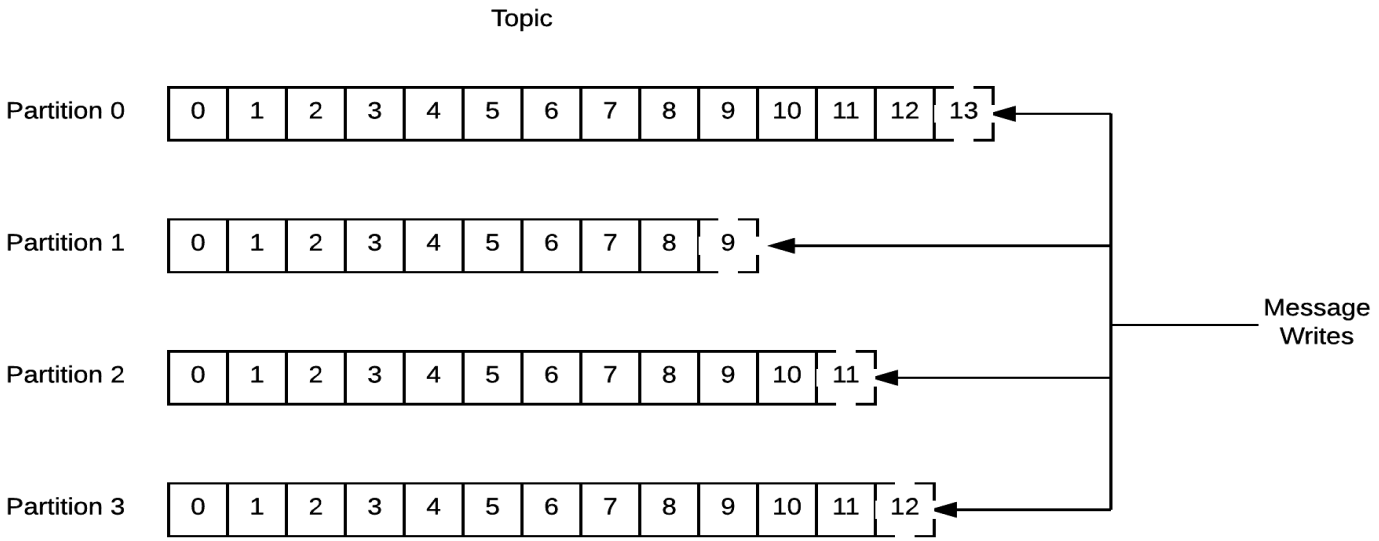
Scaling Kafka

Topics and Partitions

- Messages in Kafka are categorized into *topics*
- Think of a topic as a database table or folder in a filesystem
- Topics are broken down into a number of *partitions*
- A topic generally has multiple partitions

