Kafka Overview

By: Hayden Wade

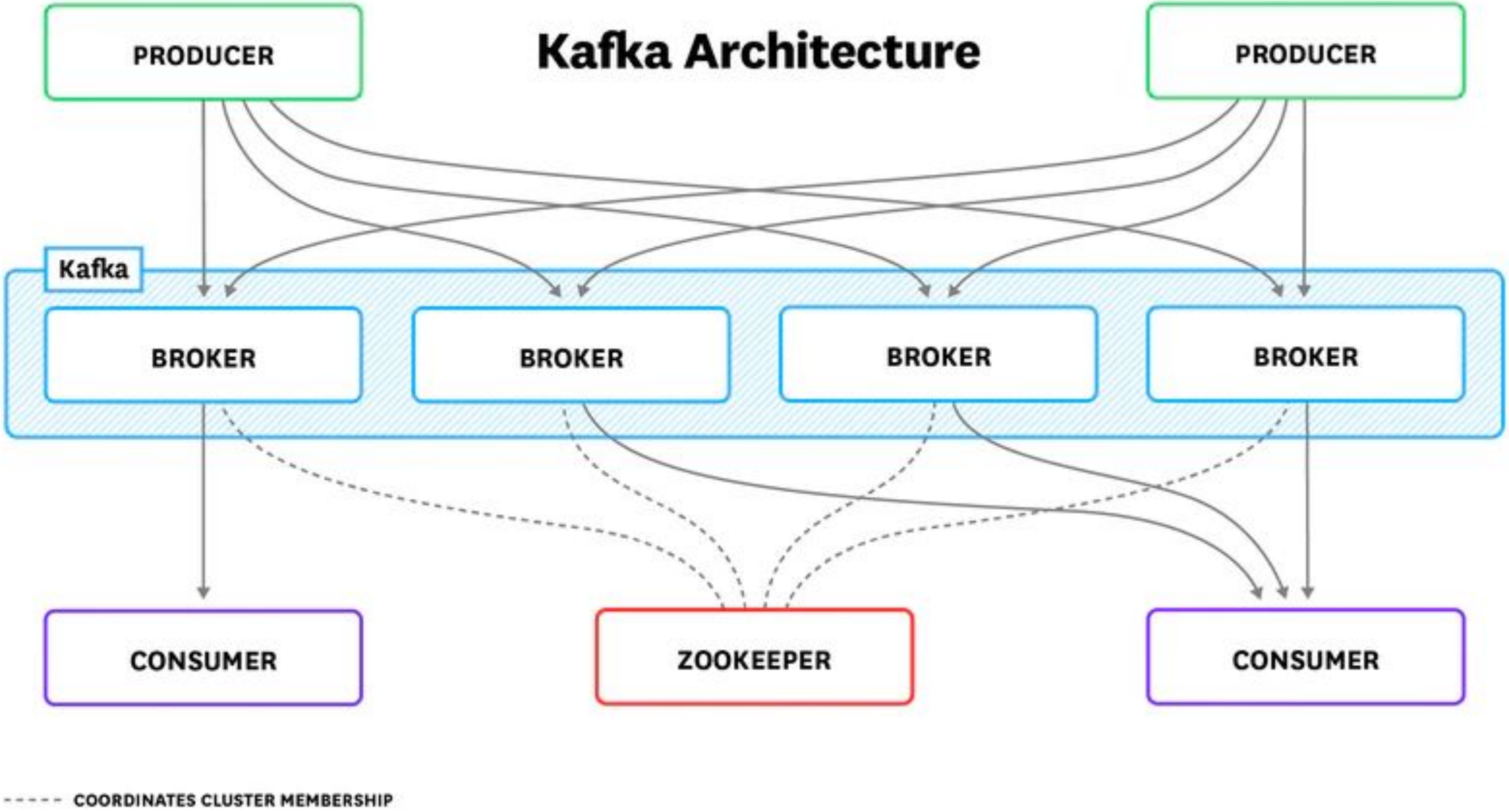
What we will cover...

- What is Kafka?
- Use Cases and Comparison
- Components of Kafka
- Demo

What is Kafka?

- Distributed streaming platform
- Publish and subscribe to streams of records, similar to a message queue or enterprise messaging system.
- Store streams of records in a fault-tolerant durable way.
- Process streams of records as they occur.

