



شرکت خدمات انفورماتیک  
informatics services corporation

## آشنایی ، استفاده و مدیریت آپاچی کافکا

[www.isc.co.ir](http://www.isc.co.ir)

طبقه بندی: عادی

تاریخ: ۱۳۹۹/۱۲/۰۳



- ١ Introduction - Architecture - Concept
- ٢ Primary Elements
- ٣ Messages – Metadata- Schema
- ٤ Messages - Order and Delivery
- ٥ Consumers
- ٦ Configuration and Dependency



- Clustering – Scalability** ۷
- Clustering - Partition** ۸
- Clustering - Replication** ۹
- Broker – Fail Detection - Imbalance** ۱۰
- Logs – Compression - Retention** ۱۱
- Security – TLS – Authentication - Authorization** ۱۲





شرکت خدمات انفورماتیک  
informatics services corporation

## 1 - Introduction - Architecture and Concept



# Introduction

- Community
- Distributed event streaming platform
- Capable of handling trillions of events a day
- Based on an abstraction of a distributed commit log
- Created and open sourced by LinkedIn in 2011



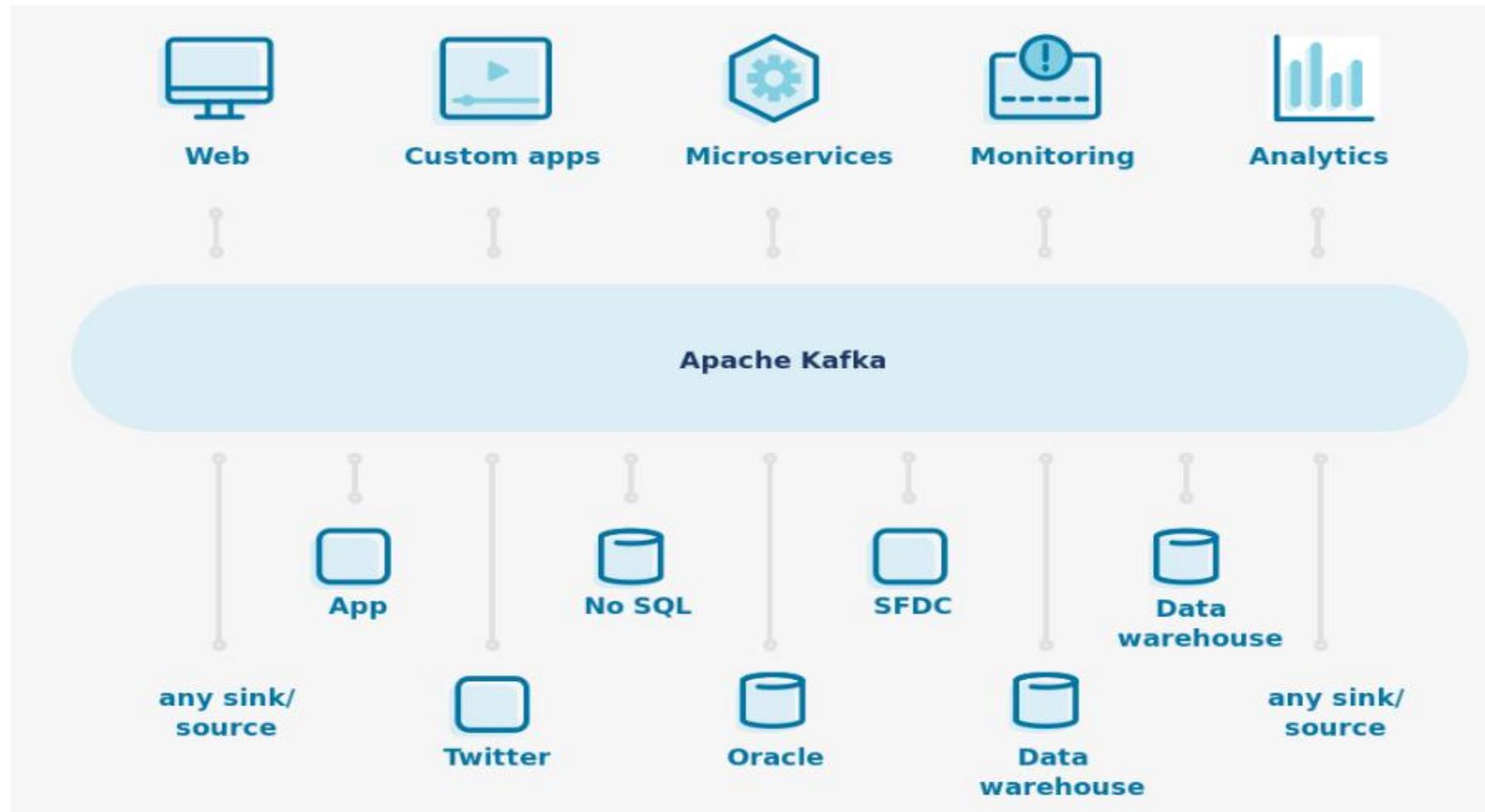
شرکت خدمات انفورماتیک

informatics service corporation



# Introduction

## Where Apache Kafka fits in



# Usage

Thousands of companies are built on Kafka



# How can Kafka help you



Publish + Subscribe



Store



Process



## How does Kafka work?

- Kafka is a distributed system
- Consisting of servers and clients
- Communicate via a high-performance TCP



شرکت خدمات اینفورماتیک

informatics services corporation



# Kafka Client

- Java
- Scala
- C/C++
- Python
- Go (AKA golang)
- Erlang
- .NET
- Clojure
- Ruby
- Node.js
- Proxy( Http Rest)
- Perl
- Rust
- Swift
- PHP