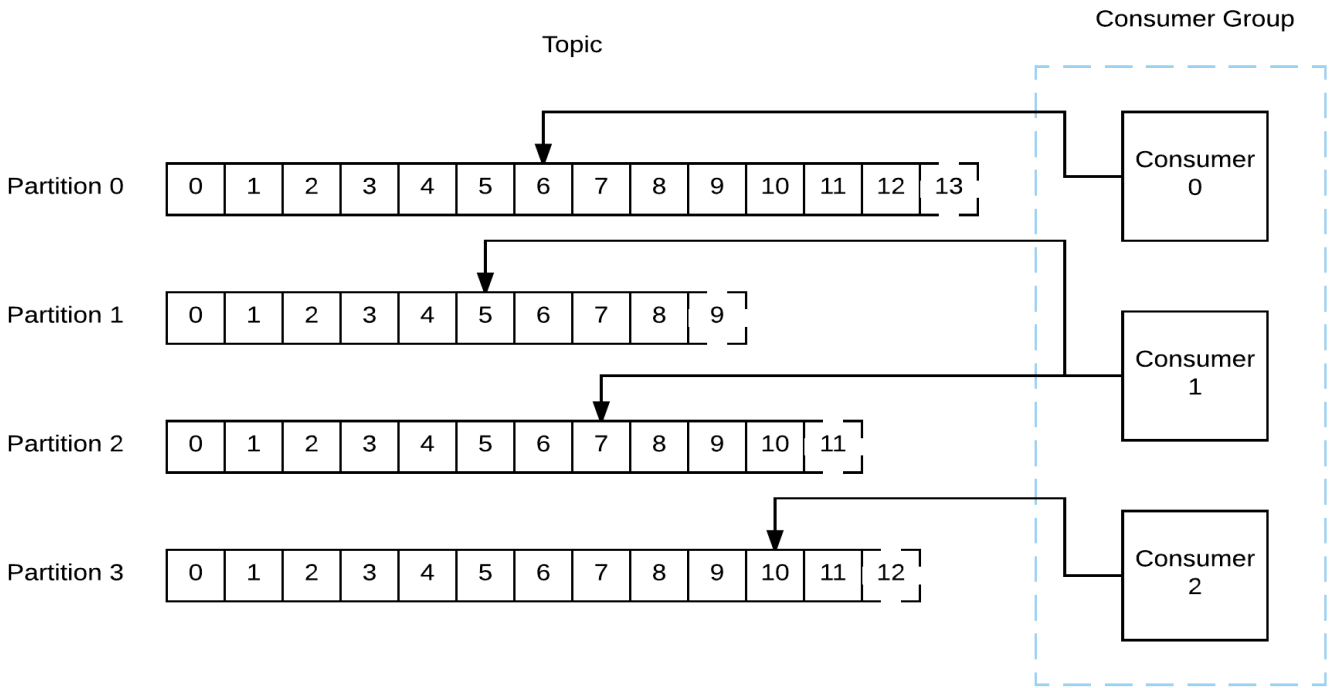
Topics and Partitions



Consumer Groups

- Consumers work as part of a *consumer group*
- One or more consumers that work together to consume a topic
- Group assures that each partition is only consumed by one member
- Mapping of a consumer to a partition is called *ownership* of the partition by the consumer

Consumer Group Illustrated



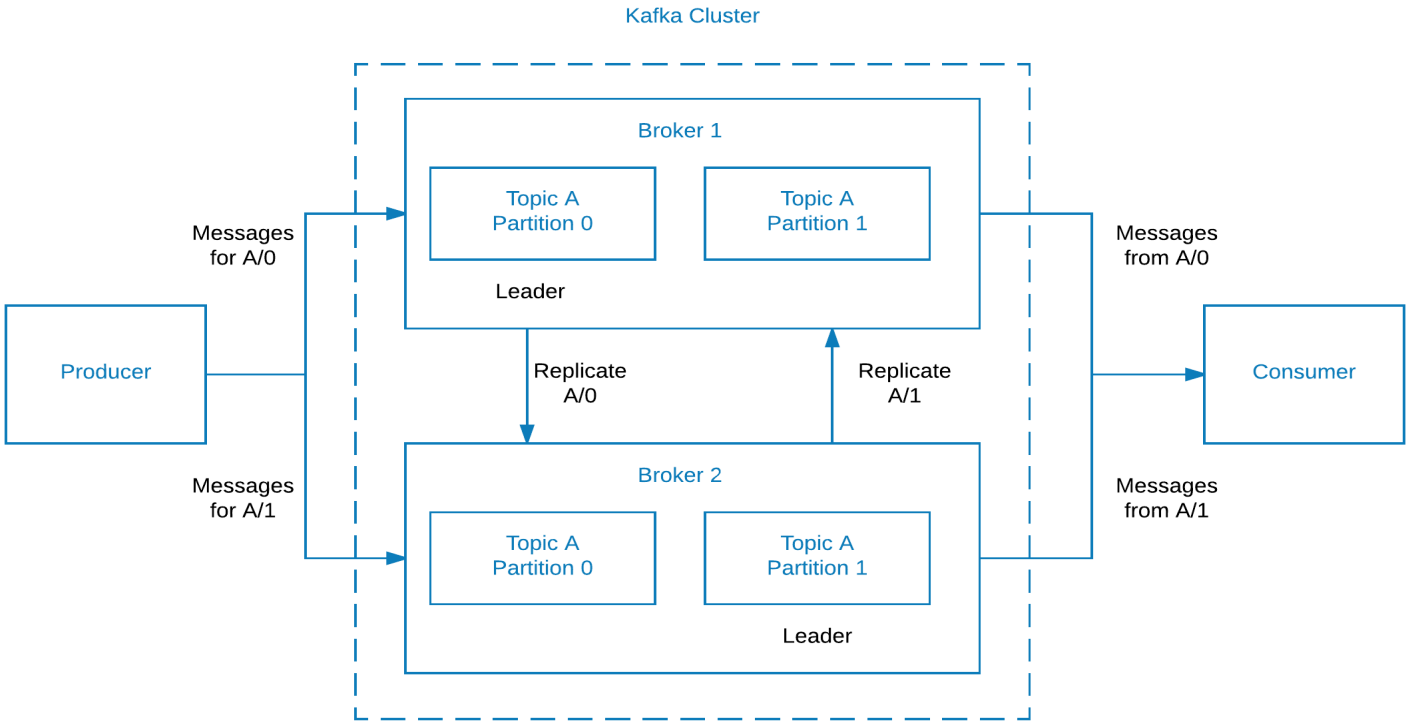
Brokers

- A single Kafka server is called a *broker*
- The broker receives messages from producers, assigns offsets to them, and commits the messages to storage on disk
- Responds to consumer fetch requests with the messages in the requested partition
- **A single broker can handle thousands of partitions and millions of messages per second**

Clusters

- Kafka brokers operate as part of a cluster
- Within a cluster of brokers, one broker acts as the ***cluster controller***
- The controller handles administrative operations such as assigning partitions to brokers and monitoring for broker failures
- A partition is owned by a single broker in the cluster known as the ***leader*** for the partition
- Another broker can take over leadership if there is a broker failure
- All consumers and producers on that partition must connect to the leader