Kafka Architecture



Use Cases

- Building real-time streaming data pipelines that reliably get data between systems or applications
- Building real-time streaming applications that transform or react to the streams of data

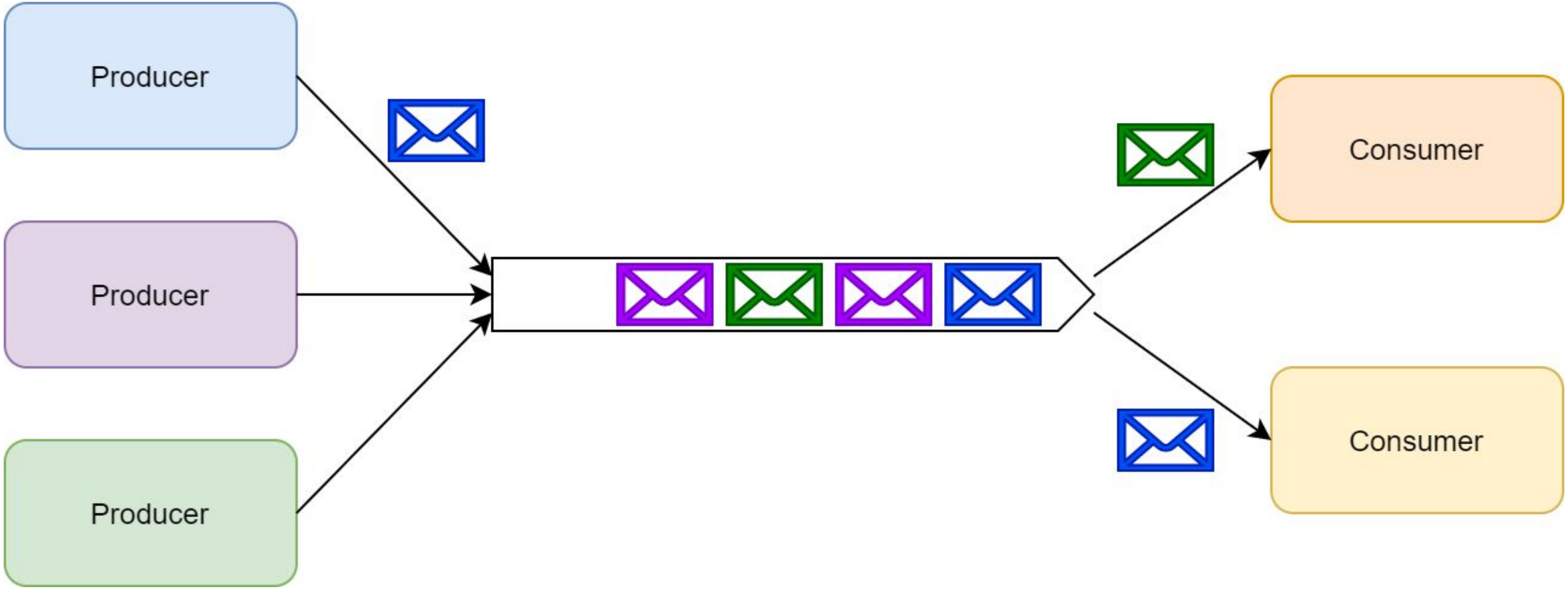
How’s Kafka compare to other messaging systems?

Two other models for messaging:

