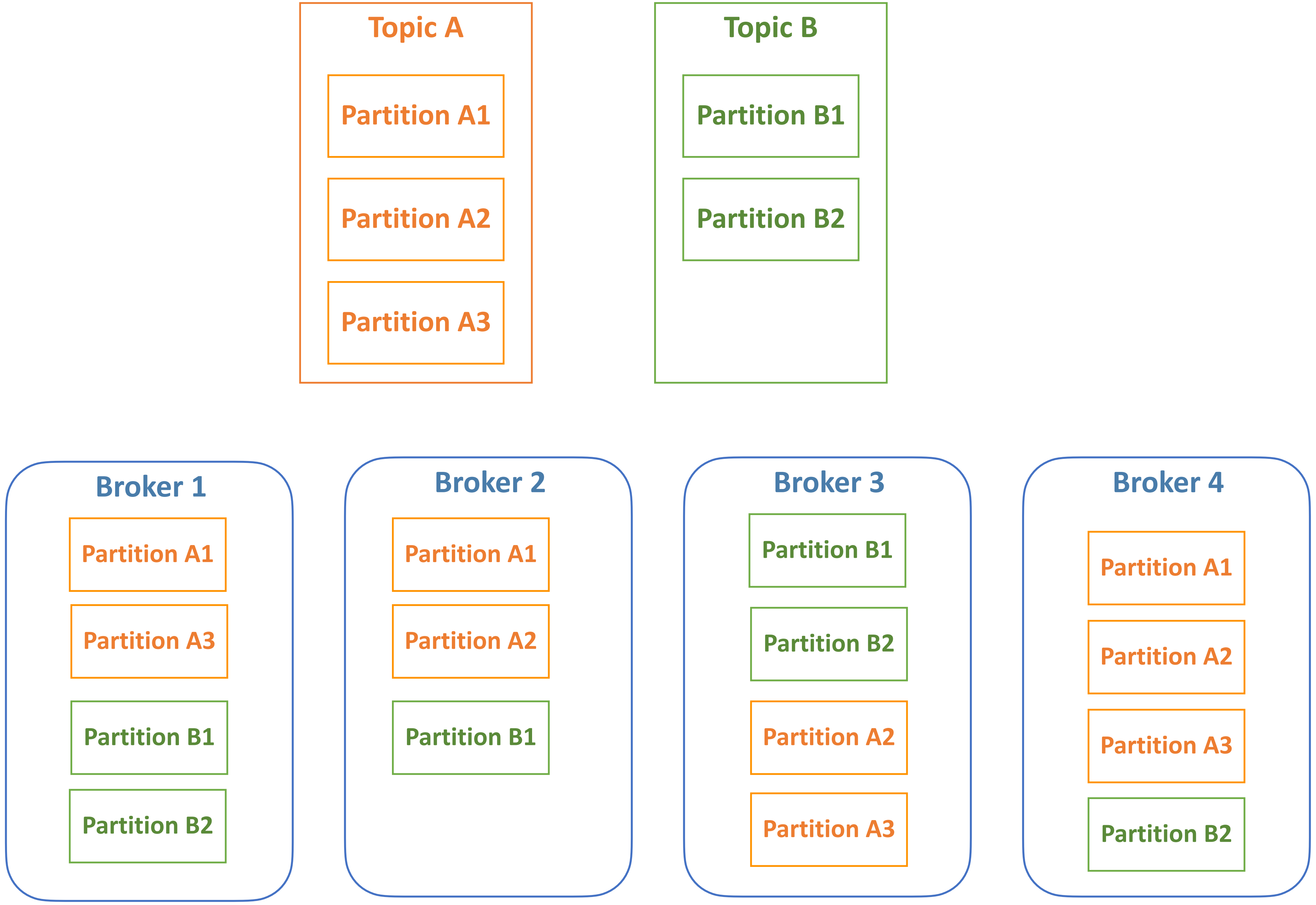
- **Kafka** Cluster - A distributed system consisting of one or more Kafka servers, called brokers.
  - These brokers store the partitions of topics and serve requests from producers and consumers. A cluster provides the underlying infrastructure for storing and managing Kafka topics and their partitions, ensuring data availability and fault tolerance through replication.



Kafka Architecture

Kafka Cluster - A distributed system consisting of one or more Kafka servers, called brokers.

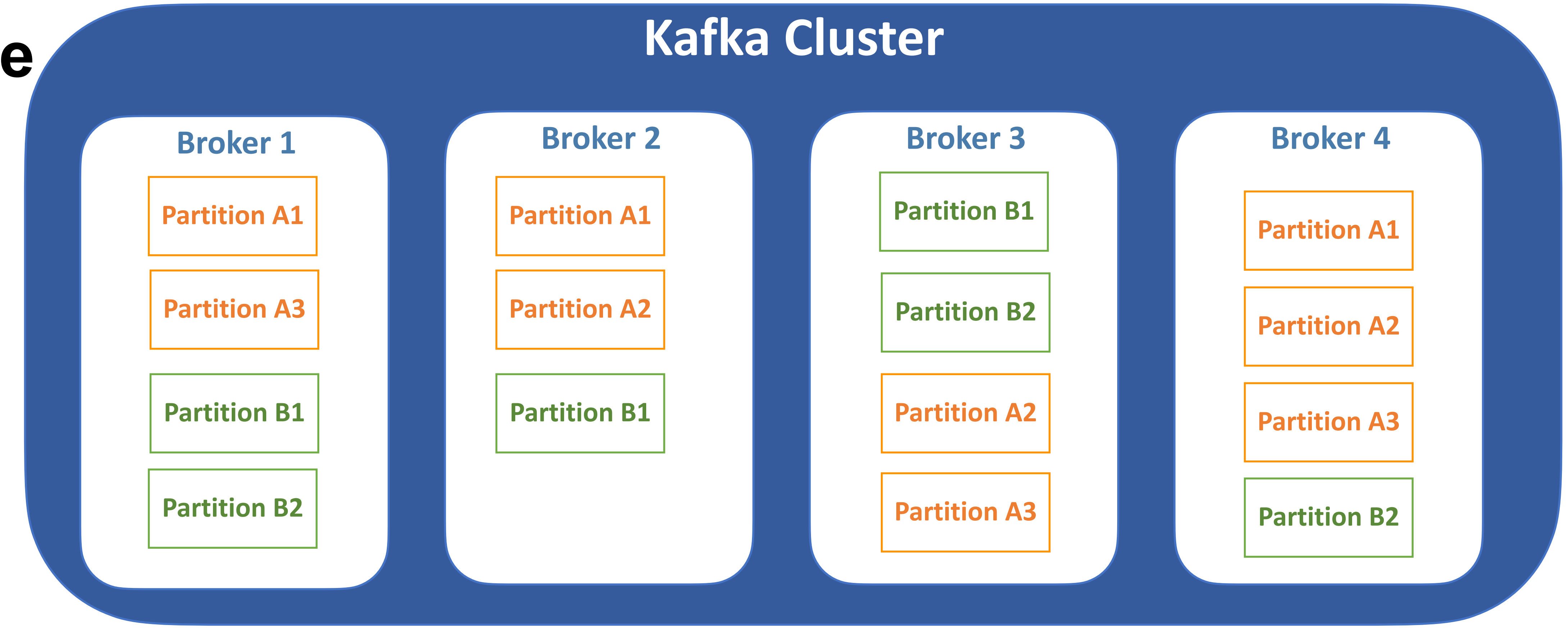
- These brokers store the partitions of topics and serve requests from producers and consumers.
- A cluster provides the underlying infrastructure for storing and managing Kafka topics and their partitions, ensuring data availability and fault tolerance through replication.



Kafka Architecture

Kafka Cluster - A distributed system consisting of one or more Kafka servers, called brokers.