Replication of Partitions in a Cluster



Retention

- Kafka brokers are configured with a default retention setting for topics either retaining messages for some period of time or until the topic reaches a certain size in bytes
- When limits are reached, messages are expired and deleted
- Topics can be configured with their own retention settings
  - Application metrics may only be useful for a few hours
  - Tracking topics may be useful for several days
- Topics can be *log compacted* – Kafka only retains the last message produced with a specific key
  - For example, changelog data where only the last update is useful

Compression

- In some of the largest Kafka deployments, hundreds of billions of messages per day are processed amounting to **moving hundreds of terabytes of data**

- Enabling compression allows you to reduce network utilization which can be a bottleneck when sending messages to Kafka

- Compression is an optional configuration setting in the broker and producer

Supported Compression Algorithms

- `snappy`
  - Created by Google
  - Provides good compression ratio with low CPU overhead
  - Recommended when both performance and bandwidth are a concern
- `gzip`
  - Uses more CPU and time but results in better compression ratios
  - Recommended where network bandwidth is more restricted
- `lz4`
  - Newest algorithm to Kafka
  - Faster than `gzip` and smaller than `snappy`

Compression Overview

- Multiple messages are bundled and compressed
- The compressed messages are then appended to Kafka's log file
- Compression works on a batch of messages rather than individual to take advantage of the fact that compressors work more efficiently with bigger data

Multiple Clusters

- As your Kafka deployment grows, it can be advantageous to have multiple clusters:
  - Segregation of types of data
  - Isolation for security requirements
  - Multiple datacenters (disaster recovery)

- With multiple datacenters, messages must be copied between them

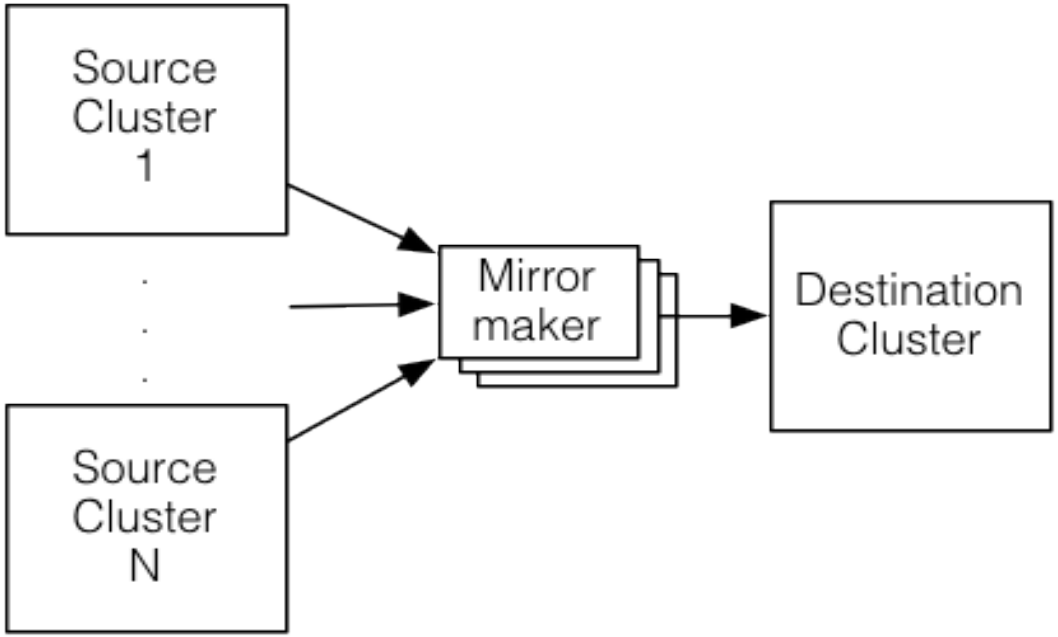
- Replication within a Kafka cluster only works within a single cluster, not between multiple clusters

Cluster Mirroring

- Kafka includes a tool called *Mirror Maker* to mirror a source Kafka cluster into a target (mirror) cluster

- Uses a Kafka consumer to consume messages from the source cluster and re-publishes the messages to the local (target) cluster using an embedded Kafka producer

Mirror Maker Illustrated



Scaling with MirrorMaker

- Designate clusters into two categories
- **Mission critical,** production
  - E.g. supporting microservices
- **Non-mission critical**
  - Consumers which are not mission critical read from another cluster
  - E.g. run analytics, feed real-time analytics and dashboards
- Mission critical cluster is not affected by readers

Multiple MirrorMakers

- Production cluster
- Analytics cluster
- Backup cluster

- Mirrormakers:
  - Production → MirrorMaker1 → Analytics
  - Production → MirrorMaker2 → Backup

Summary

- Topics and Partitions
- Producers and Consumers
- Consumer Groups
- Brokers
- Clusters
- Retention
- Compression
- Mirror Maker