1. Queuing
2. Publish-Subscribe

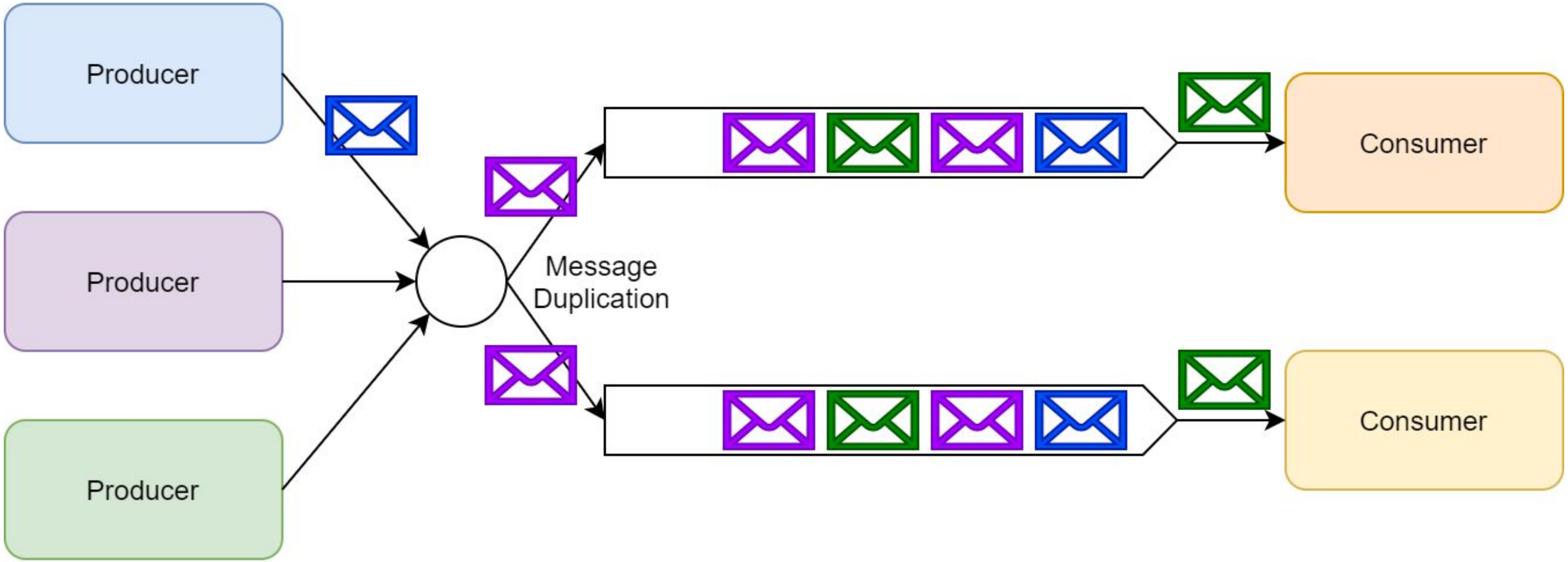
Queuing Messaging Model

- A pool of consumers may read from a server and each record goes to one of them
  - Pros:
    - Allows you to divide up the processing of data over multiple consumer instances, which lets you scale your processing
  - Cons:
    - Queues aren't multi-subscriber—once one process reads the data it's gone.


Producer
Producer
Producer
Consumer
Consumer

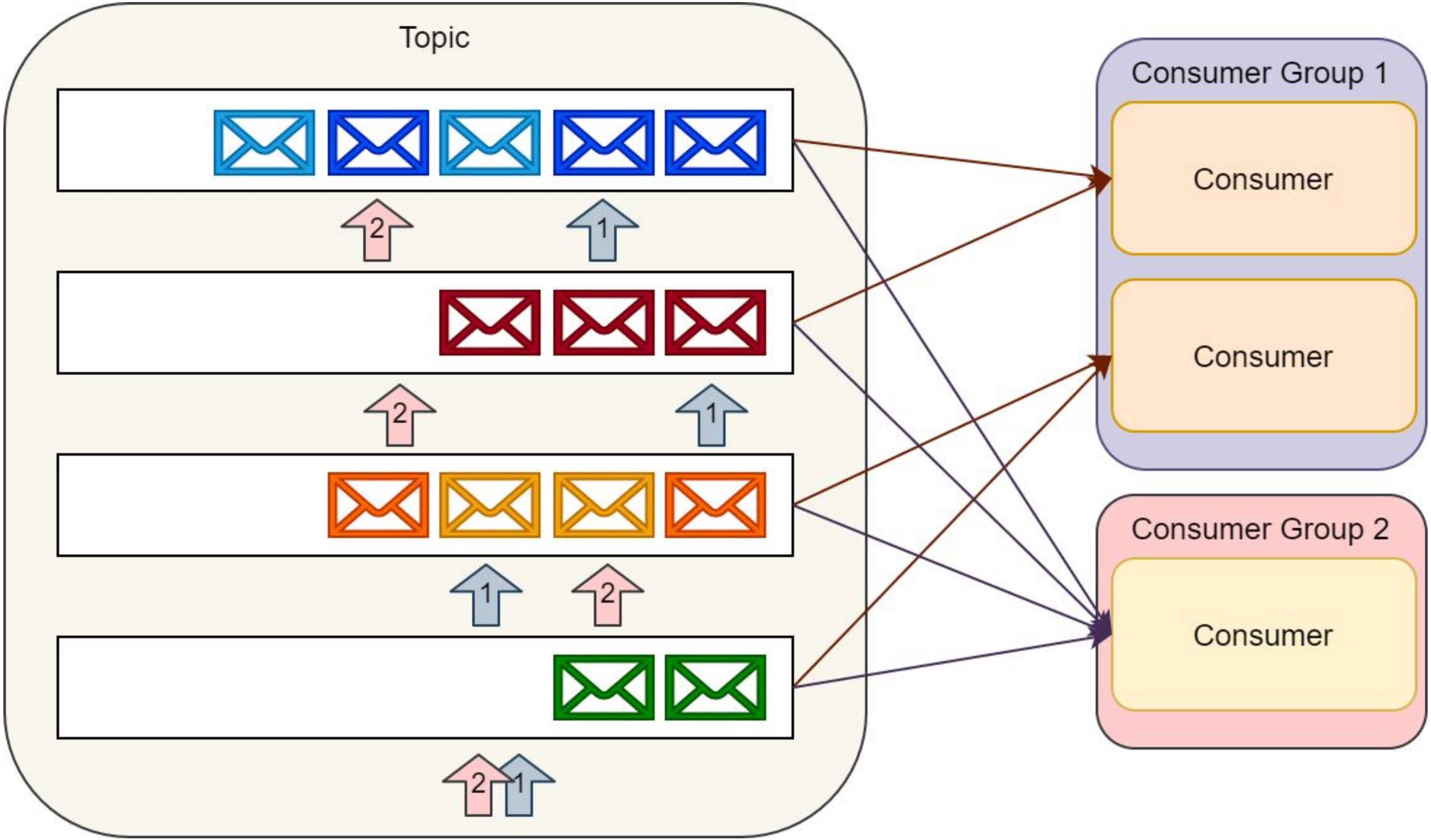
Publish-Subscribe Messaging Model

- The record (message) is broadcast to all consumers
- Pros:
  - Allows you broadcast data to multiple processes
- Cons:
  - Has no way of scaling processing since every message goes to every subscriber