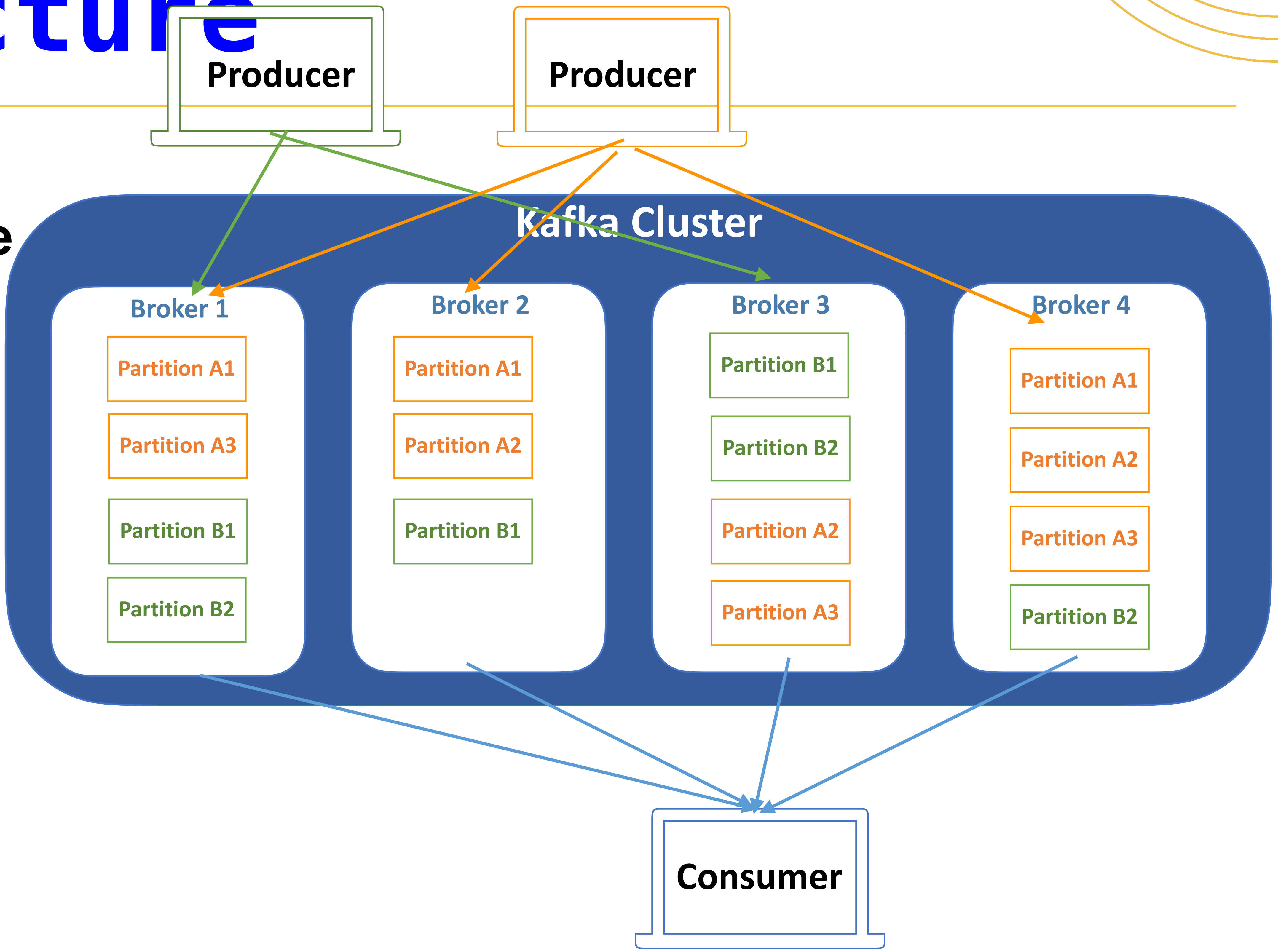
- These **brokers** store the **partitions** of topics and serve requests from producers and consumers.
- A **cluster** provides the underlying infrastructure for storing and managing Kafka topics and their partitions, ensuring data availability and fault tolerance through replication.



Kafka Architecture

Kafka Cluster - A distributed system consisting of one or more Kafka servers, called brokers.

- These brokers store the partitions of topics and serve requests from producers and consumers.
- A cluster provides the underlying infrastructure for storing and managing Kafka topics and their partitions, ensuring data availability and fault tolerance through replication.



Contents

Messaging System and Pub/Sub Paradigm

Kafka Architecture

- Producer
- Consumer
- Topic
  - Messages
- Kafka Cluster
  - **Partitions**
  - Replications

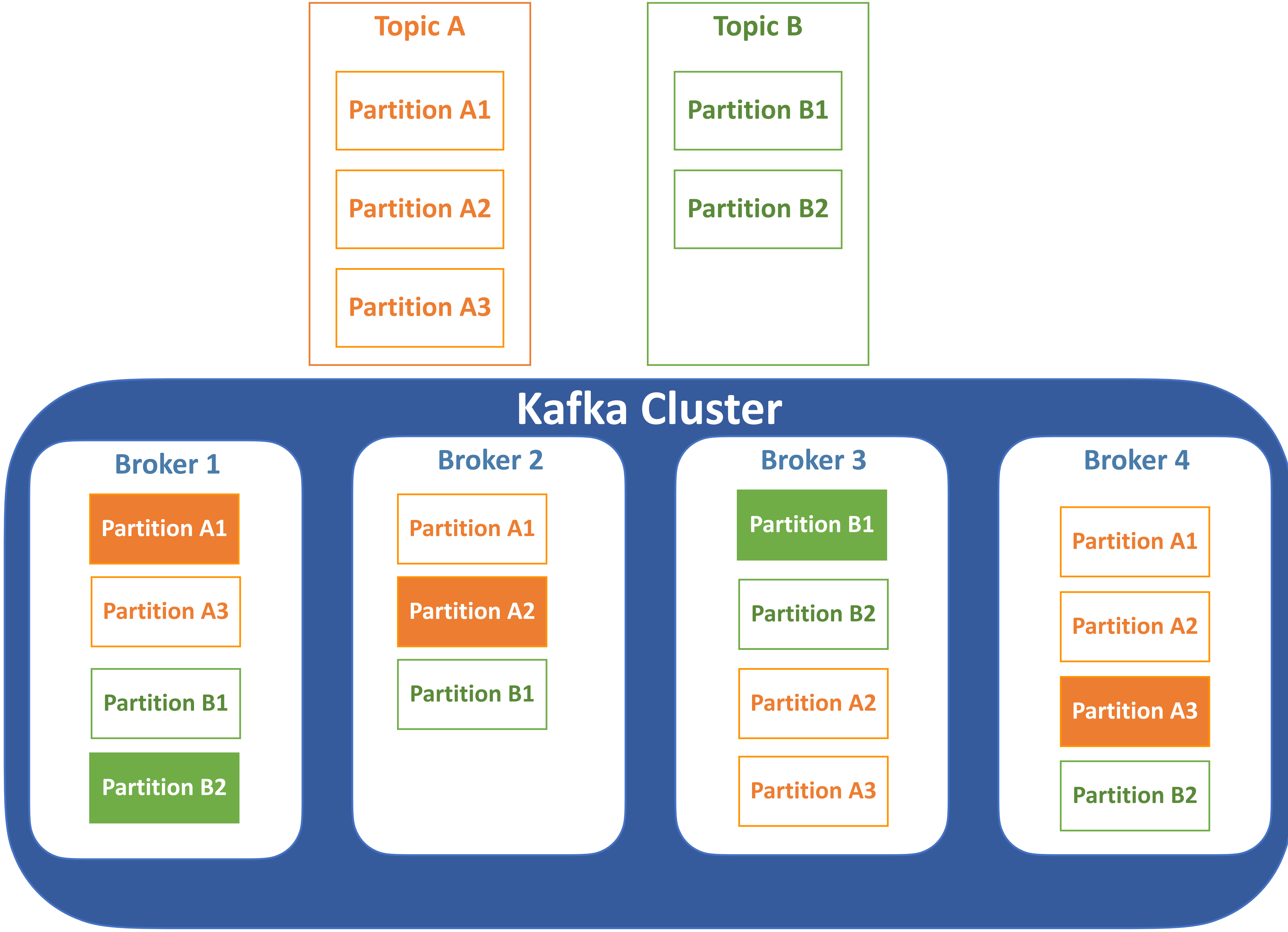
Confluent

- Create a cluster, topic, and produce/subscribe messages.

Kafka Partitions

Partitions : Fundamental unit of parallelism in Kafka.

