OpenNMS Kafka Training for NCR

Tarus Balog

tarus@opennms.com

http://www.opennms.org/~tarus/Class

Apache Kafka

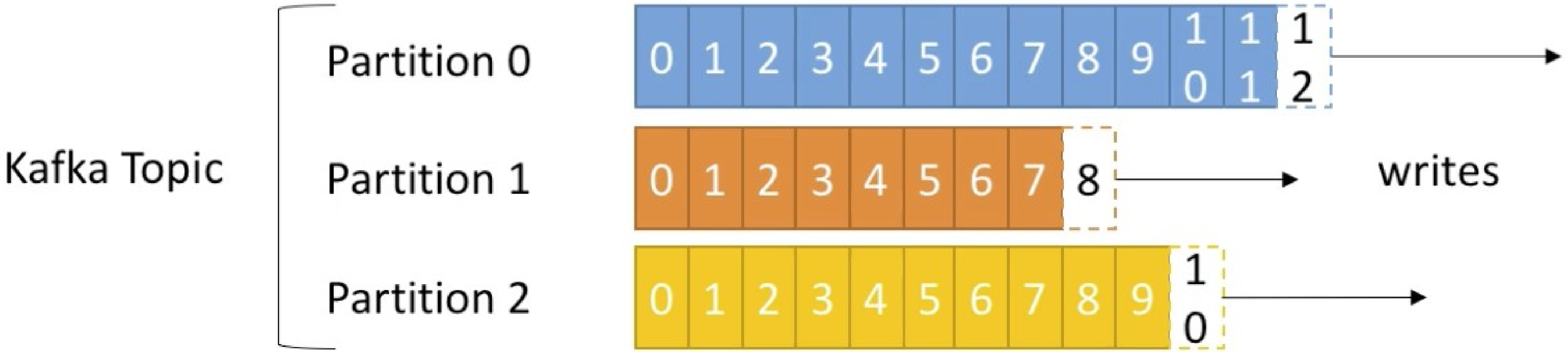
Kafka is an event streaming platform

http://kafka.apache.org/

Data Structure Overview

- Topics (a particular stream of data, similar to a table in a database).
- Partitions (a topic is split in partitions).
- Offsets (each message within a partition gets an incremental ID, called offset; starting at 0).
- Order is guarantee only within a partition (not across partitions or at the topic level).
- Data is kept for a limited time.
- Once the data is written to a partition, it can't be changed (immutability).
- Data is assigned randomly to a partition unless a key is provided (round-robin).

Data Structure Overview

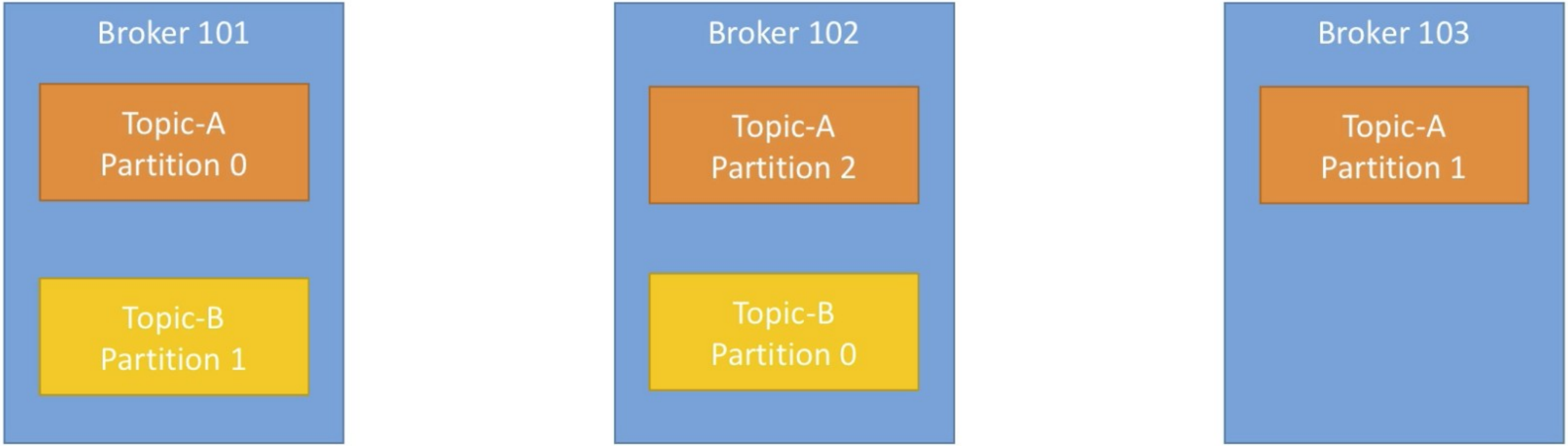


Brokers

- A Kafka cluster is composed of multiple brokers (servers).
- Each broker is identified with its ID (integer).
- Each broker contains certain topic partitions.
- After connecting to any broker, you'll be connected to the entire cluster.