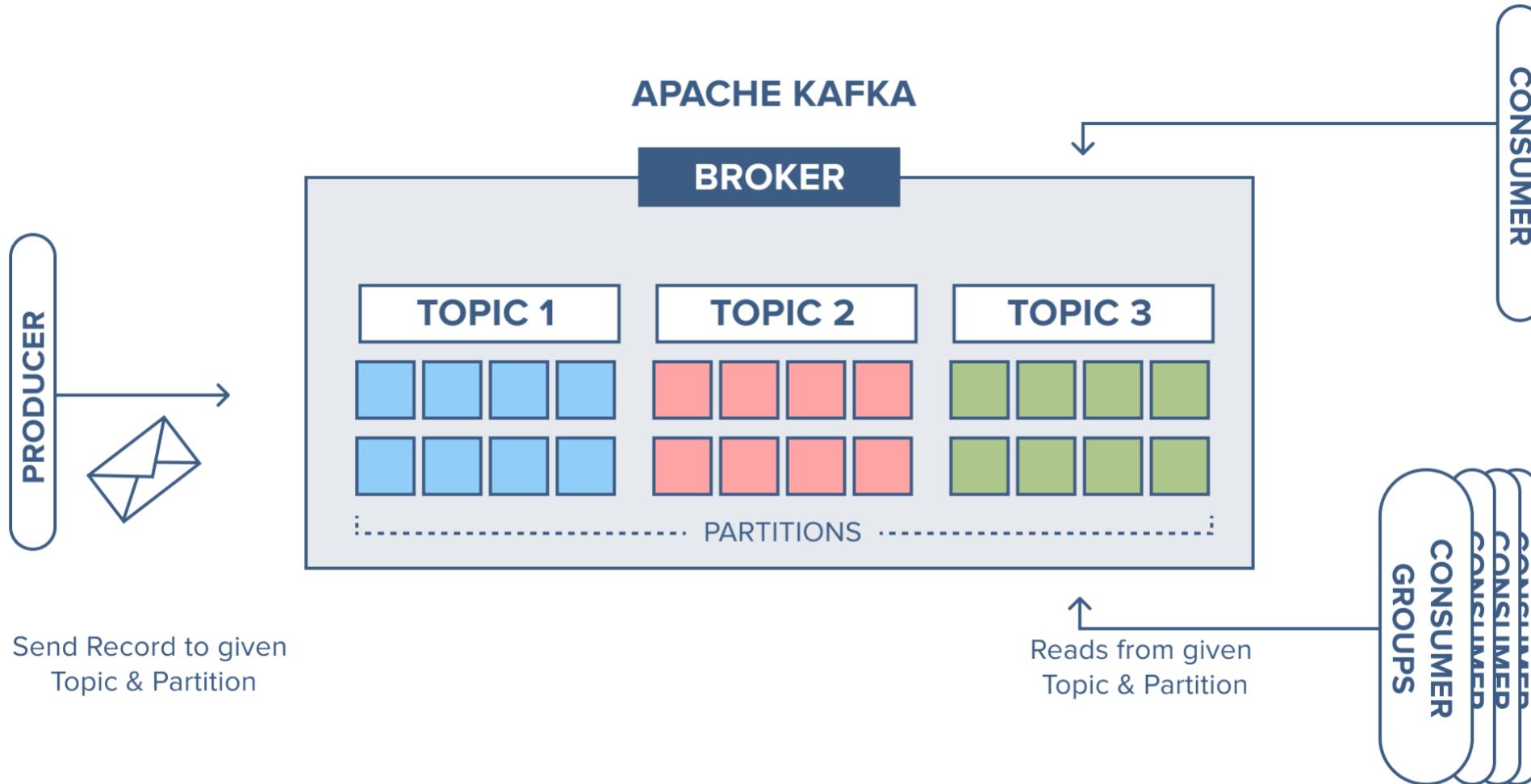


شرکت خدمات انفورماتیک  
informatics services corporation

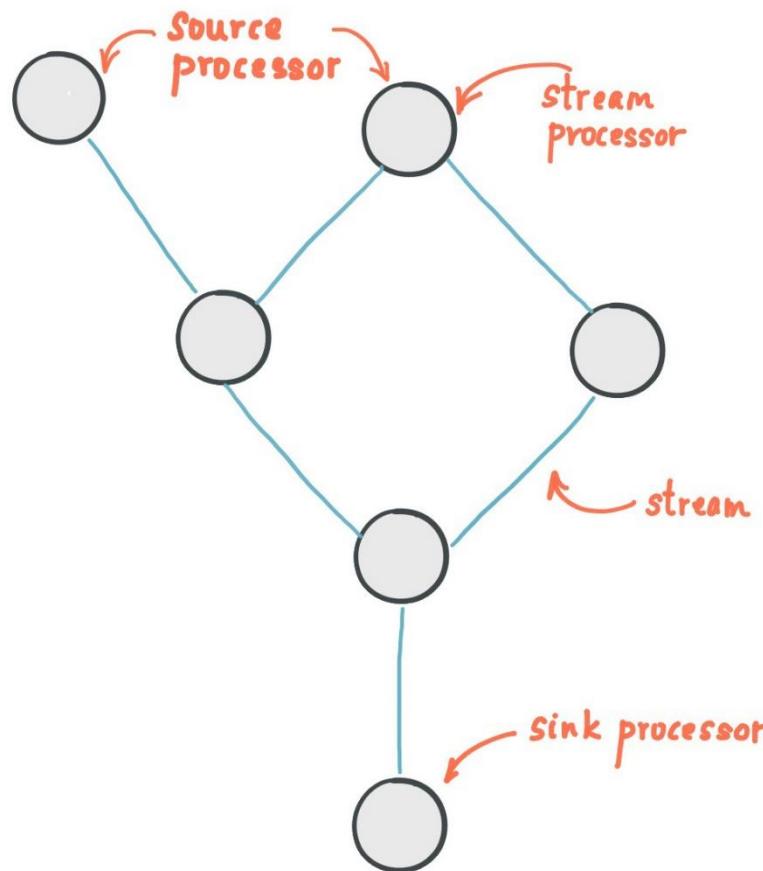
## 2 - Primary Elements



# Main Concepts & Terminology



# Stream Processing Topology



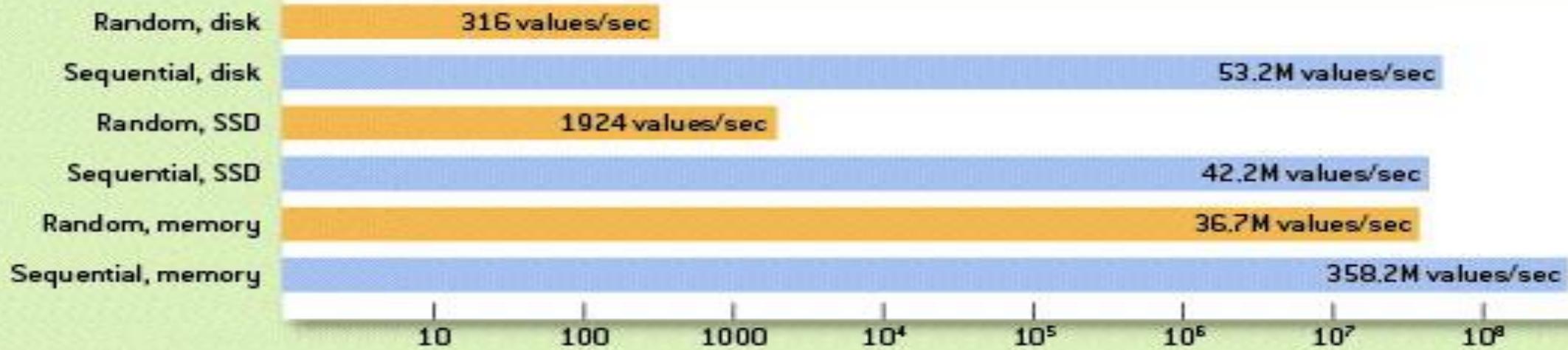
PROCESSOR TOPOLOGY



Don't fear the file system

FIGURE  
**3**

## Comparing Random and Sequential Access in Disk and Memory



Note: Disk tests were carried out on a freshly booted machine (a Windows 2003 server with 64-GB RAM and eight 15,000-RPM SAS disks in RAID5 configuration) to eliminate the effect of operating-system disk caching. SSD test used a latest-generation Intel high-performance SATA SSD.



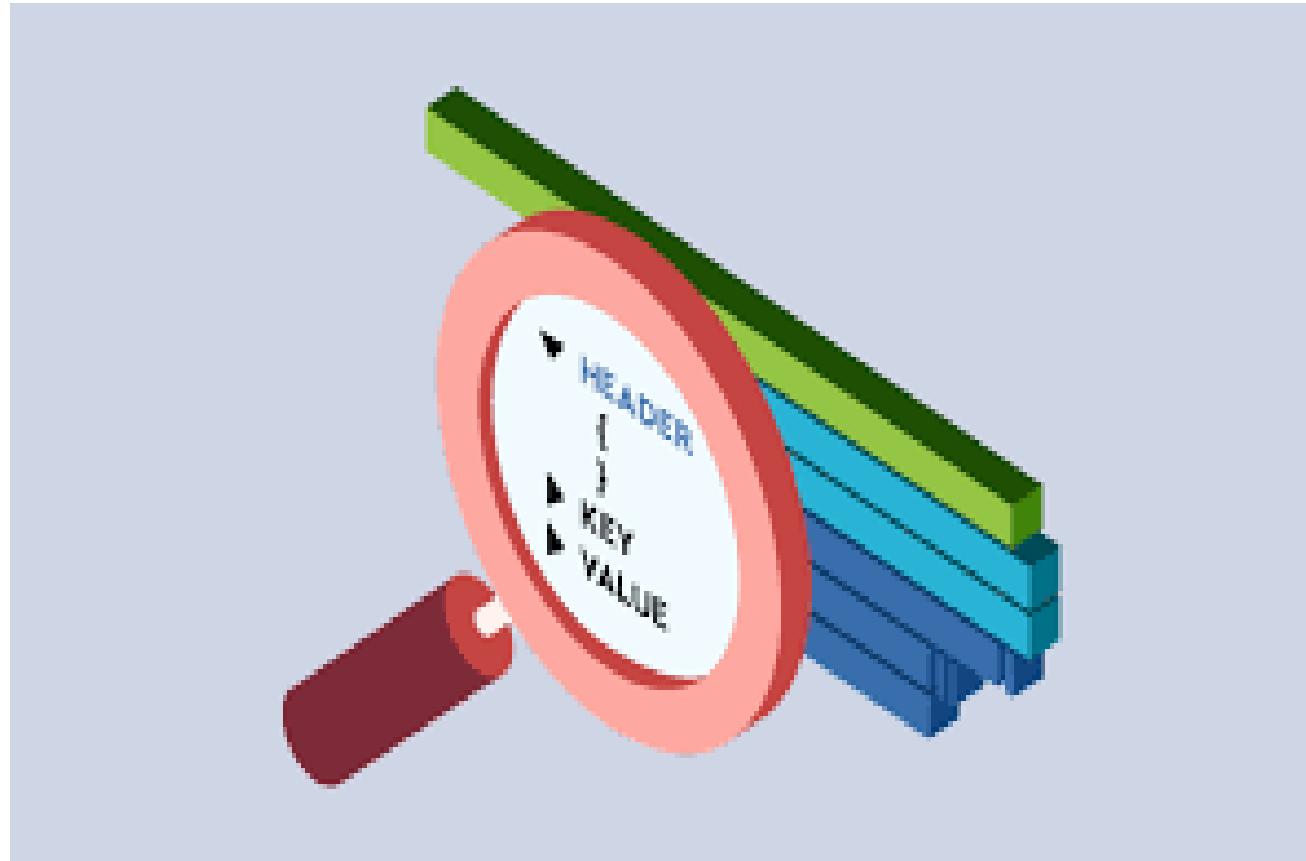


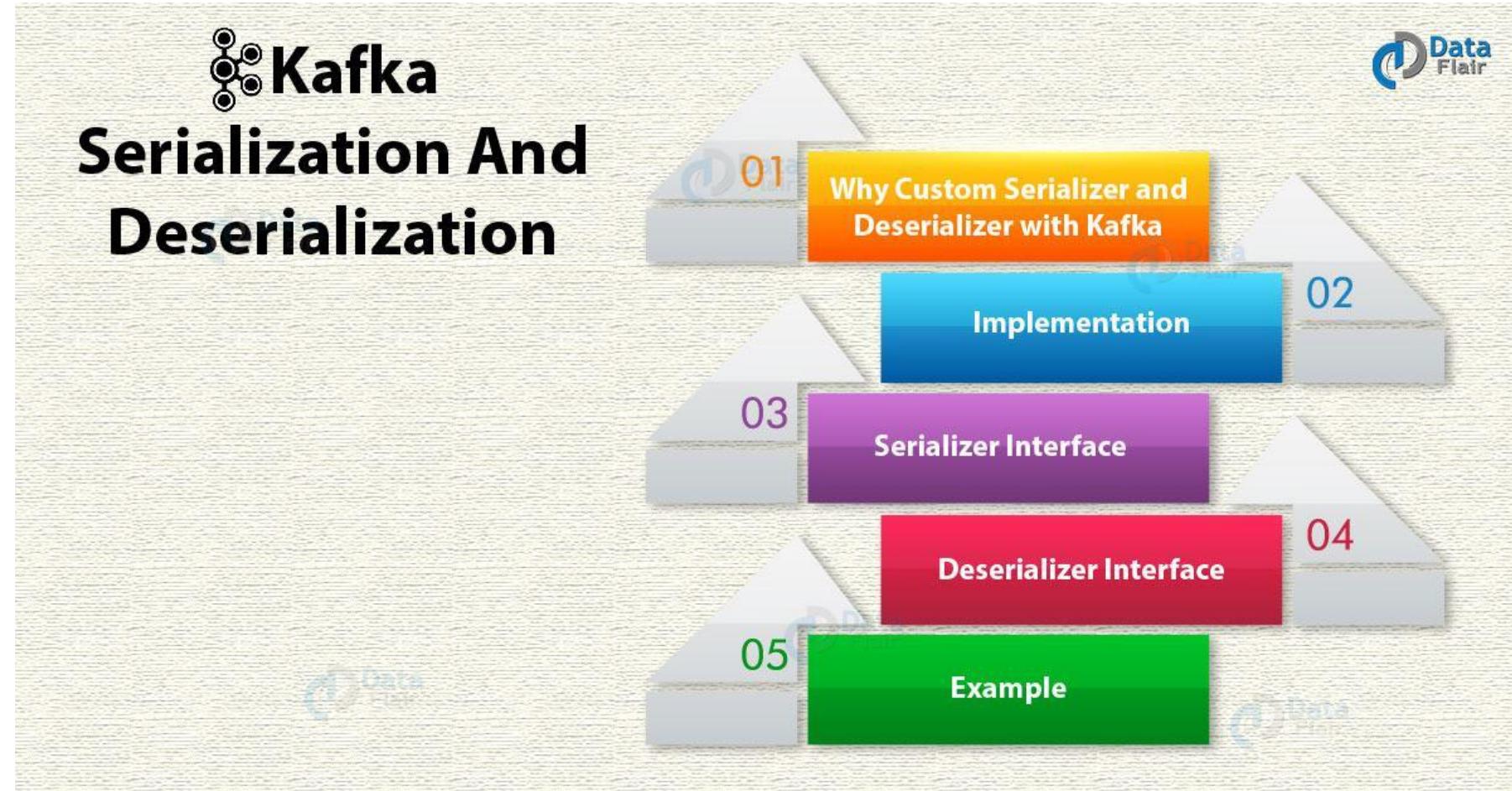
شرکت خدمات انفورماتیک  
informatics services corporation