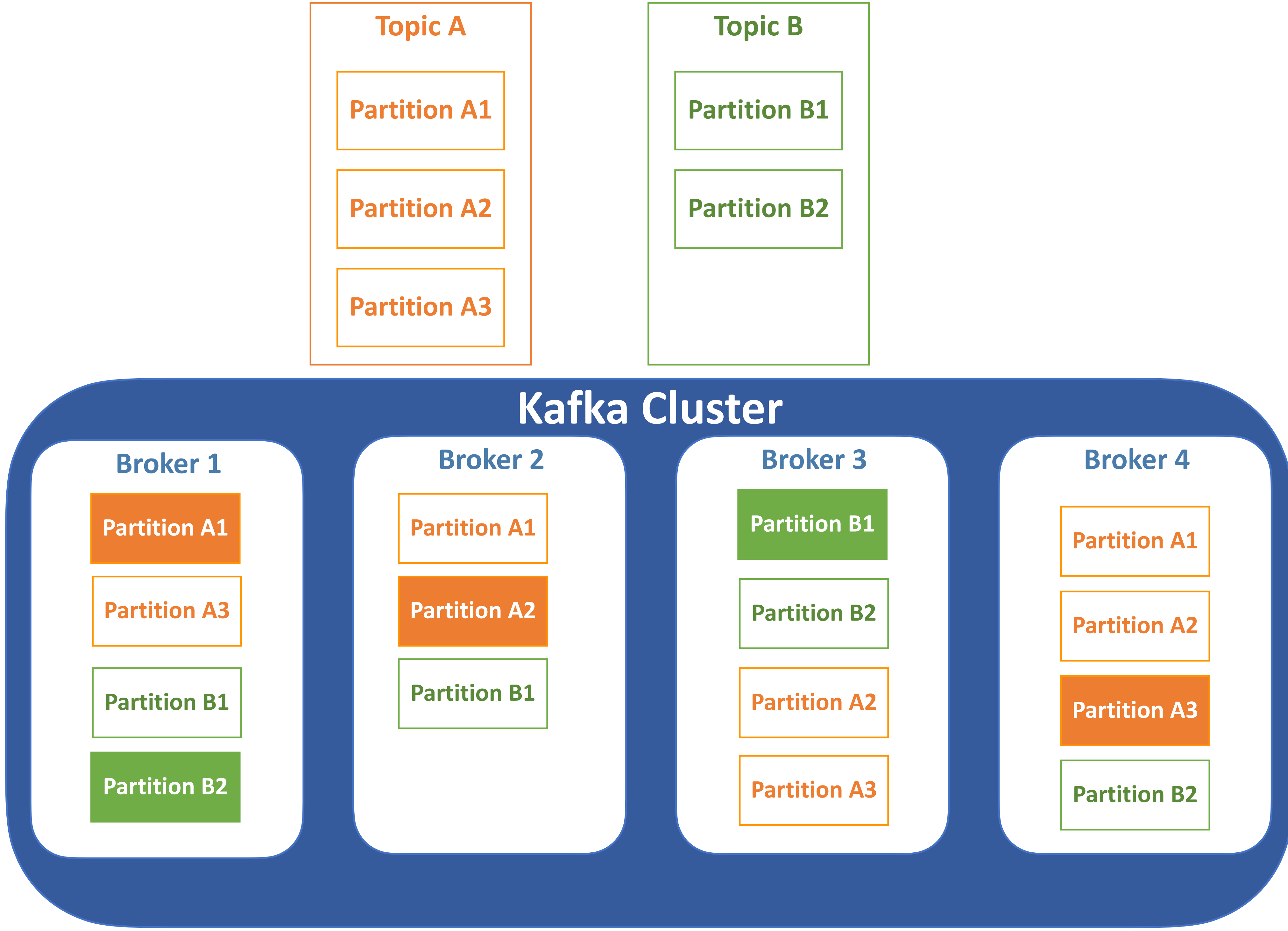
- Each topic is divided into multiple partitions, which enables distributed data processing.
- Replicated across brokers for fault tolerance and high availability.
- Each partition has:
    - One leader (handles reads/ writes).
    - One or more followers (replicas for backup, handles read).



Kafka Partitions

Partition Characteristics

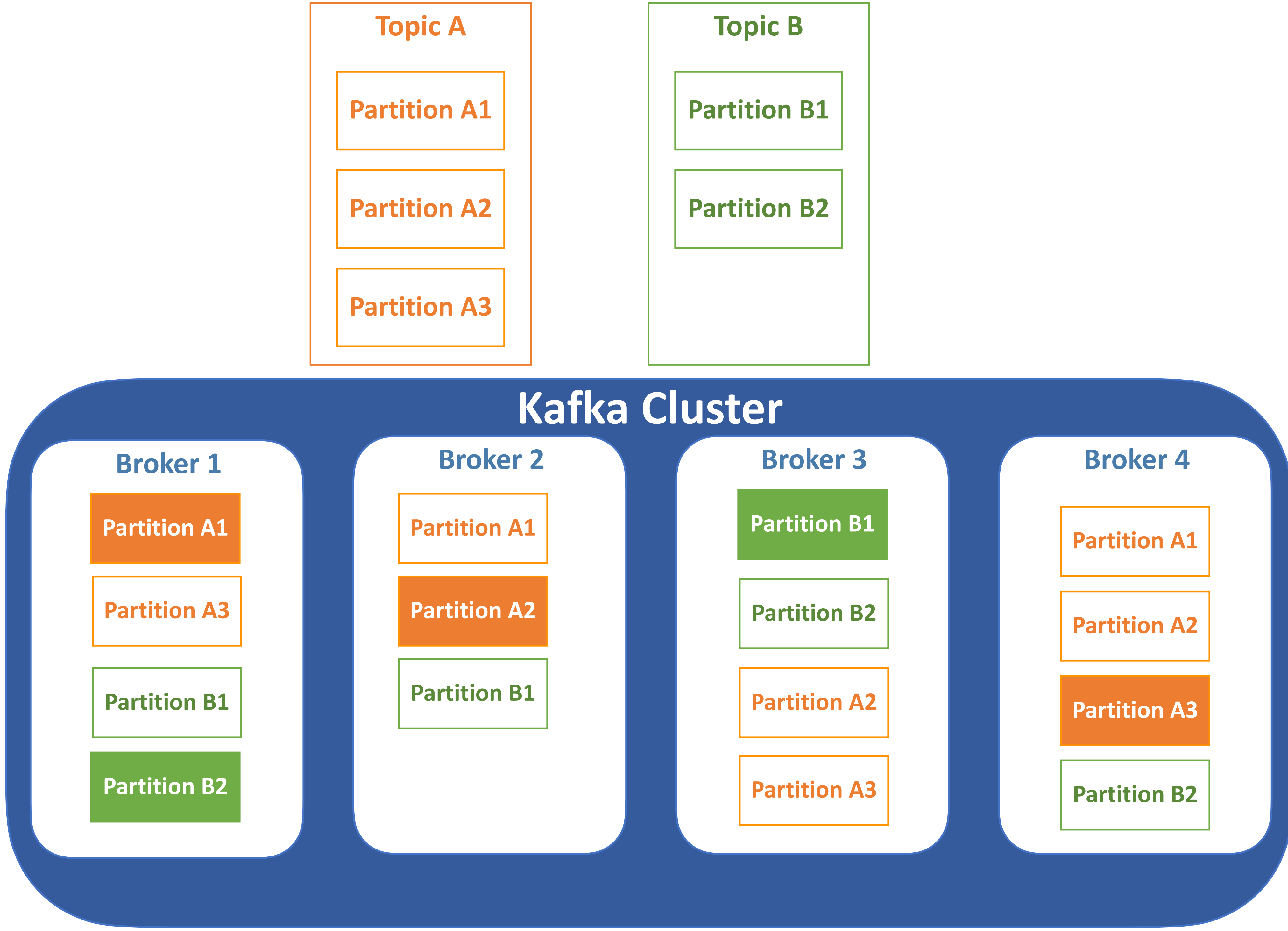
- **Distributed**: Partitions are spread across multiple brokers.
- **Replicated**: Provides redundancy. Copies exist on multiple brokers.
- **Ordered**: Message order is guaranteed only within a partition, not across partitions.
- **Offset**: Each message within a partition has a unique offset ID (used for tracking and recovery).



Kafka Partitions

Benefits

- **Producer Scaling:** Producers can write to multiple partitions simultaneously → Parallel ingestion.
- **Consumer Scaling:** Multiple consumers can read from different partitions concurrently → Higher throughput.
- **Data Distribution:** Data is allocated based on a partition strategy (e.g., hash, round-robin).



Kafka Topics – Partitioning Topics

Choosing the Number of Partitions

- The optimal number of partitions depends on workload and scalability goals.
  - According to Confluent’s best practices:
    - Over-partition to anticipate future growth.
    - Avoid too many partitions per broker — may cause overhead.
    - Continuously monitor metrics and adjust configurations as needed.

Estimating from Throughput Needs

- To determine partition count:
  - Compare producer throughput and consumer throughput.
  - Ensure that neither producers nor consumers are bottlenecked.
  - Example Calculation:
    - Desired Throughput: 200 MB/s
      - Producers: 4 producers × 15 MB/s each
      - Consumers: 6 consumers × 20 MB/s each
      - Producers need 3.33 partitions, consumers 1.67 partitions→ Use 4 partitions (max of both).

https://www.confluent.io/blog/how-choose-number-topics-partitions-kafka-cluster/

Contents

Messaging System and Pub/Sub Paradigm

Kafka Architecture

- Producer
- Consumer
- Topic
  - Messages
- Kafka Cluster
  - Partitions
  - **Replications**

Confluent

- Create a cluster, topic, and produce/subscribe messages.