Producer
Producer
Producer
Message
Duplication
Consumer
Consumer

Kafka’s Consumer Group Model

- Consumer group allows you to divide up processing over a collection of processes (similar to a queue)
- Kafka allows you to broadcast messages to multiple consumer groups (similar to publish-subscribe)
- It can scale processing and is also multi-subscriber—there is no need to choose one or the other (applies to every topic)
- Kafka has stronger ordering guarantees than a traditional messaging system
    - See [Apache Kafka Docs](https://kafka.apache.org/) for the how


Topic
2
1
2
1
1
2
2
1
Consumer Group 1
Consumer
Consumer
Consumer Group 2
Consumer

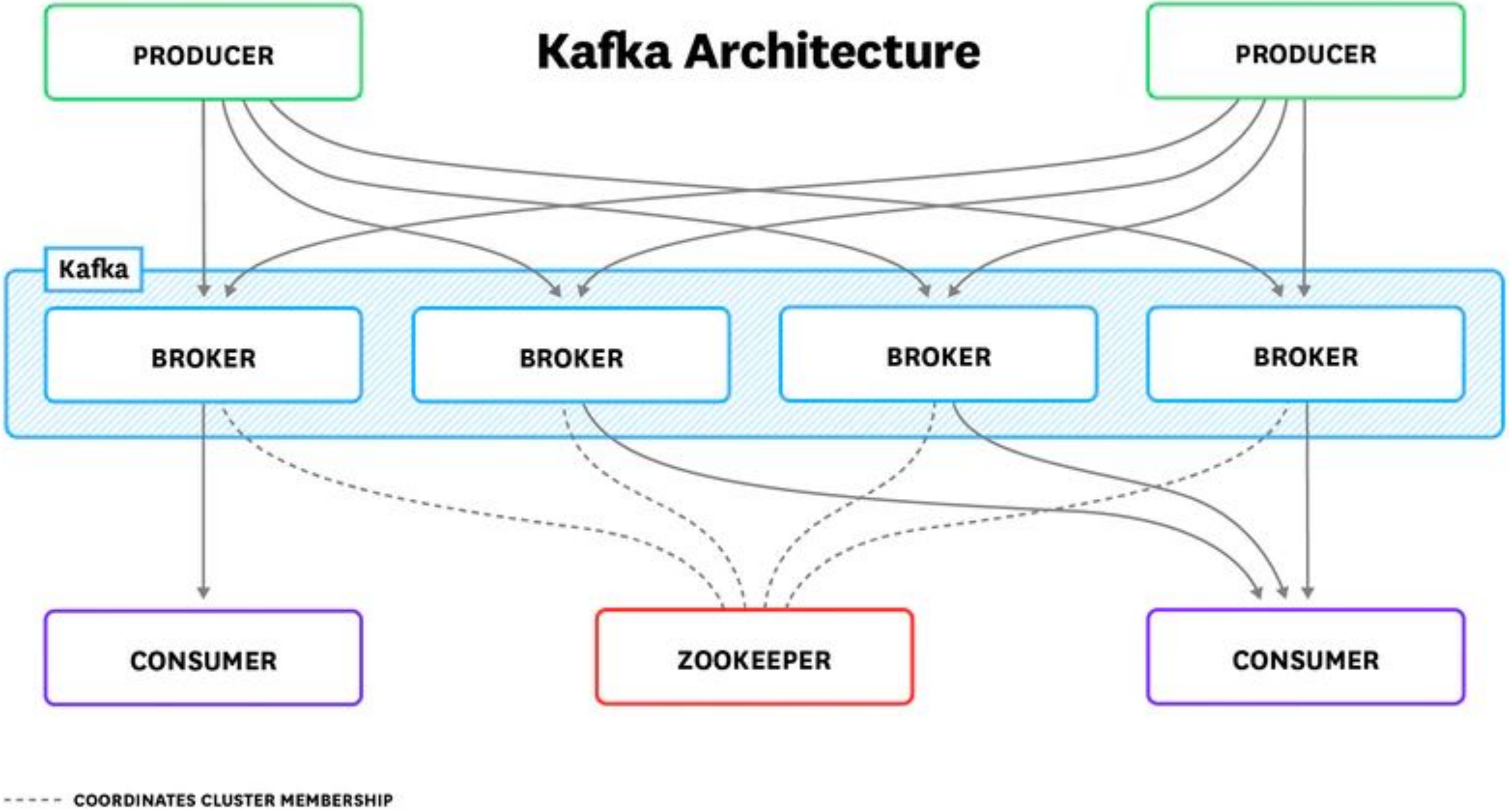
Guarantees

- Messages sent by a producer to a particular topic partition will be appended in the order they are sent. That is, if a record M1 is sent by the same producer as a record M2, and M1 is sent first, then M1 will have a lower offset than M2 and appear earlier in the log.
- A consumer instance sees records in the order they are stored in the log.
- For a topic with replication factor N, we will tolerate up to N-1 server failures without losing any records committed to the log.