### 3 - Messages – Metadata- Schema



## Use the power of record headers





# Message Serializer & Deserializer



# Serializer Interface

```
public interface Serializer extends Closeable {  
    void configure(Map<String, ?> var1, boolean var2);  
    byte[] serialize(String var1, T var2);  
    void close();  
}
```



شرکت خدمات اینفورماتیک  
informatics services corporation



# Deserializer Interface

```
public interface Deserializer extends Closeable {  
    void configure(Map<String, ?> var1, boolean var2);  
    T deserialize(String var1, byte[] var2);  
    void close();  
}
```



شرکت خدمات انفورماتیک  
informatics services corporation



# Example

```
public class User {  
    private String name;  
    private int age;  
  
    // Constructor  
    // Getter & Setter  
}
```

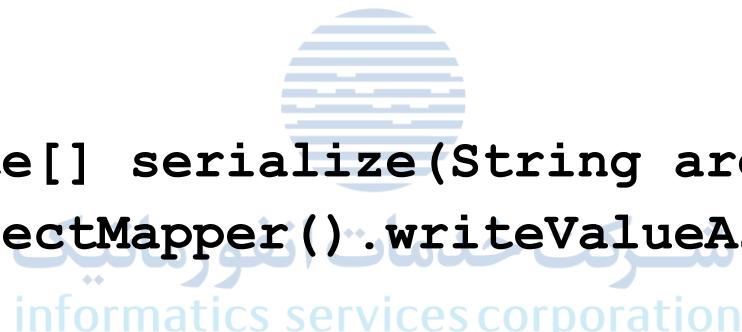


شرکت خدمات انفورماتیک  
informatics services corporation



# Example

```
public class UserSerializer implements Serializer {  
  
    @Override public void configure(Map<String, ?> map, boolean b) {  
    }  
  
    @Override public byte[] serialize(String arg0, User arg1) {  
        return new ObjectMapper().writeValueAsString(arg1).getBytes();  
    }  
  
    @Override public void close() {  
    }  
  
}
```



# Example

```
public class UserDeserializer implements Deserializer {  
  
    @Override public void close() {  
    }  
  
    @Override public void configure(Map<String, ?> arg0, boolean arg1) {  
    }  
  
    @Override public User deserialize(String arg0, byte[] arg1) {  
        return new ObjectMapper().readValue(arg1, User.class);  
    }  
}
```



# Producer and Consumer Configuration

```
// Producer side  
props.put("value.serializer",  
          "com.isc.nps.npsd.kafka.serializers.UserSerializer");  
  
//Consumer side  
props.put("value.deserializer",  
          "com.isc.nps.npsd.kafka.deserializer.UserDeserializer");
```



شرکت خدمات انفورماتیک  
informatics services corporation





شرکت خدمات انفورماتیک  
informatics services corporation

## 4 - Messages - Order and Delivery

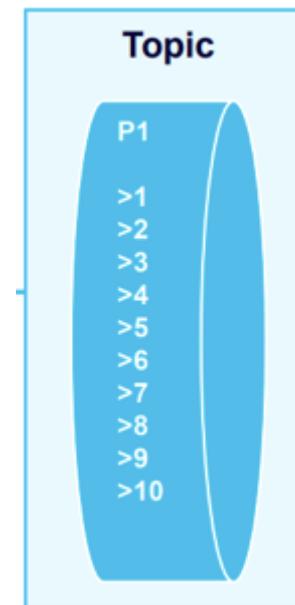


# Message Order - Single Partitioning

```
./kafka-topics.sh --create \  
--zookeeper localhost:2181/kafka \  
--replication-factor 1 --partitions 1 \  
--topic my-topic
```

```
./kafka-console-producer.sh \  
--broker-list localhost:9092 \  
--topic my-topic
```

```
> 1  
> 2  
> 3  
> 4  
> 5  
> 6  
> 7  
> 8  
> 9  
> 10
```

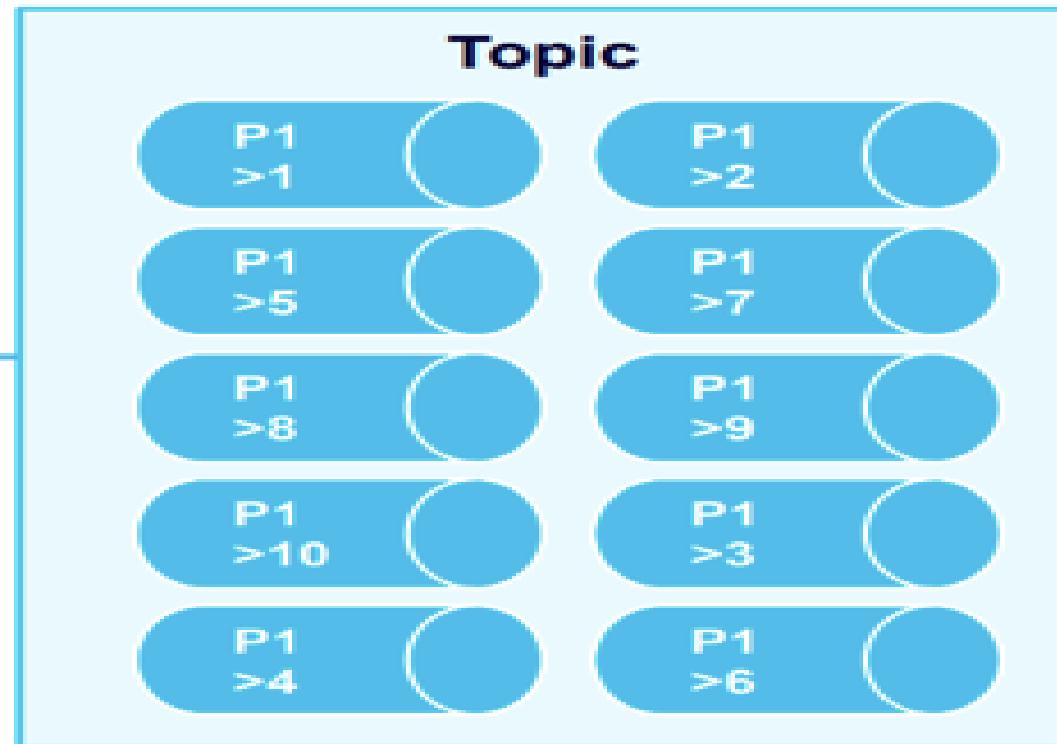


# Message Order - Multiple Partitioning

```
./kafka-console-producer.sh \
--broker-list localhost:9092 \
--topic my-topic
```

> 1  
> 2  
> 3  
> 4  
> 5  
> 6  
> 7  
> 8  
> 9  
> 10

```
./kafka-topics.sh --create \
--zookeeper localhost:2181/kafka \
--replication-factor 1 --partitions 10 \
--topic my-topic
```



```
./kafka-console-consumer.sh \
--bootstrap-server localhost:9092 \
--topic my-topic \
--from-beginning
```

> 2  
> 5  
> 9  
> 3  
> 4  
> 7  
> 6  
> 10  
> 8  
> 1



## How to achieve strict ordering?

- ❖ Most businesses require things to be in the correct order



- ❖ You must use only one partition
- ❖ Prefer messages to be ordered based on a certain property in the message
- ❖ Use multiple partitions and still reach the result you wish