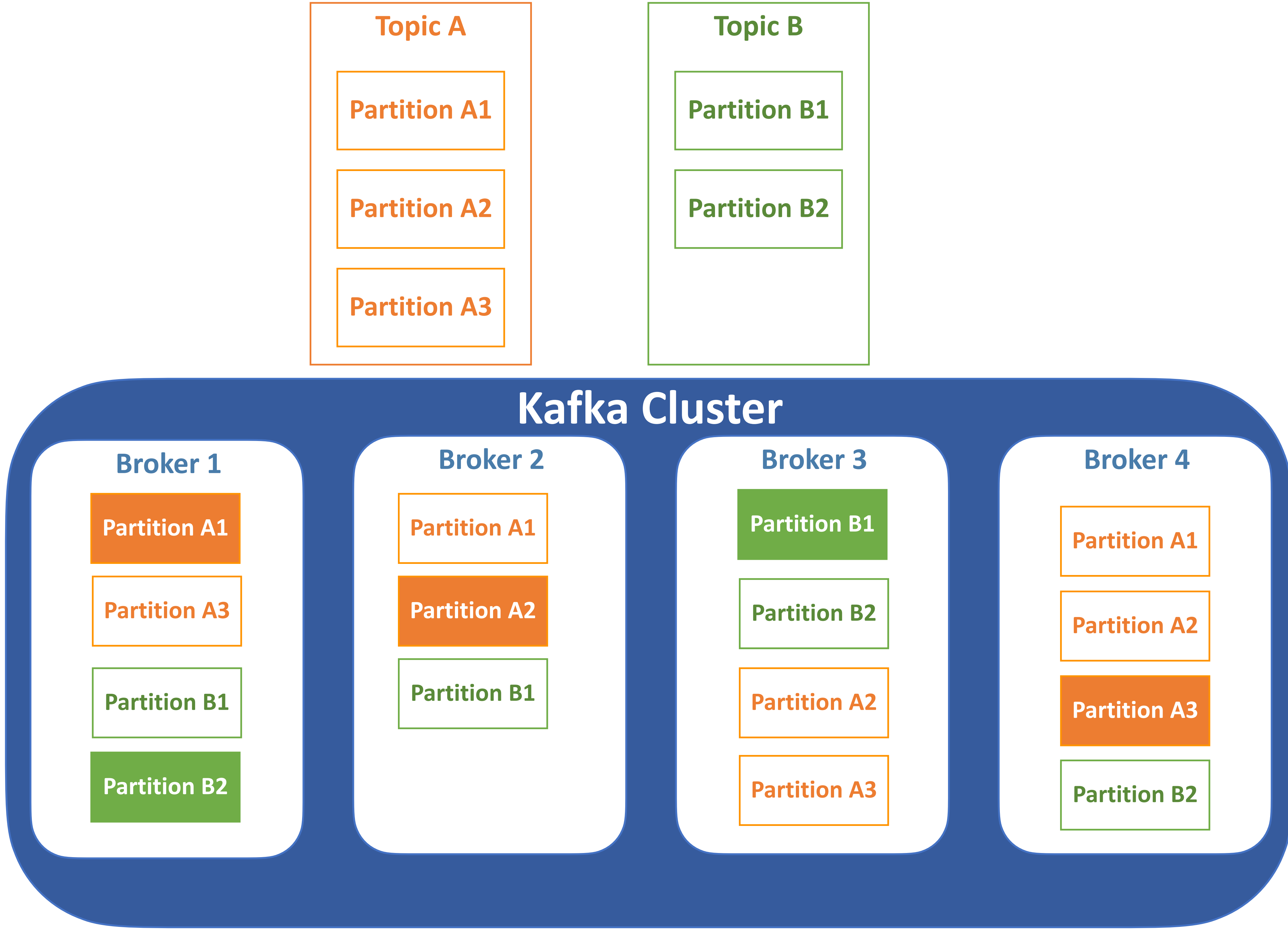
Kafka – Data Replication

Goal: Prevent data loss and ensure reliability in case of broker failure.

Replication Factor: Defines how many copies of each partition exist.

- Can’t exceed number of brokers.

Trade-off: More replicas provides greater fault tolerance, but higher resource usage (I/O, bandwidth, storage).

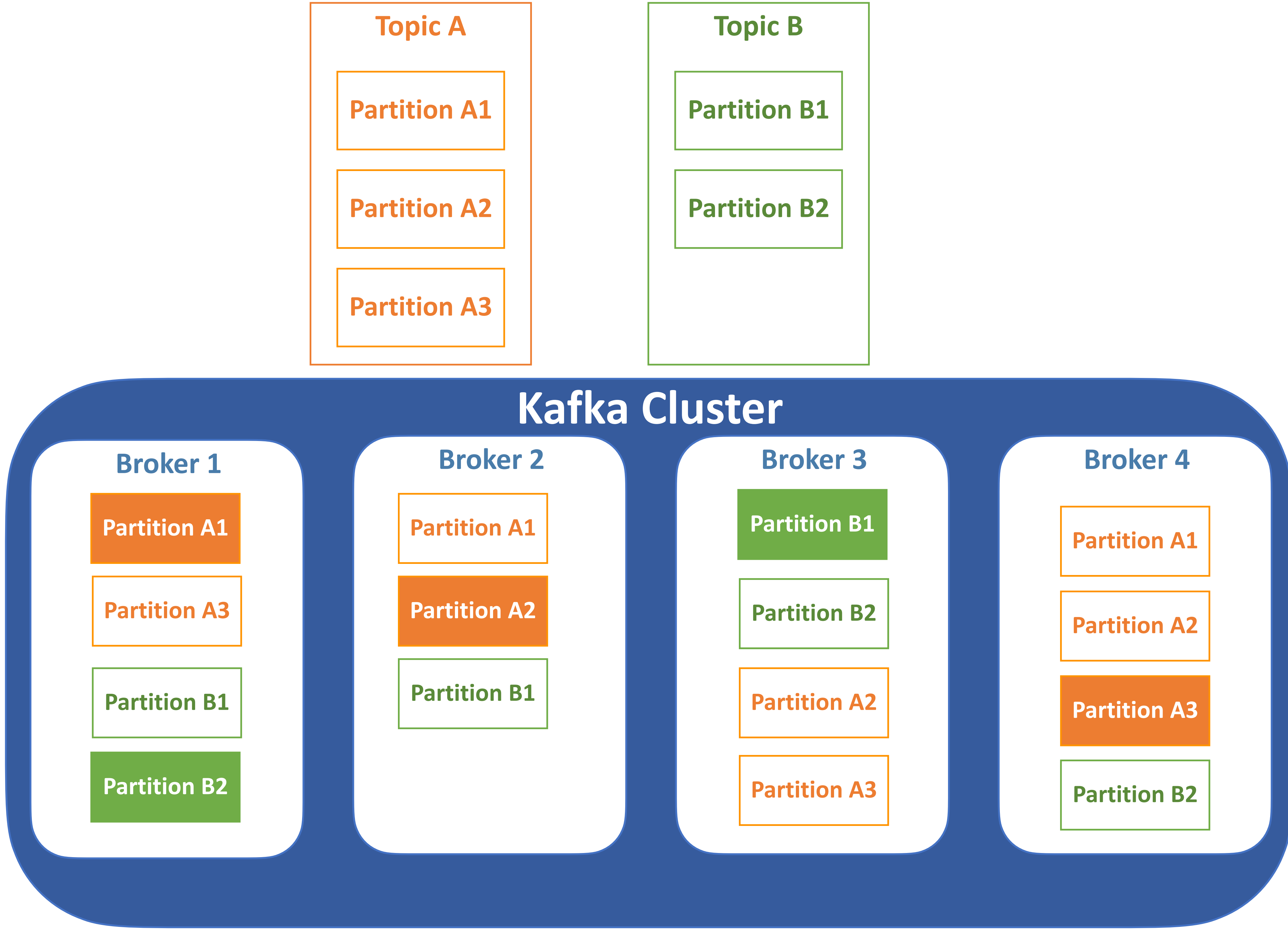


Kafka – Data Replication

Leader and Follower

- Leader
  - Handles all read/write requests.
  - Ensures data integrity and durability.
- Followers
  - Continuously replicate data from the leader.
  - Stay in sync to take over if the leader fails.
- **Automatic Failover**
  - If the leader goes down, a follower is promoted to leader → minimal downtime.