Kafka Components

- Connect API
  - Used to migrate data from a data store into a topic or vice versa (data integrations)
- Streams API
  - Used to enrich(join), filter, or aggregate messages across topics
- Admin API
  - Allows managing and inspecting topics, brokers and other Kafka objects
- Producer API
  - Allows applications to publish records to one or more topics
- Consumer API
  - Allows applications to subscribe to one or more topics

Kafka Architecture



PRODUCER

PRODUCER

Kafka

BROKER

BROKER

BROKER

BROKER

CONSUMER

ZOOKEEPER

CONSUMER

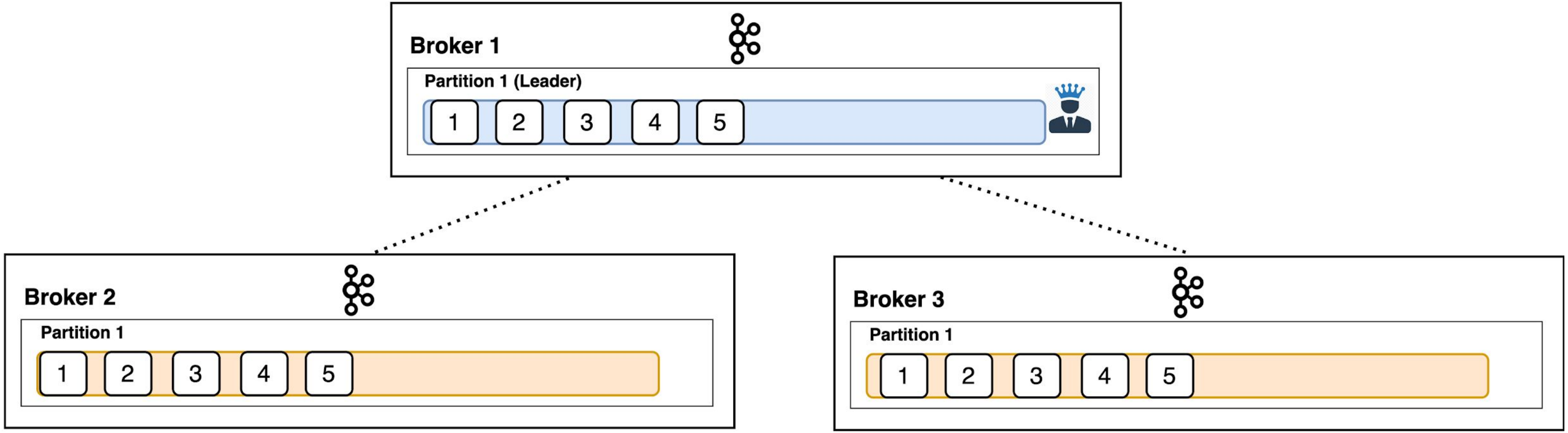
- - - - COORDINATES CLUSTER MEMBERSHIP

Topics

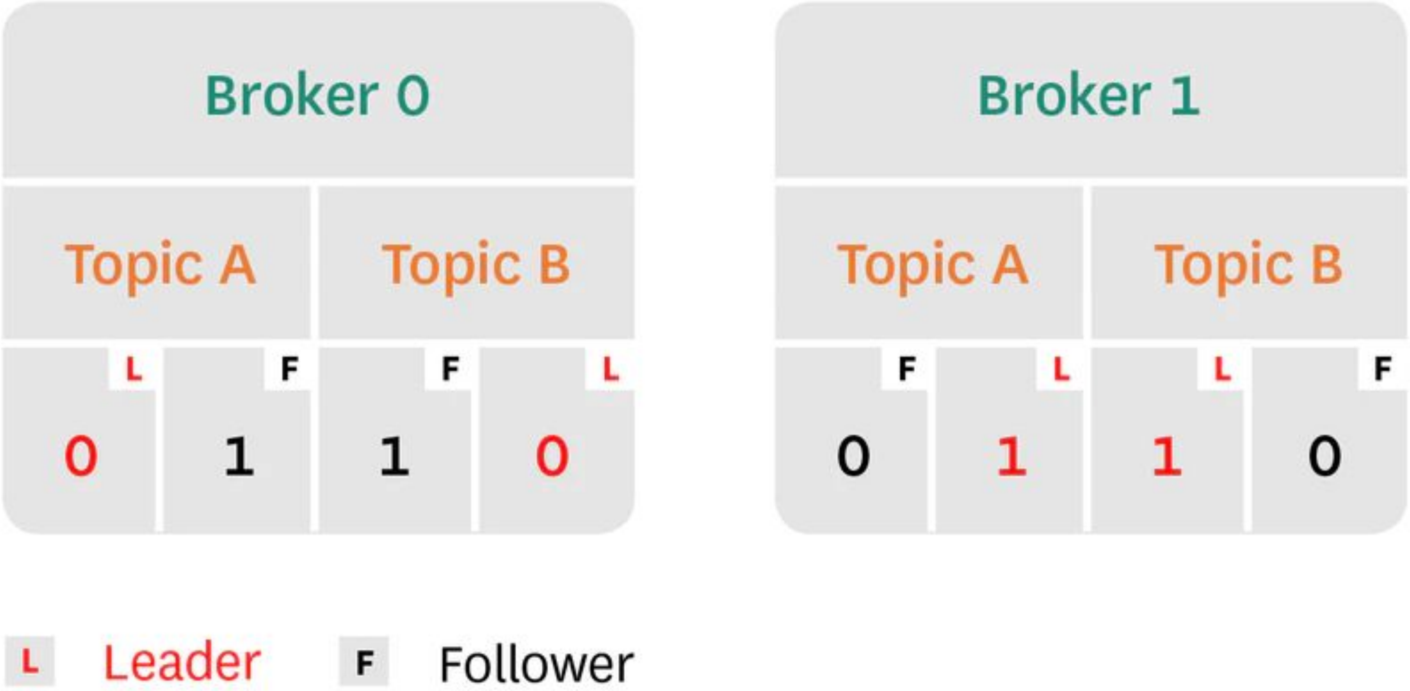
- Defined by their name
- Ordered collection of events stored in a durable way (saved to disk with replication)
- Can be very large or very small
- Topics are broken into partitions and each partition has its own offset - messages in a partition are ordered
- Topics have a replication factor
- Order is only guaranteed within a single partition