# Message Order - Keying

Key: Costco  
Value: 400

Key: Walmart  
Value: 400

Key: Target  
Value: 400

Key: BestBuy  
Value: 400

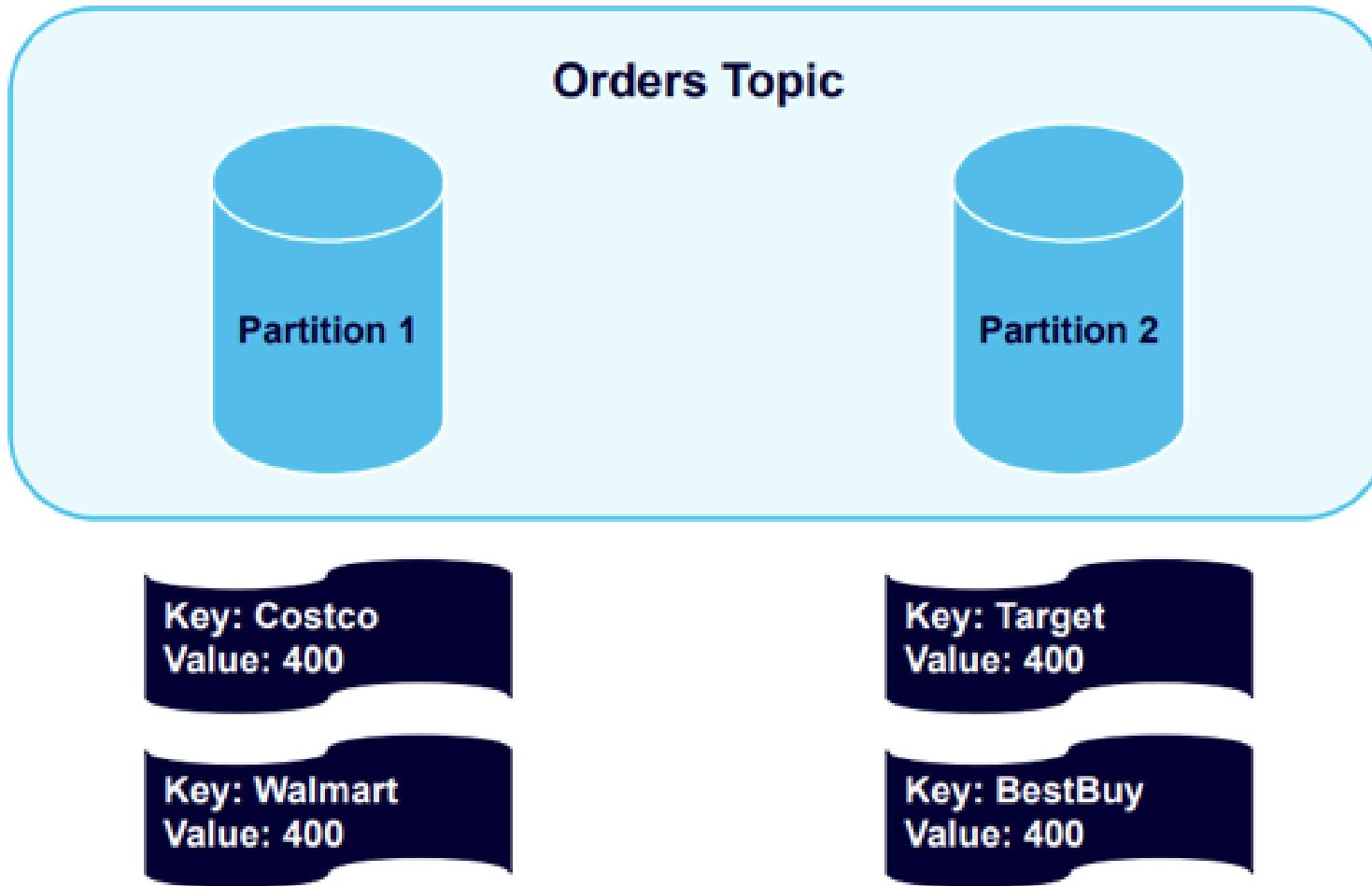
Orders Topic

Partition 1

Partition 2



# Message Order - Keying



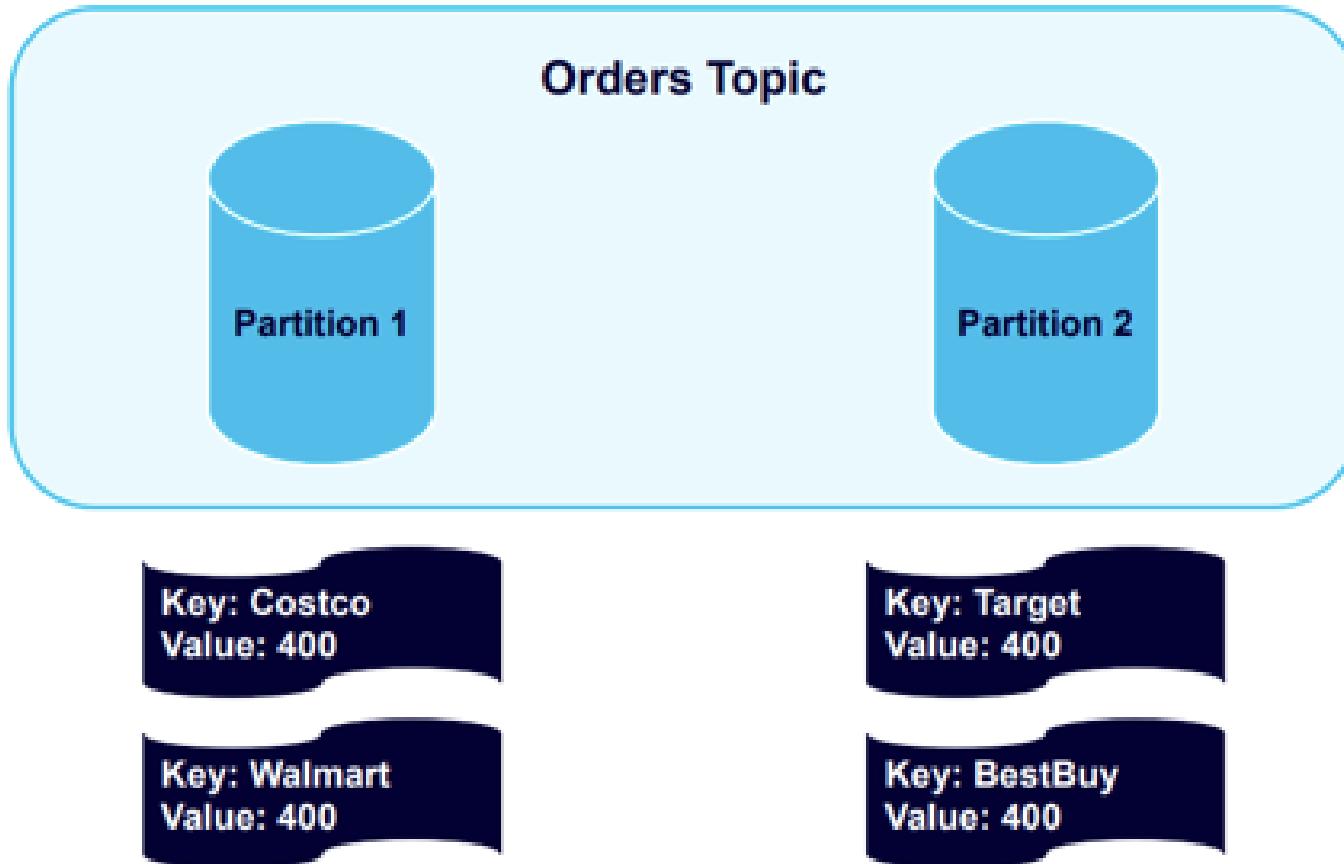
# Message Order - Keying

Key: Costco  
Value: 100

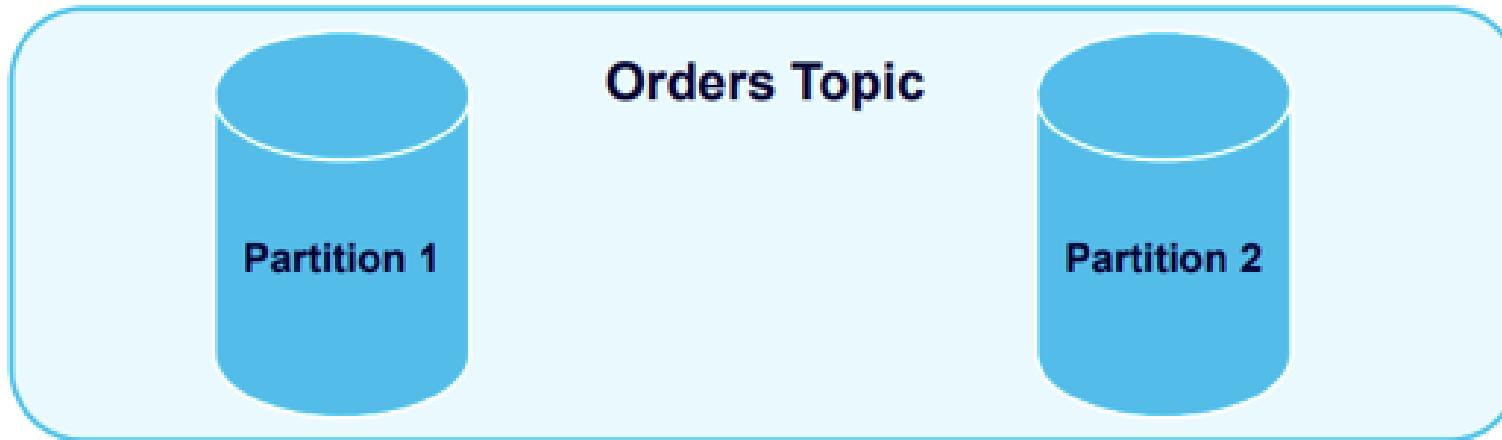
Key: Walmart  
Value: 200

Key: Target  
Value: 100

Key: BestBuy  