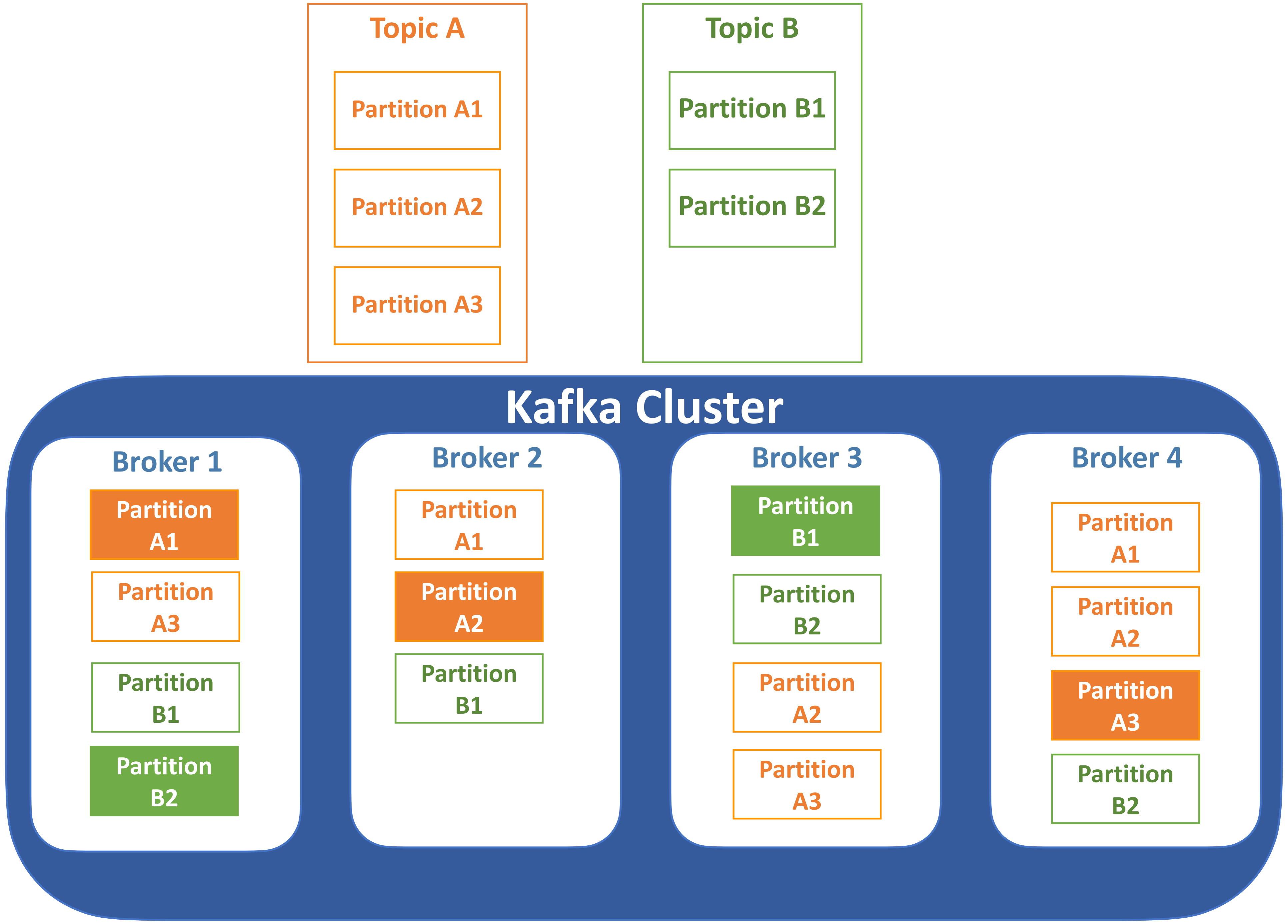
* Balancing replication and resource cost is key for production environments.



Kafka Architecture

In this example,

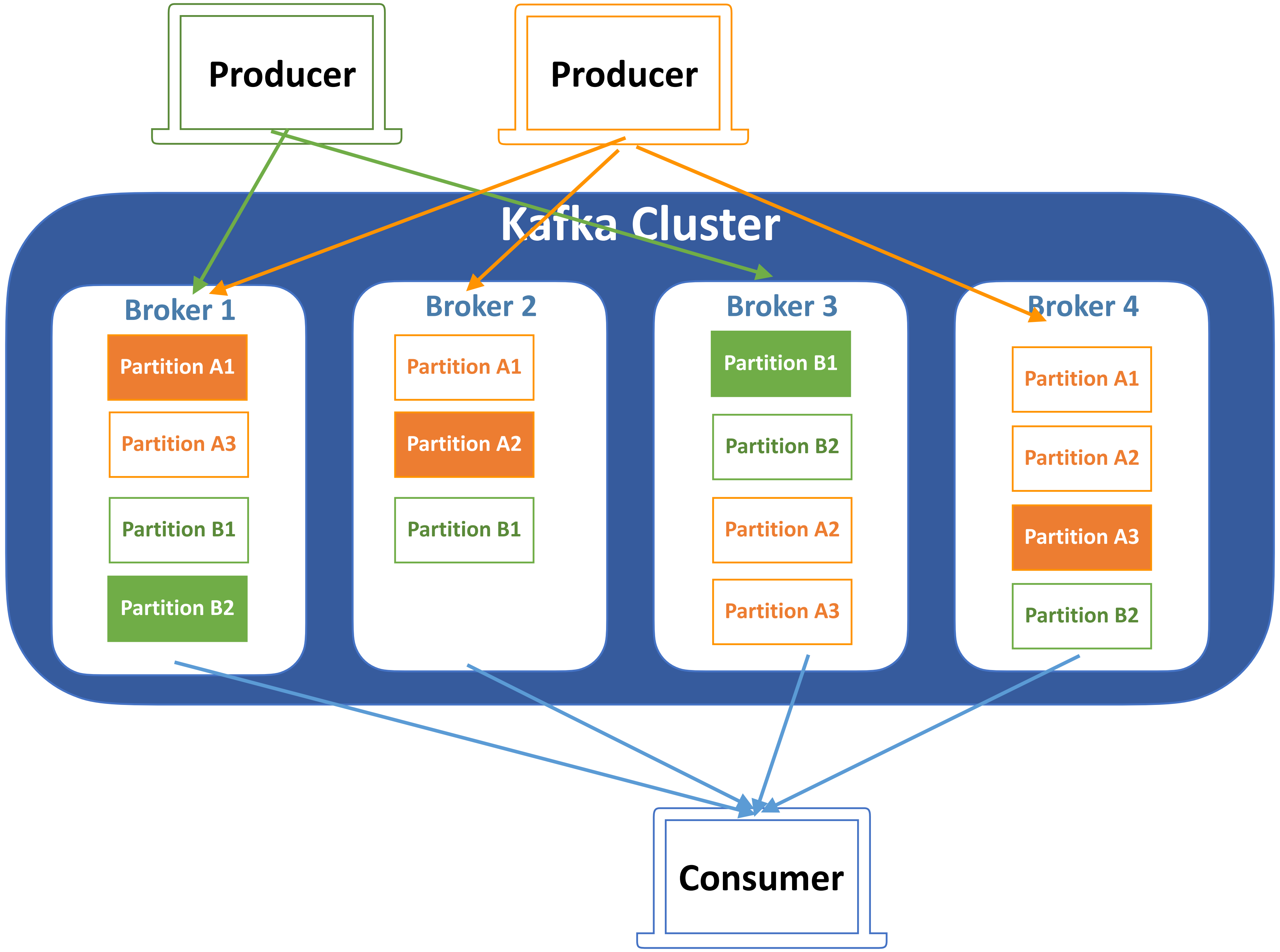
- There are 4 brokers in the Kafka cluster.
- Each partition has 3 replicas for quorum.
  - For each partition, it has a partition leader for handling all read and write requests for a given partition, acting as the primary point of contact for producers.
  - The followers replicate the leader. If the leader fails, one of the followers will automatically become the new leader.



Kafka Architecture

In this example,

- Producers send messages to
  - Broker 1 for Partition A1, Broker 2 for Partition A2, and Broker 4 for Partition A3.
  - Broker 1 for Partition B2, and Broker 3 for Partition B1.



Note:
While the leader typically handles all reads, Kafka introduced the ability for consumers to read from follower replicas in version 2.4.

Contents

Messaging System and Pub/Sub Paradigm

Kafka Architecture

- Producer
- Consumer
- Topic
  - Messages
- Kafka Cluster
  - Partitions
  - Replications

Confluent

- Create a cluster, topic, and produce/subscribe messages.

Confluent

Data streaming platform built around Apache Kafka, created by the original Kafka developers at LinkedIn.

- **Key Components**
  - **Confluent Platform**: A self-managed distribution of Kafka for private deployment with additional enterprise features including schema registry, ksqlDB (streaming SQL engine), Kafka Connect connectors, REST Proxy, etc.
  - **Confluent Cloud**: A fully managed Kafka service, hosted on AWS, GCP, or Azure.