Broker Structure Overview

- Example of Topic-A with **3 partitions**
- Example of Topic-B with **2 partitions**



Broker 101

Topic-A Partition 0

Topic-B Partition 1

Broker 102

Topic-A Partition 2

Topic-B Partition 0

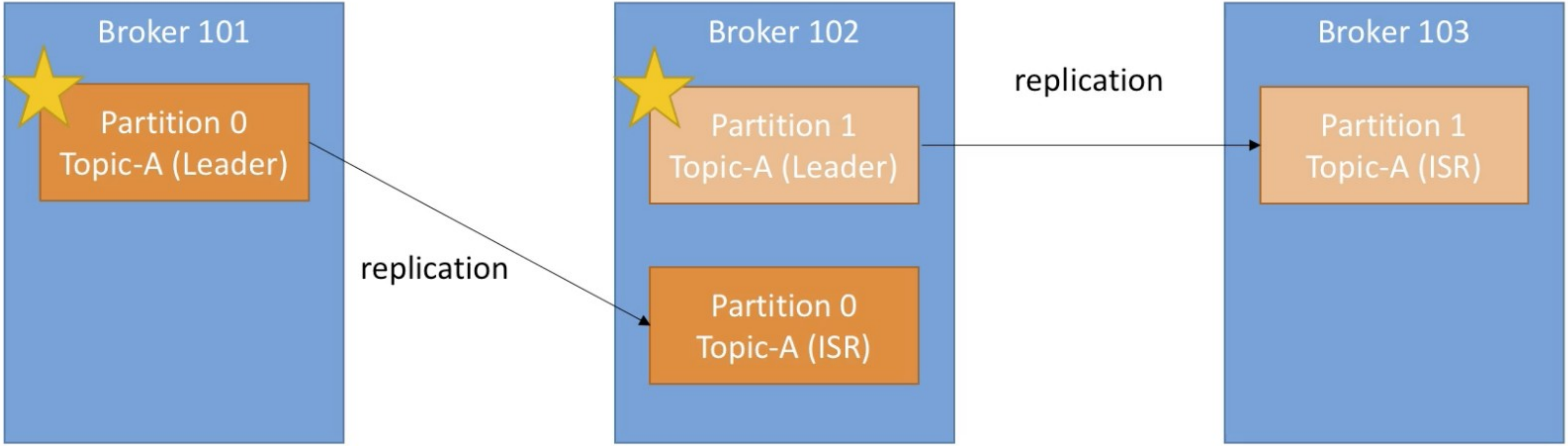
Broker 103

Topic-A Partition 1

Replication

- Topics should have a replication factor greater than one.
- At any time only ONE broker can be a leader for a given partition (managed by Zookeeper).
- Only that leader can receive and serve data for a partition.
- The other brokers will synchronize the data.
- Therefore each partition has one leader and multiple ISR (in-sync replica).