Value: 200



# Message Order - Keying



Key: Costco  
Value: 400

Key: Walmart  
Value: 400

Key: Costco  
Value: 100

Key: Walmart  
Value: 200

Key: Target  
Value: 400

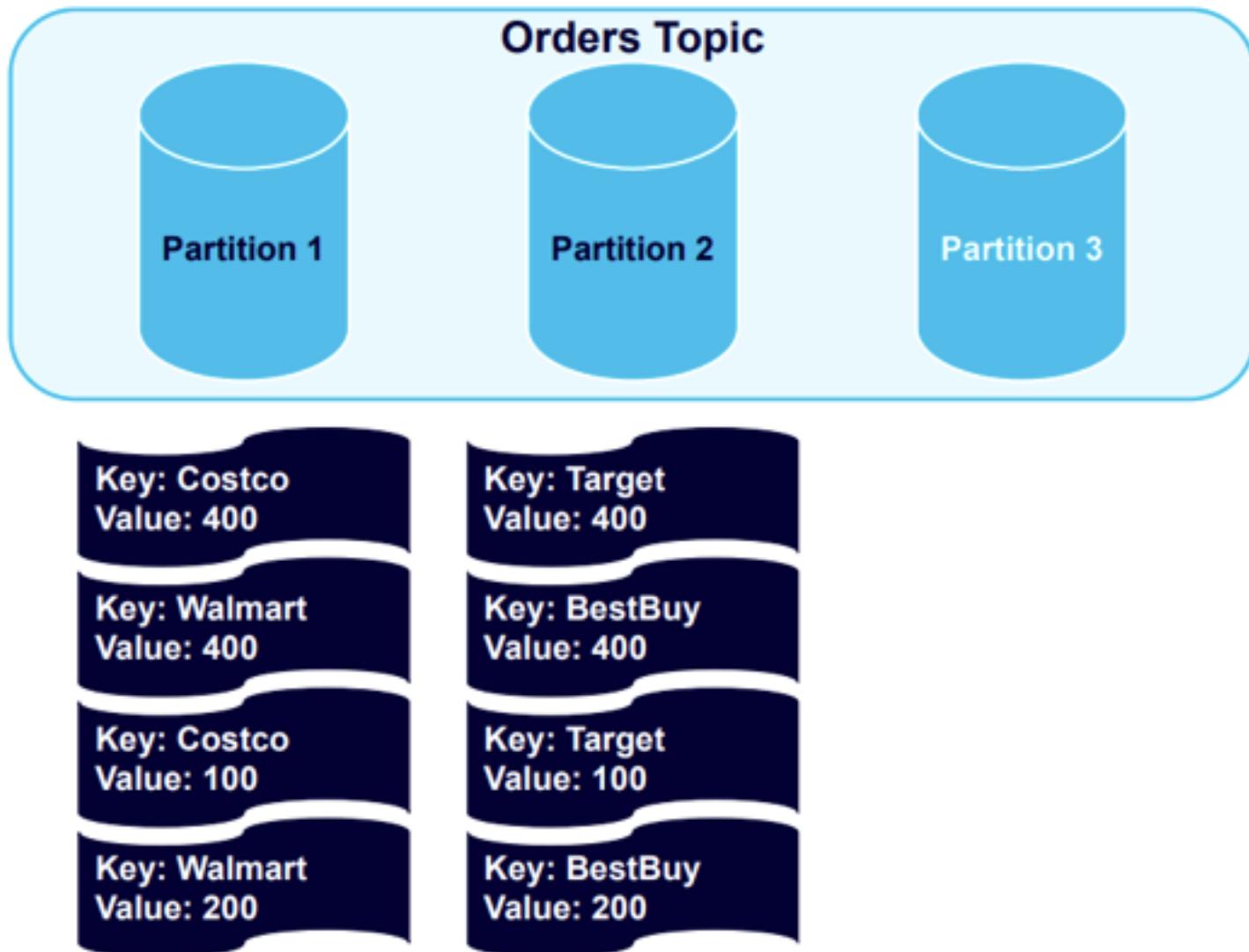
Key: BestBuy  
Value: 400

Key: Target  
Value: 100

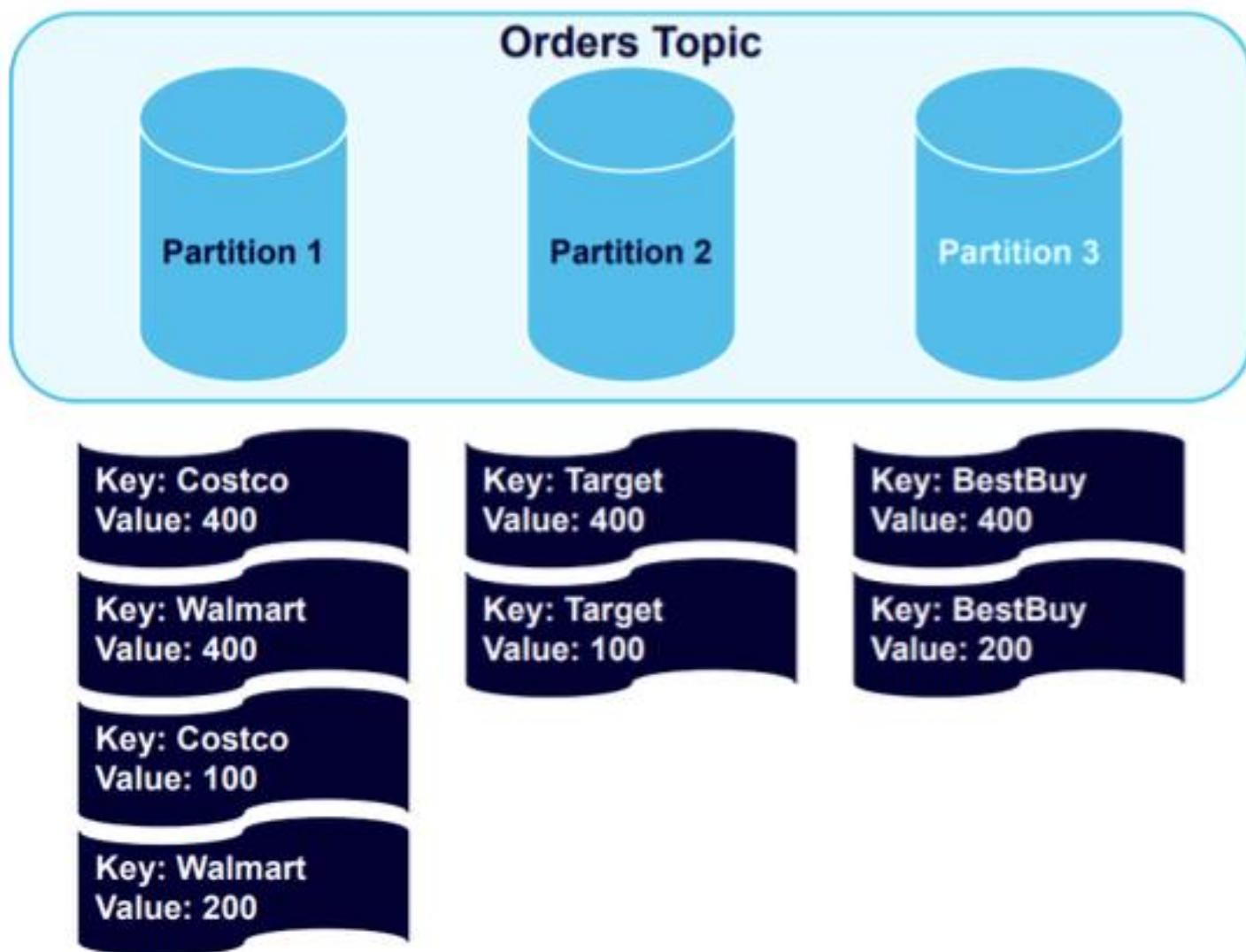
Key: BestBuy  
Value: 200



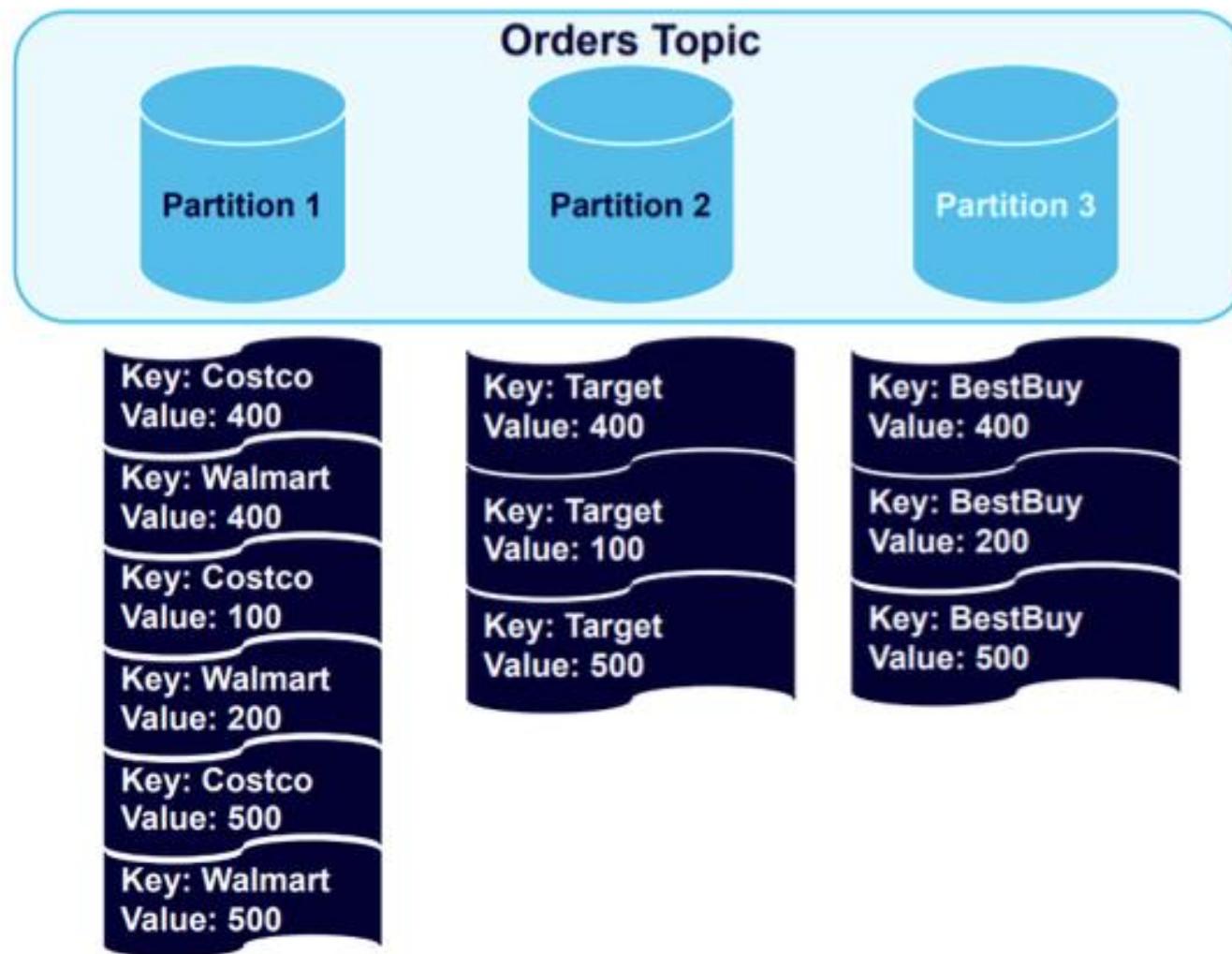
# Message Order - Keying and Partition Rebalancing