Confluent

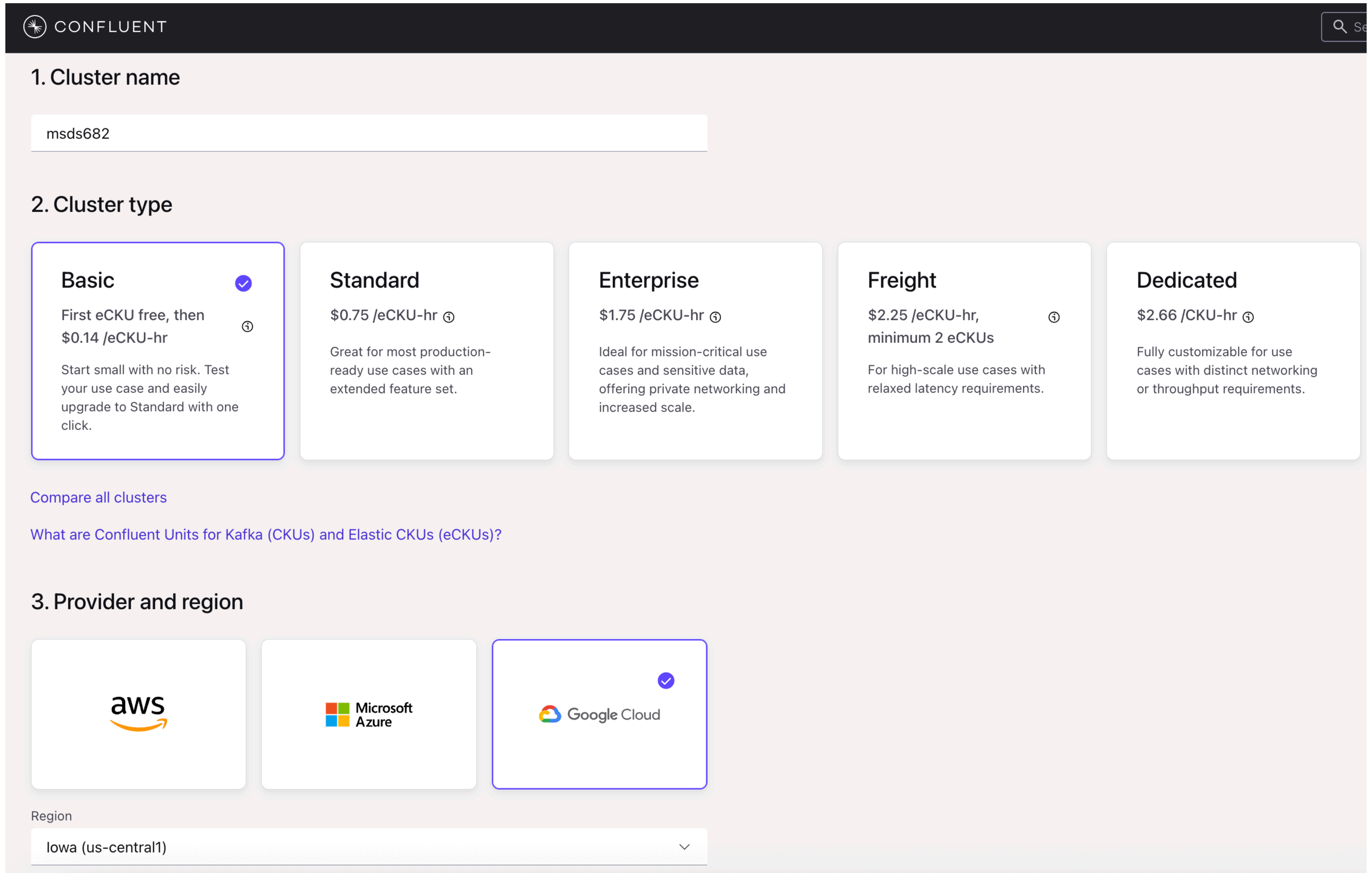
Data streaming platform built around Apache Kafka, created by the original Kafka developers at LinkedIn.

- **Benefits**
  - **Simplified Kafka setup and management** : No need to manage broker nodes and offers a web-based control center for monitoring cluster health, lag, and throughput.
  - **Improved reliability:** Automatic scaling
  - **Built-in connectors**: 120+ pre-built Kafka Connect connectors for databases, cloud services, and data warehouses and easy to integrate.
  - **Schema management**: Schema registry ensures consistent data formats across producers and consumers.

Confluent

Account Creation

- Make sure to use the $400 free credits (first month) before using your GCP credits.
    - Go to https://www.confluent.io/confluent-cloud/tryfree/
    - Choose Basic cluster, and GCP (us-west1)



CONFLUENT

1. Cluster name

msds682

2. Cluster type

Basic ✓
First eCKU free, then $0.14 /eCKU-hr
Start small with no risk. Test your use case and easily upgrade to Standard with one click.

Standard
$0.75 /eCKU-hr
Great for most production-ready use cases with an extended feature set.

Enterprise
$1.75 /eCKU-hr
Ideal for mission-critical use cases and sensitive data, offering private networking and increased scale.

Freight
$2.25 /eCKU-hr, minimum 2 eCKUs
For high-scale use cases with relaxed latency requirements.

Dedicated
$2.66 /CKU-hr
Fully customizable for use cases with distinct networking or throughput requirements.

Compare all clusters

What are Confluent Units for Kafka (CKUs) and Elastic CKUs (eCKUs)?

3. Provider and region

aws

Microsoft Azure

Google Cloud ✓

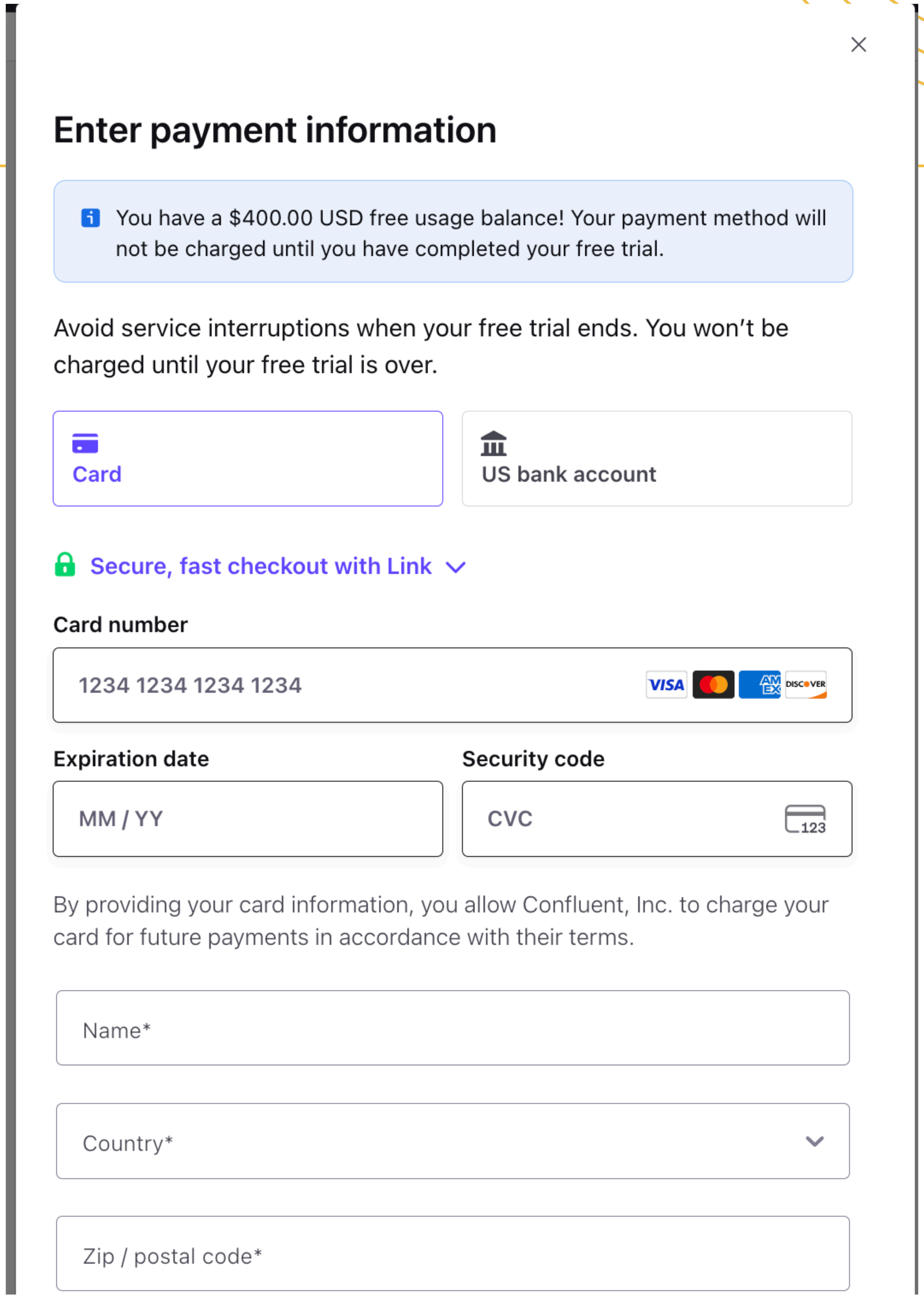
Region

Iowa (us-central1)

Confluent

Account Creation

- Make sure to use the $400 free credits (first month) before using your GCP credits.
  - Go to https://www.confluent.io/confluent-cloud/tryfree/
  - Choose Basic cluster, and GCP (us-west1)



Enter payment information

You have a $400.00 USD free usage balance! Your payment method will not be charged until you have completed your free trial.