Brokers

- Multiple per kafka cluster
- Contains certain topic partitions or replicas
- Only one broker can be a leader for a given partition (receive and send data), other brokers are used to synchronize data



Leader Follower per Topic example:



Broker 0				Broker 1			
Topic A		Topic B		Topic A		Topic B	
L	F	F	L	F	L	L	F
0	1	1	0	0	1	1	0

L Leader F Follower

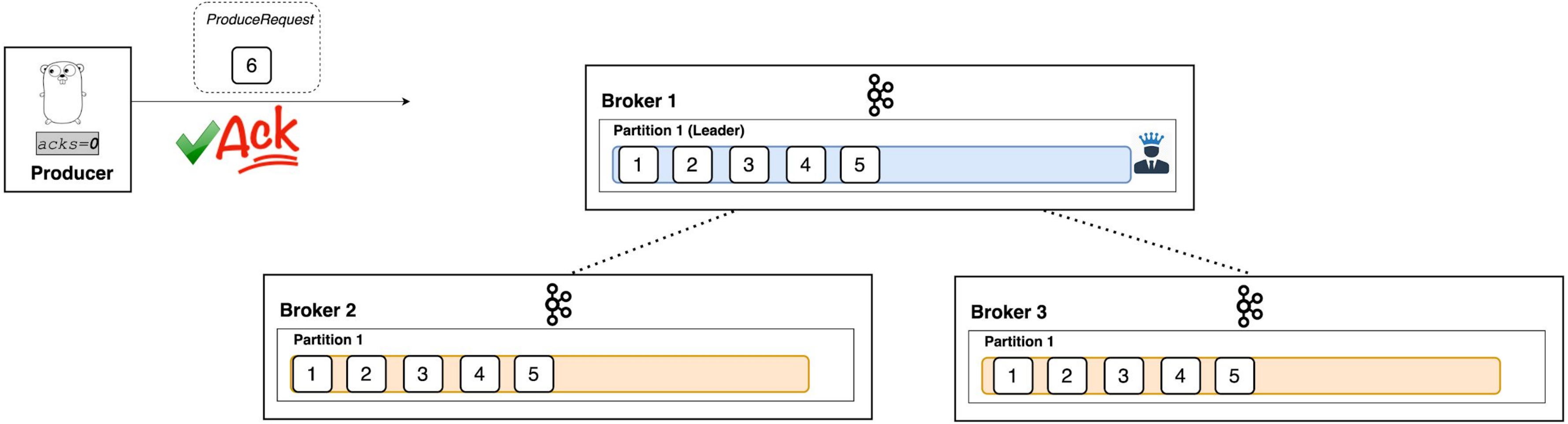
Zookeeper

- Keeps track of status of the Kafka cluster nodes and it also keeps track of Kafka topics, partitions etc.
- Plays key part in deployments