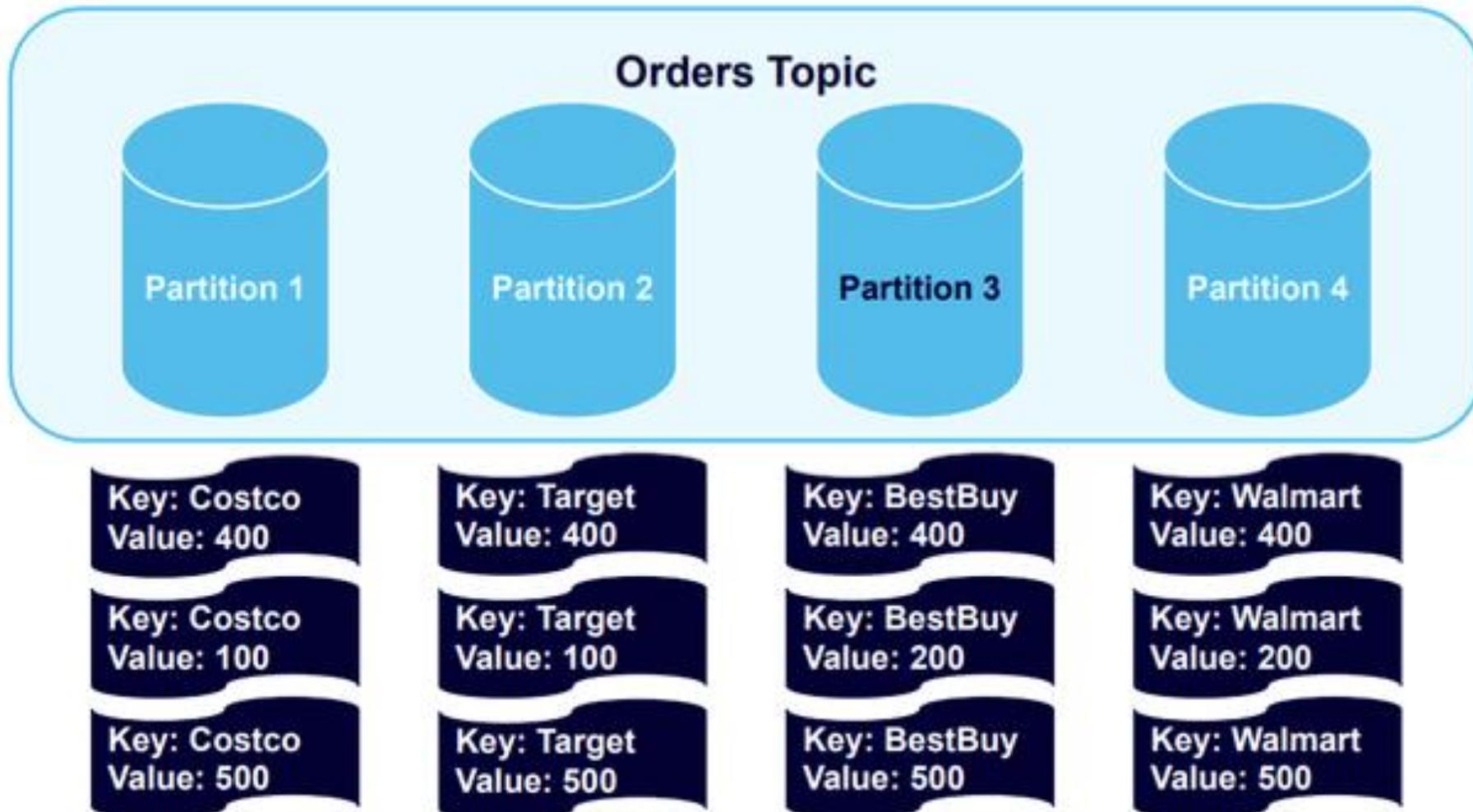
# Message Order - Keying and Partition Rebalancing



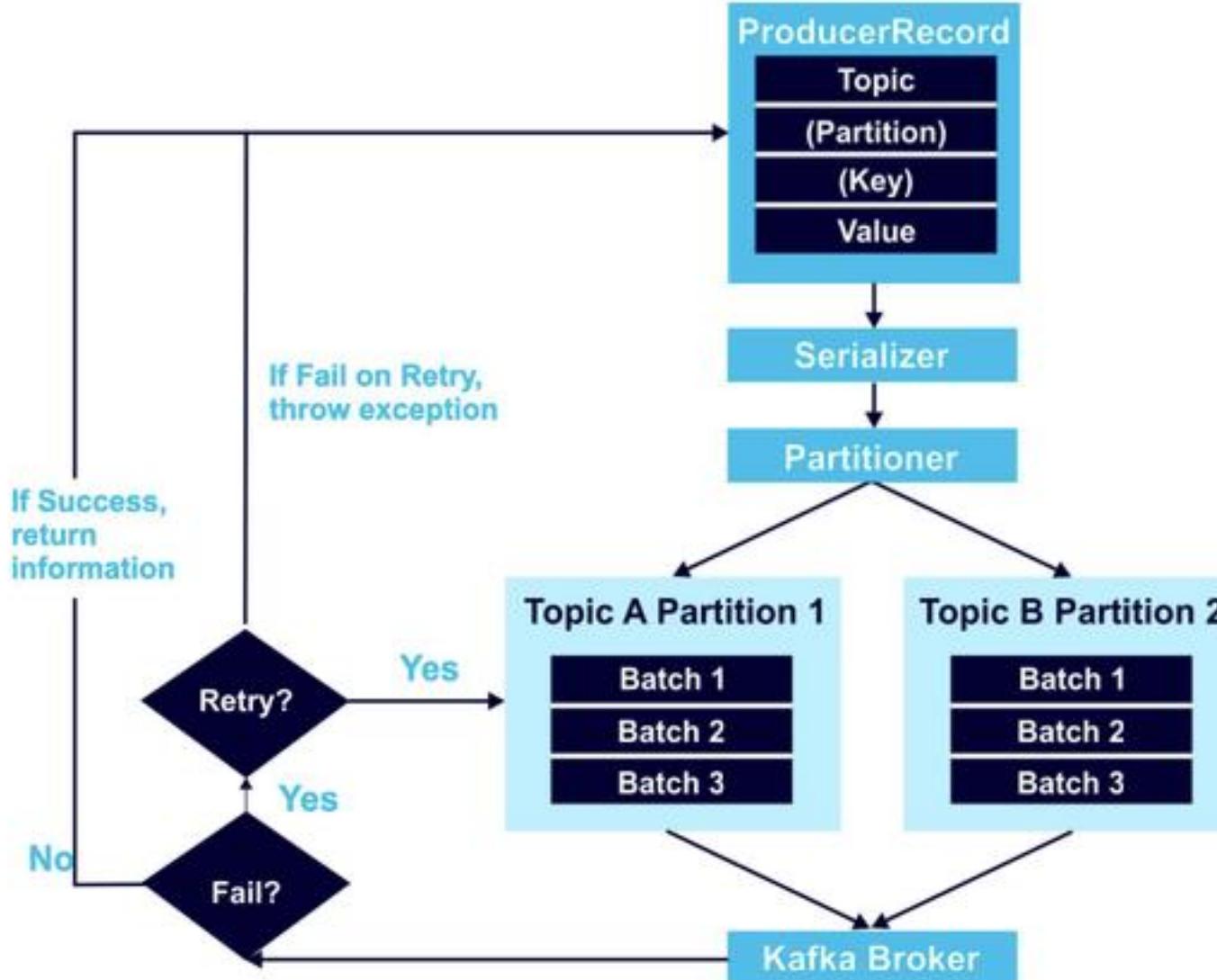
# Message Order - Keying and Partition Rebalancing



# Message Order - Keying and Partition Rebalancing



# Message Order - Ensuring that Data is Always Sent in Order



# Message Delivery and Durability

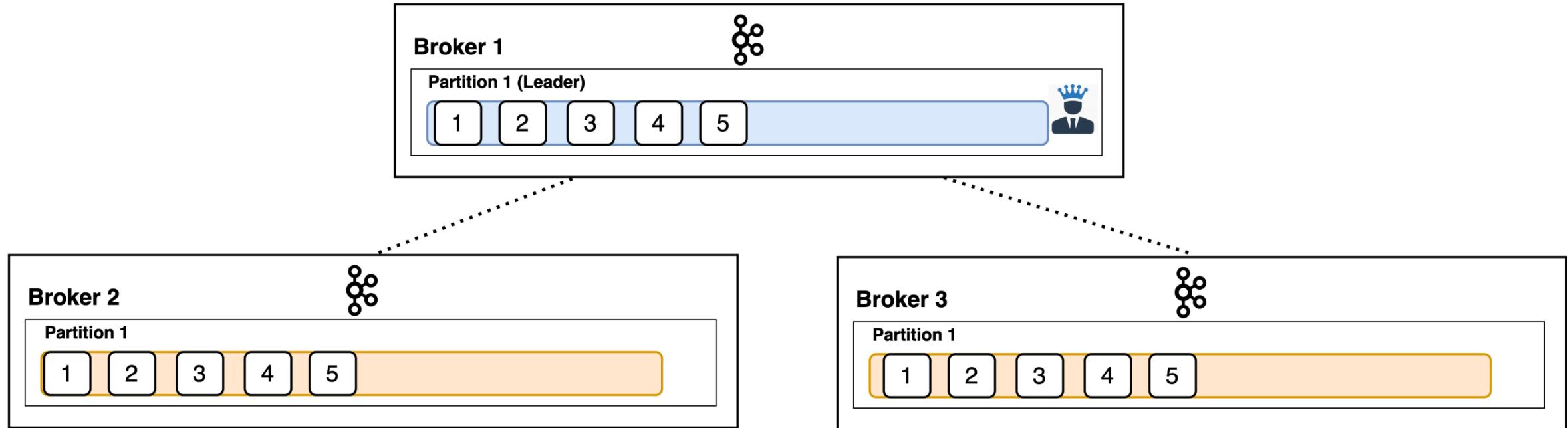
- For Data Durability
  - Producer configuration setting “acks”
- “acks” specifies how many acknowledgement receives
  - To consider record / event delivered to the broker.
- “acks” options choose
  - none
  - One
  - all



شرکت خدمات اینفورماتیک  
informatics services corporation



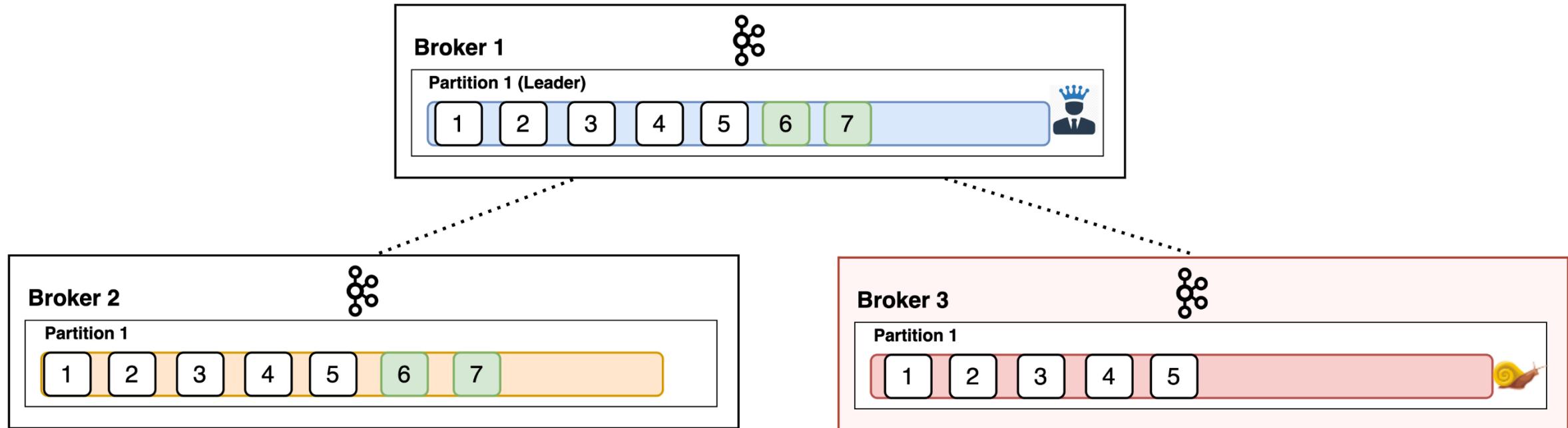
# Message Delivery and Durability – acks - replication