Avoid service interruptions when your free trial ends. You won't be charged until your free trial is over.

Card number: 1234 1234 1234 1234

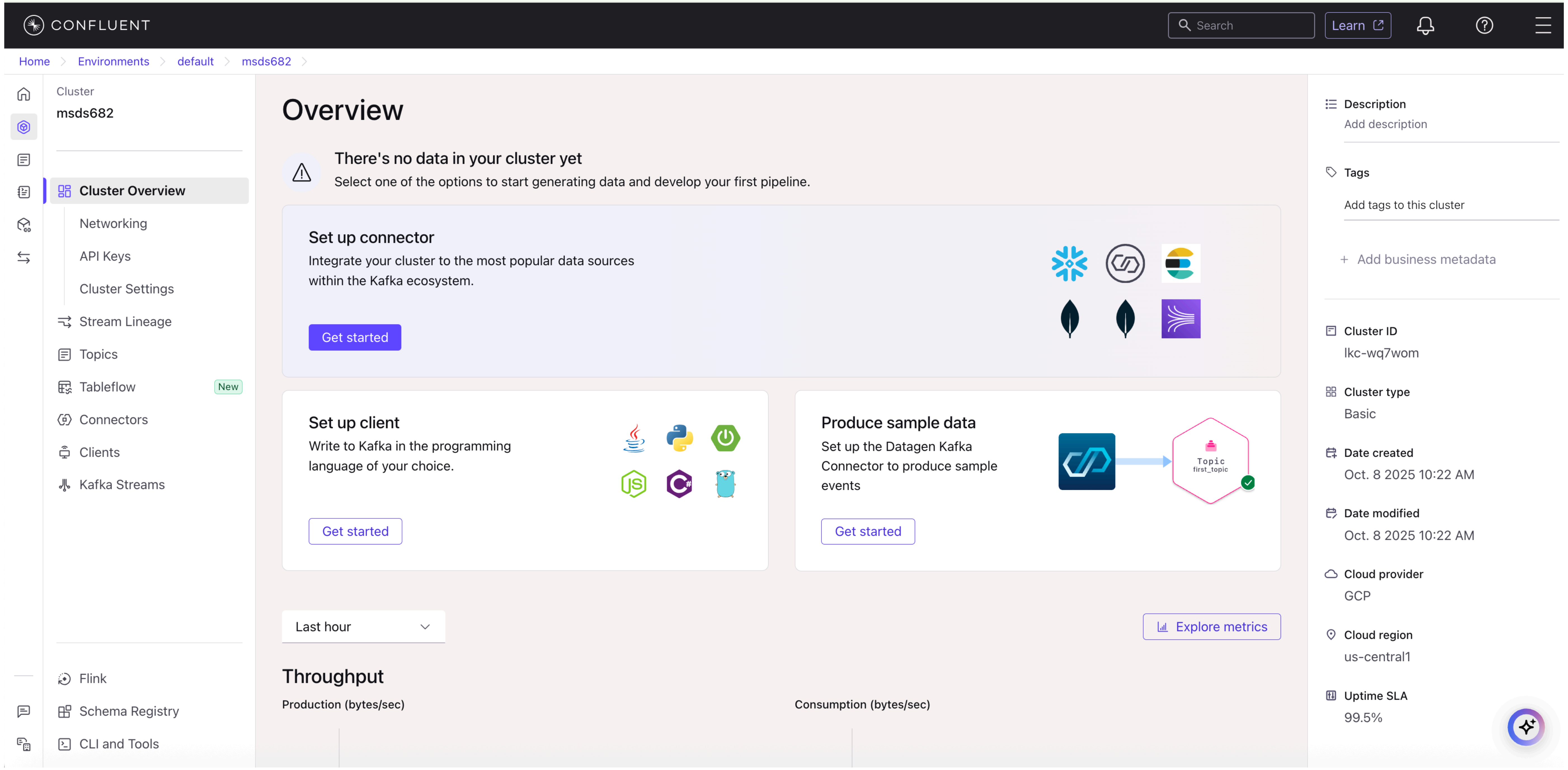
Expiration date: MM / YY

Security code: CVC

By providing your card information, you allow Confluent, Inc. to charge your card for future payments in accordance with their terms.

https://docs.confluent.io/cloud/current/get-started/free-trial.html

Confluent



CONFLUENT

Search | Learn

Home > Environments > default > msds682 >

Cluster

msds682

- Cluster Overview
- Networking
- API Keys
- Cluster Settings
- Stream Lineage
- Topics
- Tableflow New
- Connectors
- Clients
- Kafka Streams
- Flink
- Schema Registry
- CLI and Tools

Overview

There's no data in your cluster yet

Select one of the options to start generating data and develop your first pipeline.

Set up connector

Integrate your cluster to the most popular data sources within the Kafka ecosystem.

Get started

Set up client

Write to Kafka in the programming language of your choice.

Get started

Produce sample data

Set up the Datagen Kafka Connector to produce sample events

Topic
first_topic

Get started

Last hour

Explore metrics

Throughput

Production (bytes/sec)

Consumption (bytes/sec)

Description

Add description

Tags

Add tags to this cluster

+ Add business metadata

Cluster ID

lkc-wq7wom

Cluster type

Basic

Date created

Oct. 8 2025 10:22 AM

Date modified

Oct. 8 2025 10:22 AM

Cloud provider

GCP

Cloud region

us-central1

Uptime SLA

99.5%

https://docs.confluent.io/cloud/current/get-started/free-trial.html

Contents

Messaging System and Pub/Sub Paradigm

Kafka Architecture

- Producer
- Consumer
- Topic
  - Messages
- Kafka Cluster
  - Partitions
  - Replications

Confluent

- **Create a cluster, topic, and produce/subscribe messages.**

Confluent CLI

The Confluent command-line interface (CLI) enables developers to manage both Confluent Cloud and Confluent Platform.

- Installation

```
$ brew install confluentinc/tap/cli
```

- Login

```
$ confluent login
```

https://docs.confluent.io/kafka/operations-tools/kafka-tools.html

Confluent Basics

Create a Confluent Cloud environment

- Confluent Cloud environment contains Kafka clusters and deployed components, such as Connect, ksqlDB, and Schema Registry

```
$ confluent environment create environment_name -o json
```

  - This will return ID, and name of the environment

Activate the environment to use to create a cluster

```
$ confluent environment use environment_id
```

```
{
  "is_current": false,
  "id": "env-102dw3",
  "name": "msdse682",
  "stream_governance_package": "ESSENTIALS"
}
```

https://docs.confluent.io/cloud/current/security/access-control/hierarchy/cloud-environments.html

Confluent Basics

Create a Kafka Cluster on GCP

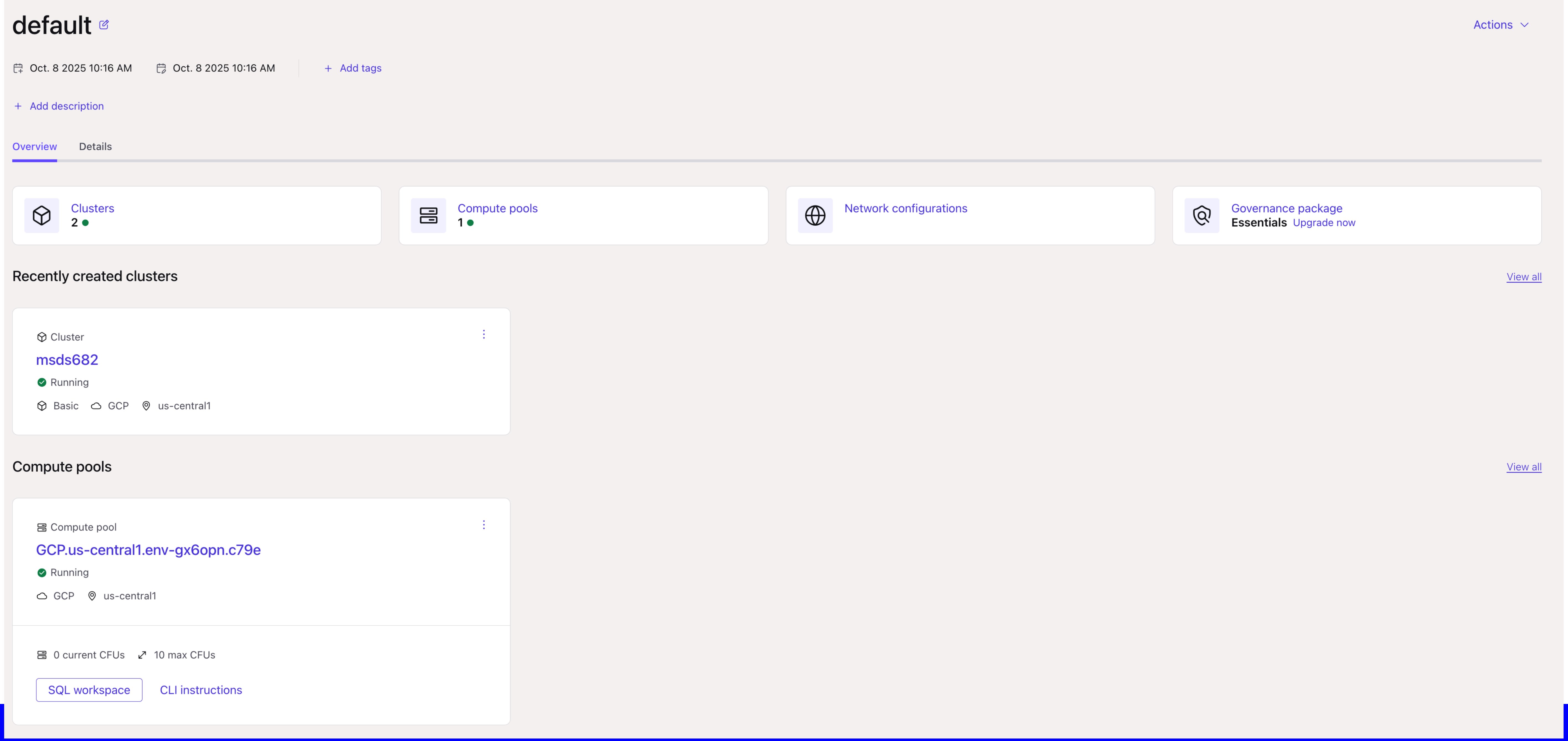
```
$  confluent kafka cluster create cluster_name
--cloud gcp --region us-west1
```