Replication



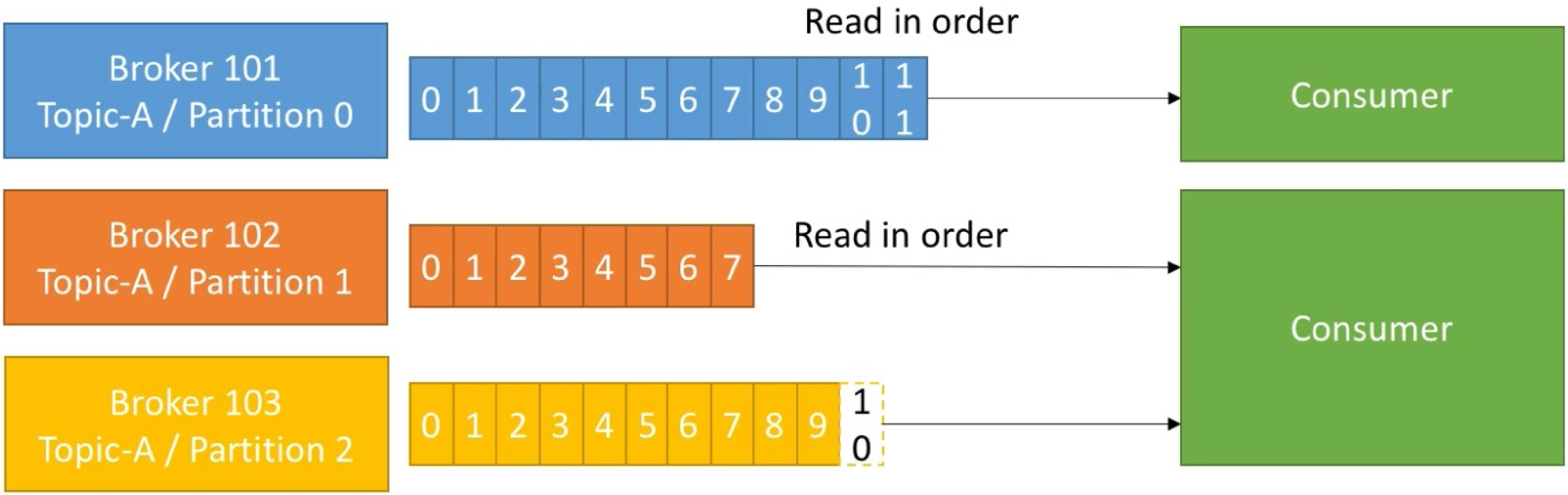
Producers

- Producer write data to topics (which is made of partitions).
- Producers automatically know to which broker and partition to write to.
- In case of broker failures, produces will automatically recover.
- acks=0: producer won't wait for acknowledgement (possible data loss).
- acks=1: producer will wait for leader acknowledgment (limited data loss).
- acks=all: leader + replicas acknowledgement (no data loss).
- Producers can choose to send a key with the message (it can be anything, a string, a number, etc).
- If key is null, data is sent round robin.
- If a key is sent, then all messages for that key will always go to the same partition (chosen by the broker; which could lead to inconsistencies if the number of partitions is altered).
- A key is basically sent if you need message ordering for a specific field.

Consumers

- Consumers read data from a topic
- Consumers know which broker to read from.
- In case of broker failures, consumer know how to recover.
- Data is read in order within each partition.

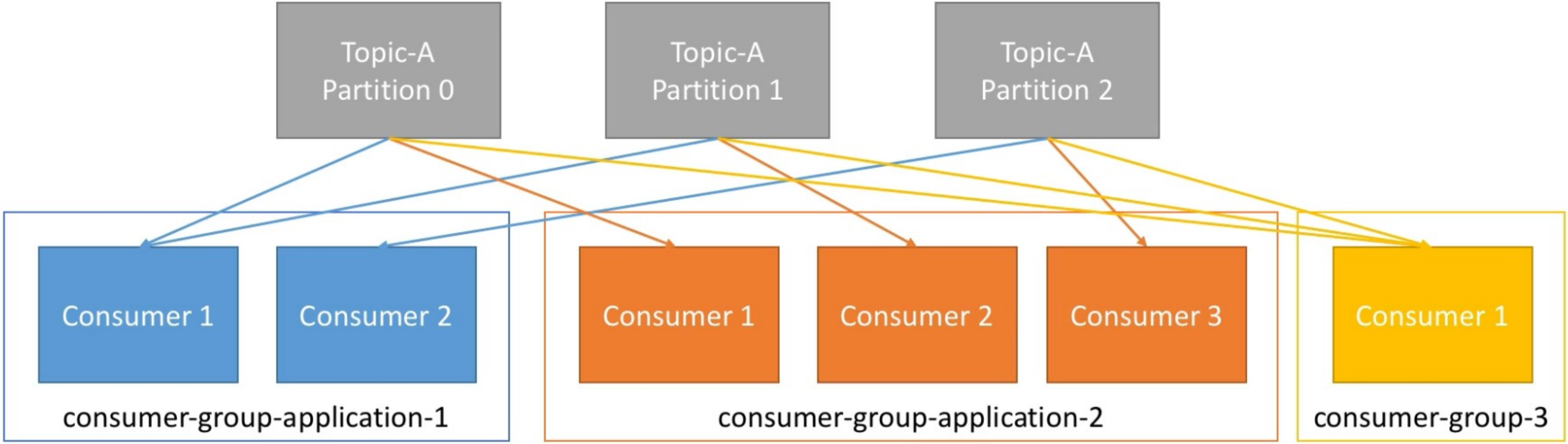
Consumers


Broker 101
Topic-A / Partition 0
0 1 2 3 4 5 6 7 8 9 10 11
Read in order
Consumer
Broker 102
Topic-A / Partition 1
0 1 2 3 4 5 6 7
Read in order
Consumer
Broker 103
Topic-A / Partition 2
0 1 2 3 4 5 6 7 8 9 10