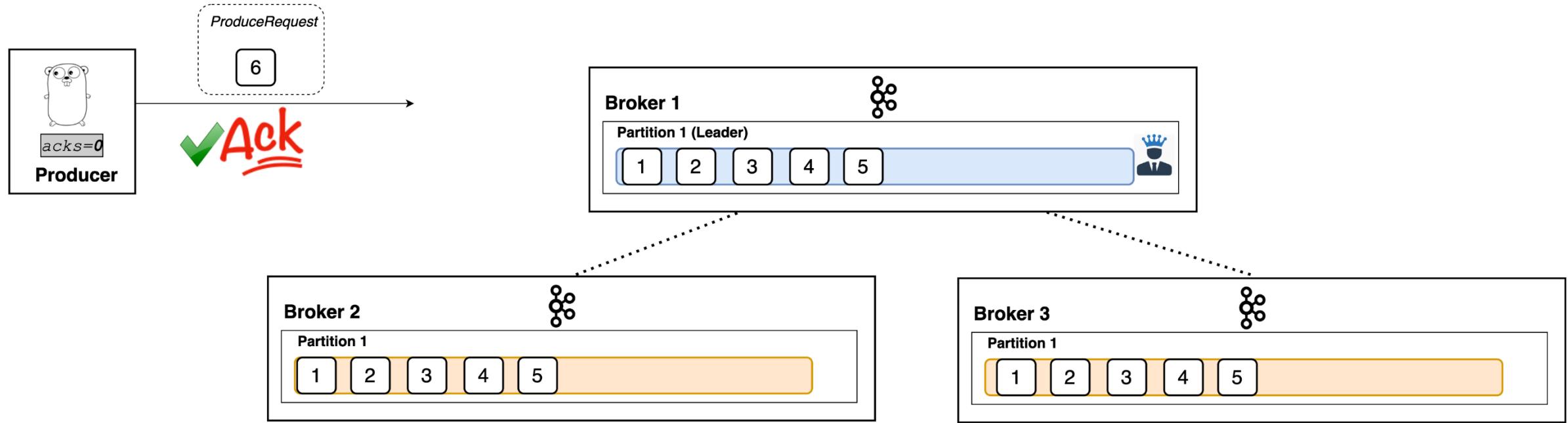
# Message Delivery and Durability – acks – isr

In Sync Replicas = [1, 2]

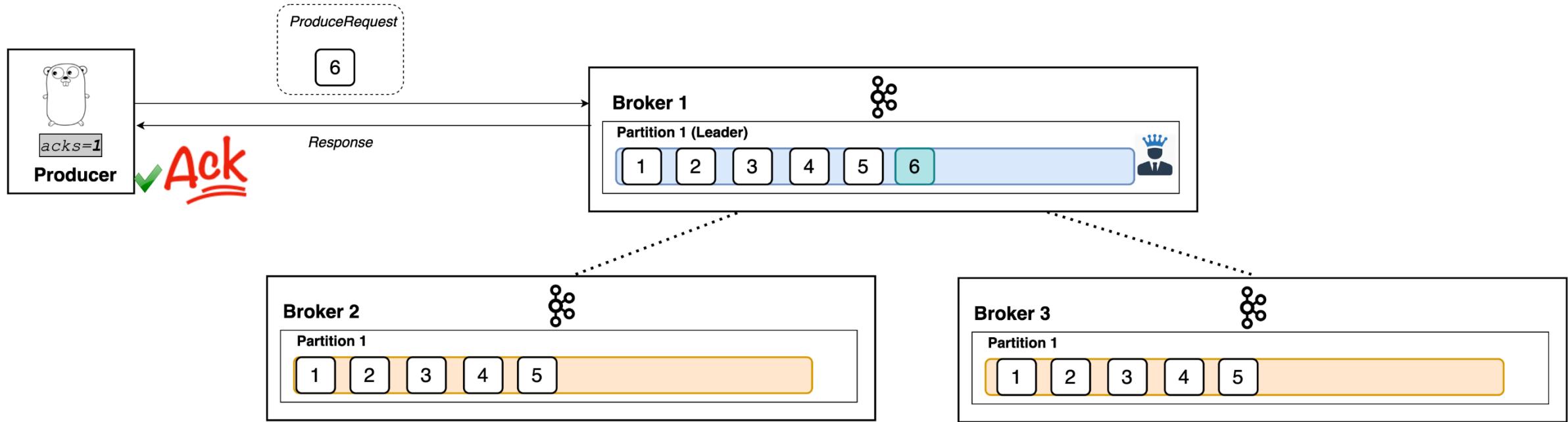
Out of Sync Replicas = [3]



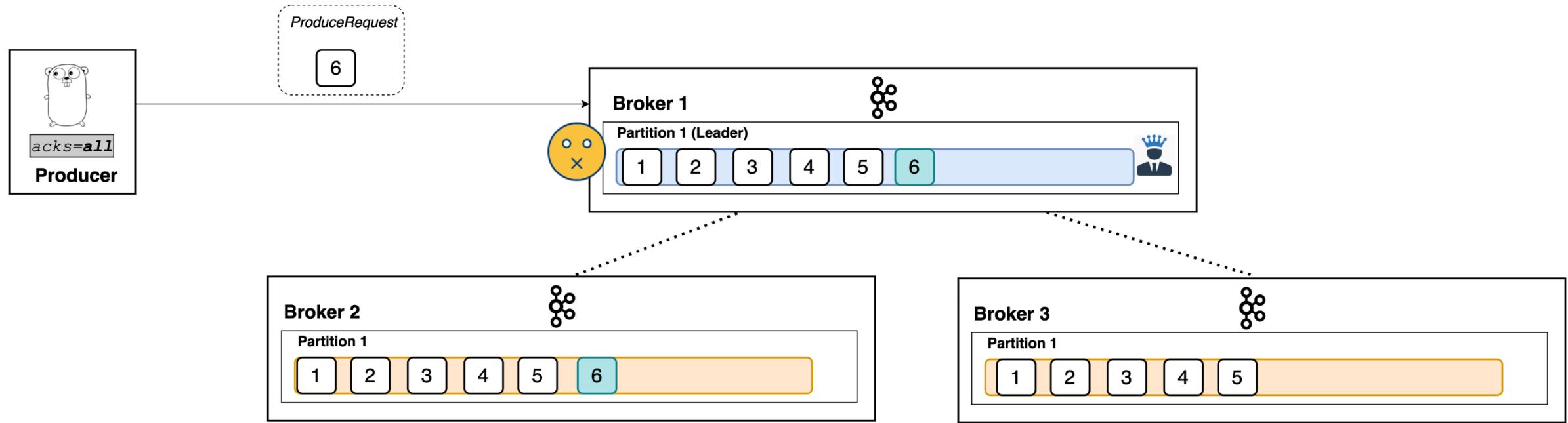
# Message Delivery and Durability – acks = 0



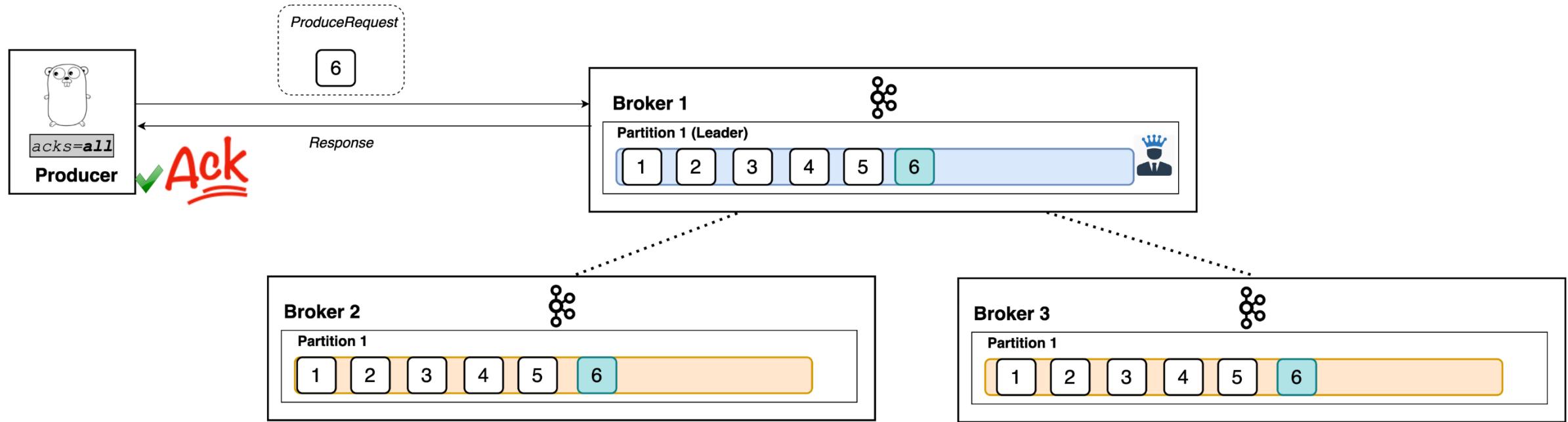
# Message Delivery and Durability – acks = 1