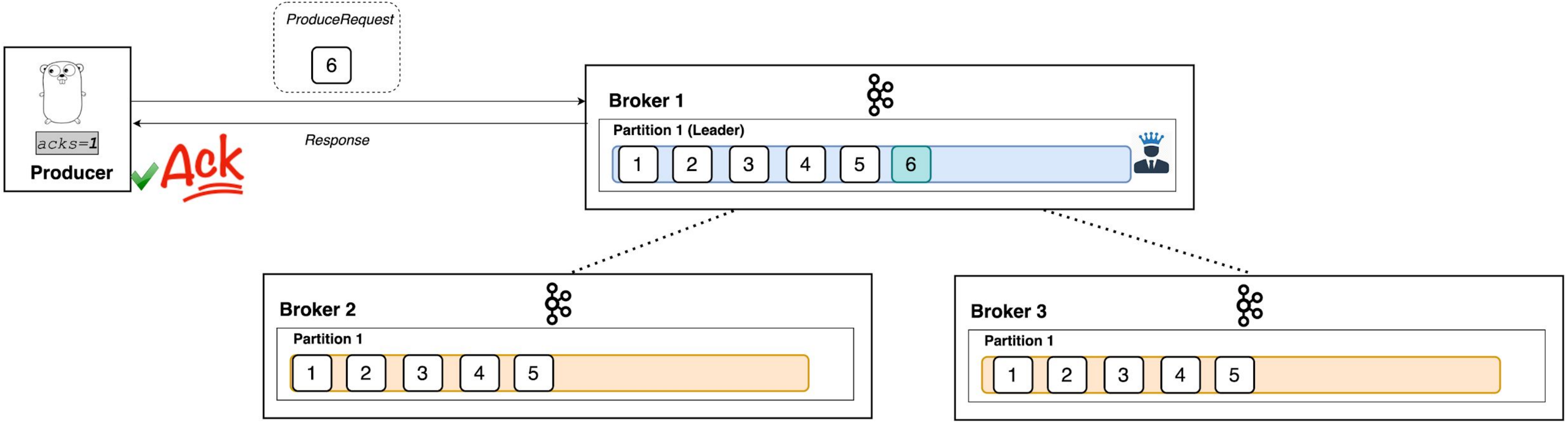
Producers

- Publish messages to a specific topic
- Connect to single broker (kafka handles routing to correct broker)
- Can choose to receive acknowledgment of writes
    - Acks=0 - don't wait for ack (possible data loss)
    - Acks=1 - wait for leader ack (limited data loss)
    - Acks=all - leader plus replicas acknowledge (no data loss)

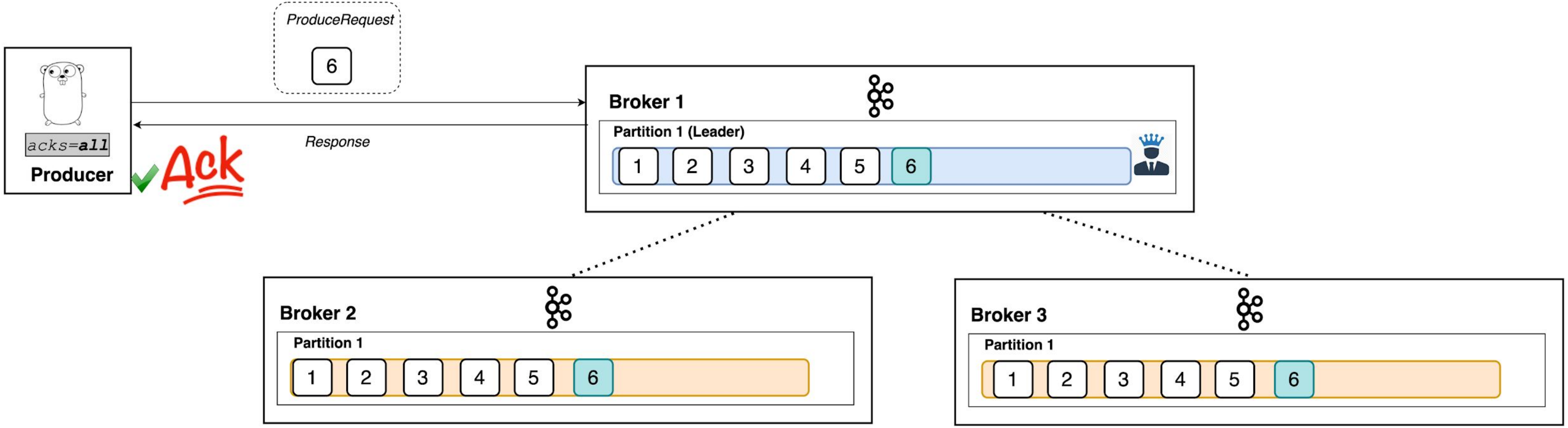
Acks=0


ProduceRequest
6
acks=0
Producer
Ack
Broker 1
Partition 1 (Leader)
1
2
3
4
5
Broker 2
Partition 1
1
2
3
4
5
Broker 3
Partition 1
1
2
3
4
5

Acks=1


ProduceRequest
6
acks=1
Producer
Ack
Response
Broker 1
Partition 1 (Leader)
1
2
3
4
5
6
Broker 2
Partition 1
1
2
3
4
5
Broker 3
Partition 1
1
2
3
4
5