- This will return ID, name, and endpoints of the Kafka cluster

```
+----------------------+--------------------------------------------------------+
| Current              | false                                                  |
| ID                   | lkc-yp2gwj                                             |
| Name                 | day2                                                   |
| Type                 | BASIC                                                  |
| Ingress Limit (MB/s) | 250                                                    |
| Egress Limit (MB/s)  | 750                                                    |
| Storage              | 5 TB                                                   |
| Cloud                | gcp                                                    |
| Region               | us-west1                                               |
| Availability         | single-zone                                            |
| Status               | PROVISIONING                                           |
| Endpoint             | SASL_SSL://pkc-lgk0v.us-west1.gcp.confluent.cloud:9092 |
| REST Endpoint        | https://pkc-lgk0v.us-west1.gcp.confluent.cloud:443     |
+----------------------+--------------------------------------------------------+
```

Confluent Basics

Create a Kafka Cluster on GCP



default

Oct. 8 2025 10:16 AM Oct. 8 2025 10:16 AM

Clusters 2 ●

Compute pools 1 ●

Network configurations

Governance package **Essentials** Upgrade now

Recently created clusters

Cluster
msds682
Running
Basic GCP us-central1

Compute pools

Compute pool
GCP.us-central1.env-gx6opn.c79e
Running
GCP us-central1

0 current CFUs 10 max CFUs

Confluent Basics

Create an API Key/Secret for authorization to produce topics

```
$  confluent api-key create --description "MSDS682 credentials" --resource cluster_id -o json
```

- This will return a json string - Make sure to save it

Specify the API Key for the cluster to use

```
$  confluent api-key use API_KEY --resource cluster_id
```

Confluent Basics

Create a topic and produce events

- Choose a cluster to use

```
$  confluent kafka cluster use cluster_id
```

- Create a Kafka topic within the cluster

```
$ confluent kafka topic create topic_name
```

- Start producing messages to the topic

```
$ confluent kafka topic produce topic_name
```

  - The CLI waits for data, and you can type on the terminal.

Confluent Basics

Consume messages from topic.

- **In a separate terminal**, read the message
  - Optional parameter: `–b` : read from the beginning

    ```
    $ confluent kafka topic consume topic_name [–b]
    ```

  - While keep producing messages on the producer’s terminal, see what is happening on the consumer’s terminal.

Confluent Basics

Clean up

- Delete topics

```
$ confluent kafka topic delete topic_name
```

- Delete clusters

```
$ confluent kafka cluster delete cluster_id
```

- Delete environment

```
$ confluent environment delete environment_id
```

Contents

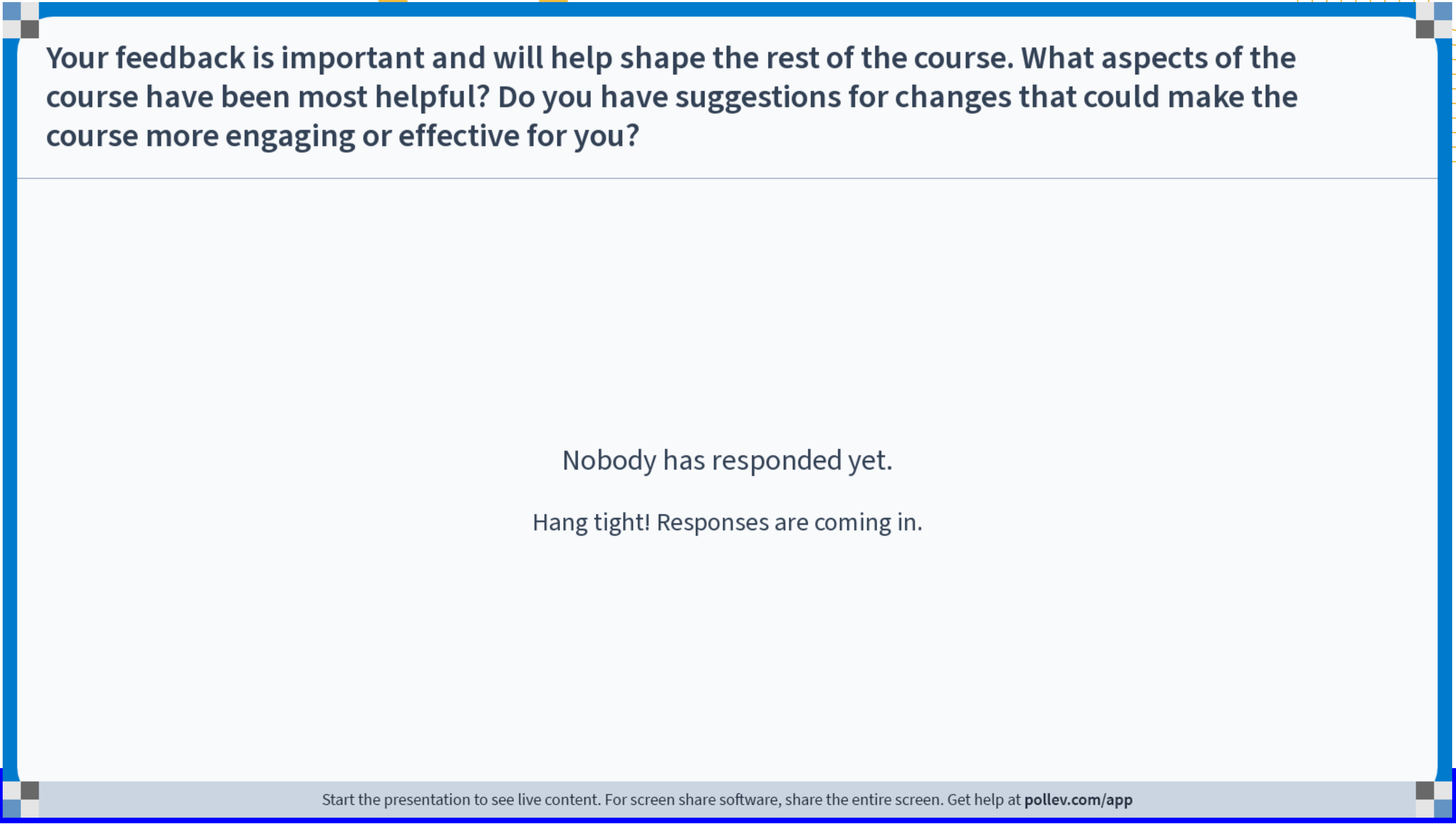
Messaging System and Pub/Sub Paradigm

Kafka Architecture

- Producer
- Consumer
- Topic
  - Messages
- Kafka Cluster
  - Partitions
  - Replications

Confluent

- Create a cluster, topic, and produce/subscribe messages.



Your feedback is important and will help shape the rest of the course. What aspects of the course have been most helpful? Do you have suggestions for changes that could make the course more engaging or effective for you?

Nobody has responded yet.

Hang tight! Responses are coming in.

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Reference

Apache Kafka, https://kafka.apache.org/documentation/

Kafka Tools, Confluent, https://docs.confluent.io/kafka/operations-tools/kafka-tools.html

Confluent Documentation, https://docs.confluent.io/

Some of the lecture materials are from Jeremy Gu, a former instructor of MSDS682.

- https://pandaisfast.github.io/msds682-fall2023-data-streaming