Consumer Groups

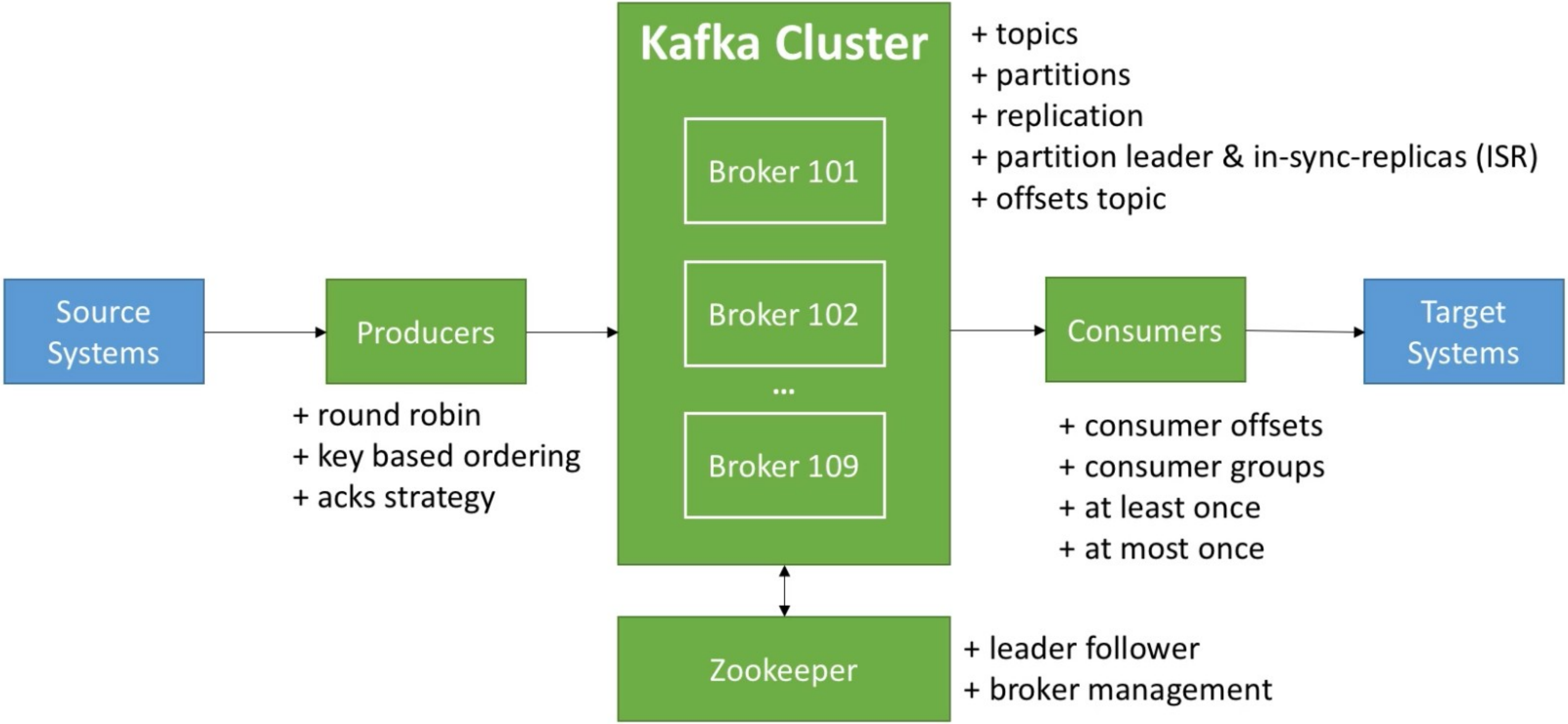
- Consumer read data in consumer groups
- Each consumer within a group reads from exclusive partitions
- If you have more consumers than partitions, some consumers will be inactive.

Consumer Groups



Zookeeper

- Zookeeper manages brokers.
- Zookeeper helps performing leader election for partitions.
- Zookeeper sends notifications to Kafka in case of changes.
- Kafka CANNOT work without Zookeeper.
- Zookeeper operates with an odd number of servers



Kafka Cluster

Broker 101

Broker 102

...

Broker 109

+ topics
+ partitions
+ replication
+ partition leader & in-sync-replicas (ISR)
+ offsets topic

Source Systems

Producers

+ round robin
+ key based ordering
+ acks strategy

Consumers

+ consumer offsets
+ consumer groups
+ at least once
+ at most once

Target Systems

Zookeeper

+ leader follower
+ broker management