Acks=all


ProduceRequest
6
acks=all
Producer
Ack
Response
Broker 1
Partition 1 (Leader)
1
2
3
4
5
6
Broker 2
Partition 1
1
2
3
4
5
6
Broker 3
Partition 1
1
2
3
4
5
6

Consumer Groups

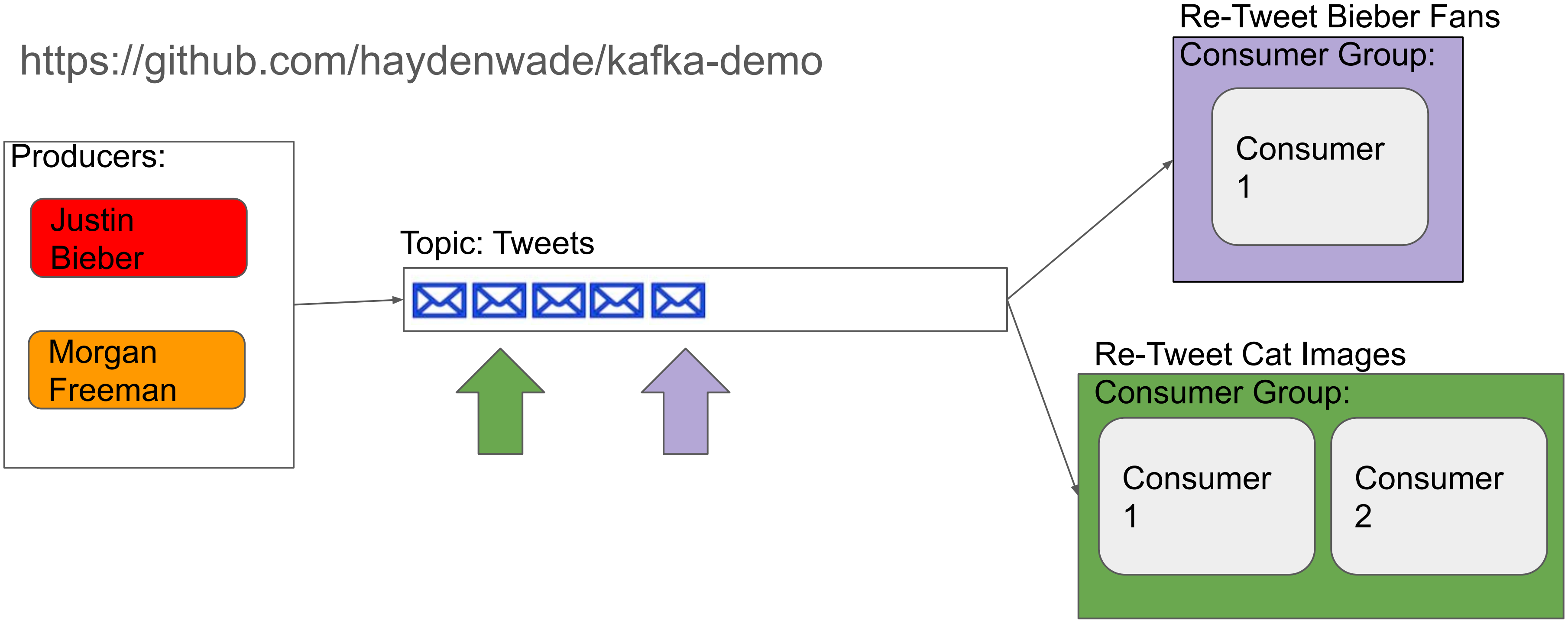
- Maintains an "offset" within a given topic to know where it left off
- Consumers within a group all read from different topic partitions

Consumers

- Subscribe to one or more topics
- Runs some computation on the message
- Multiple start settings (start from latest, start from beginning of topic)
- Connect to single broker (kafka handles routing to correct broker)
- Read messages in order per partition

Demo

https://github.com/haydenwade/kafka-demo



Monitoring - Broker Example Metrics

- `TotalTimeMs` - measures the total time taken to service a request (be it a produce, fetch-consumer, or fetch-follower request)
- `RequestsPerSec` - the rate of requests from your producers, consumers, and followers
- `OfflinePartitionsCount` - the number of partitions without an active leader

Monitoring - Producer Example Metrics

- Request Latency Average - The average request latency is a measure of the amount of time between when KafkaProducer.send() was called until the producer receives a response from the broker.

Monitoring - Consumer Example Metrics

- Record lag - the calculated difference between a consumer’s current log offset and a producer’s current log offset

Good Reads:

- Slack Engineering - Migrating to Kafka
  - https://slack.engineering/scaling-slacks-job-queue-687222e9d100
- Kafka Acks Explained
  - https://medium.com/better-programming/kafka-acks-explained-c0515b3b707e

Sources

- [Apache Kafka Docs](https://kafka.apache.org/)
- [Datadog - Monitoring](https://www.datadoghq.com/blog/monitoring-kafka-performance-metrics/)
- [Rabbitmq vs Kafka](https://medium.com/better-programming/rabbitmq-vs-kafka-1ef22a041793)