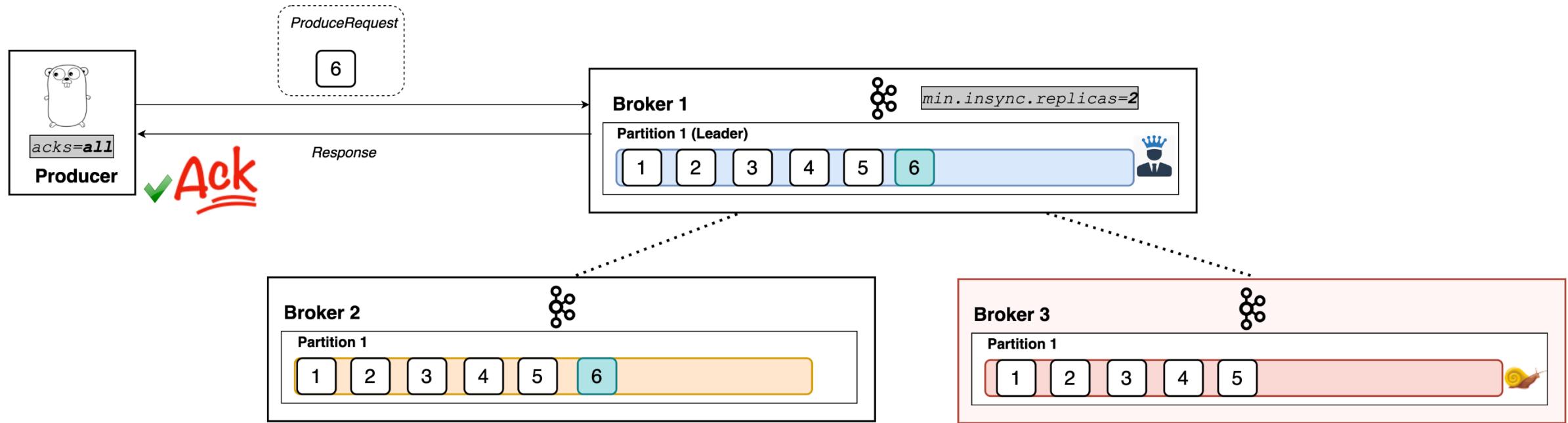
# Message Delivery and Durability – acks = all



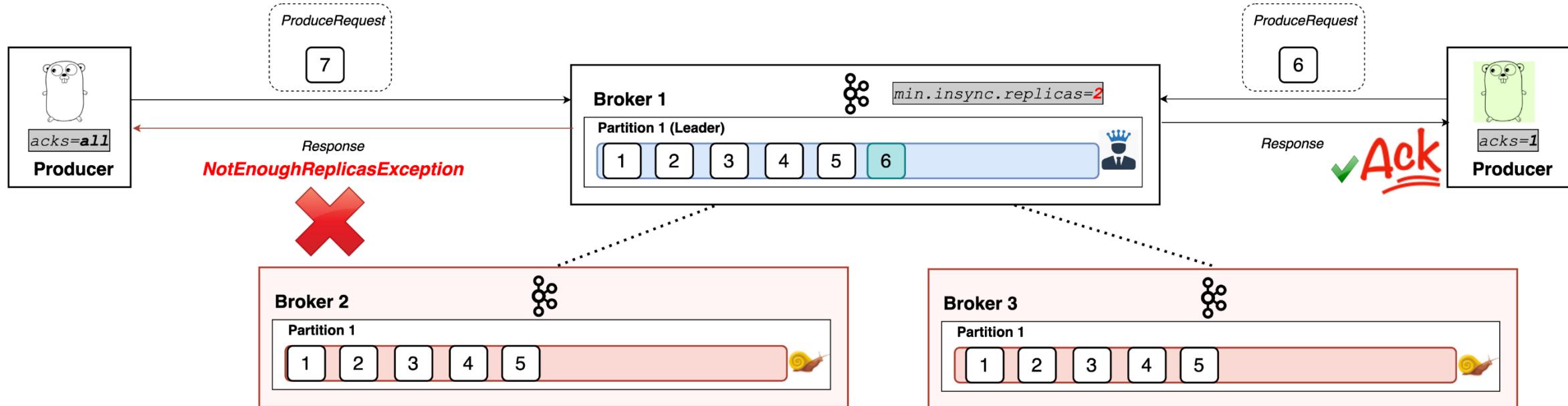
# Message Delivery and Durability – acks = all



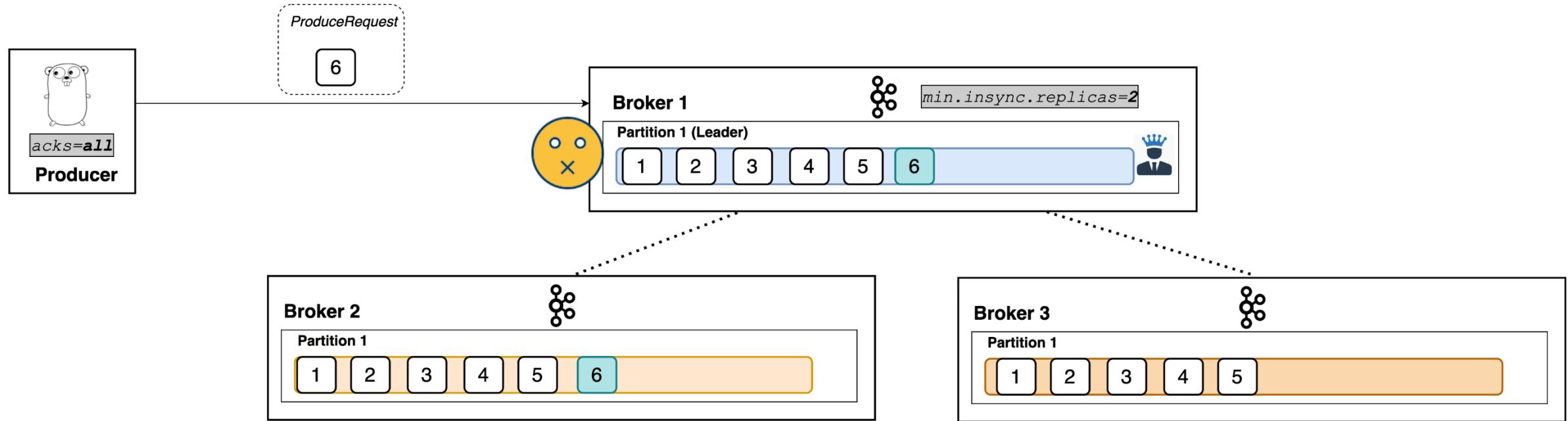
# Message Delivery and Durability – Minimum In-Sync Replica



# Message Delivery and Durability – Minimum In-Sync Replica



# Message Delivery and Durability – Minimum In-Sync Replica

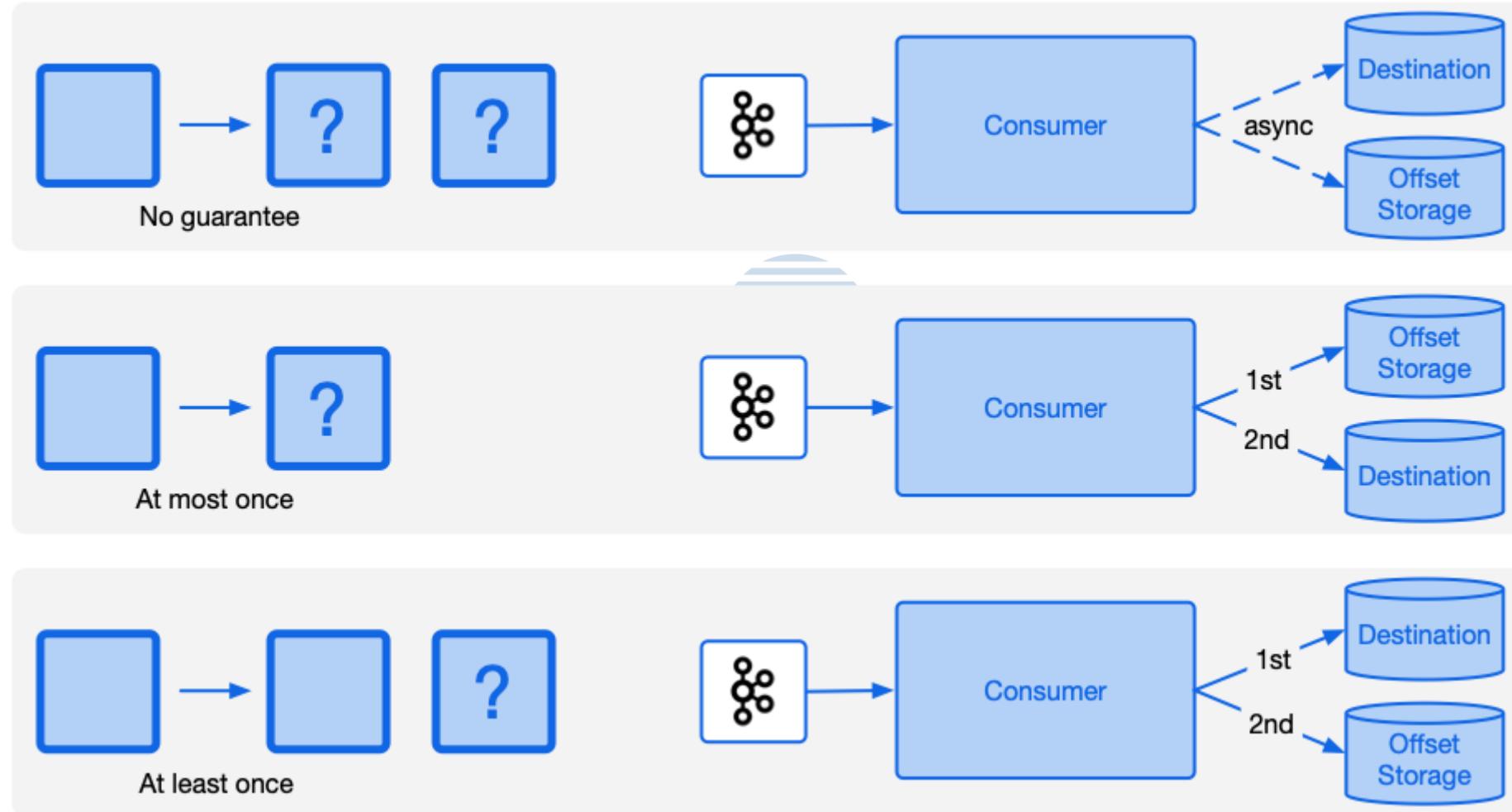


# Processing Guarantees – Issues

- How do we guarantee all messages are processed?
- How do we avoid or handle duplicate messages?



# Processing Guarantees – Concept





شرکت خدمات انفورماتیک  
informatics services corporation

## 5 - Consumers



# Consumer & Consumer Group

- Topic
- Producer - Tail
- Consumer – Own Place
- Scale - Partition
- Consumer Group
- Old - Zookeeper for Group Management
- New – Group Coordination Protocol

تیک  
infor

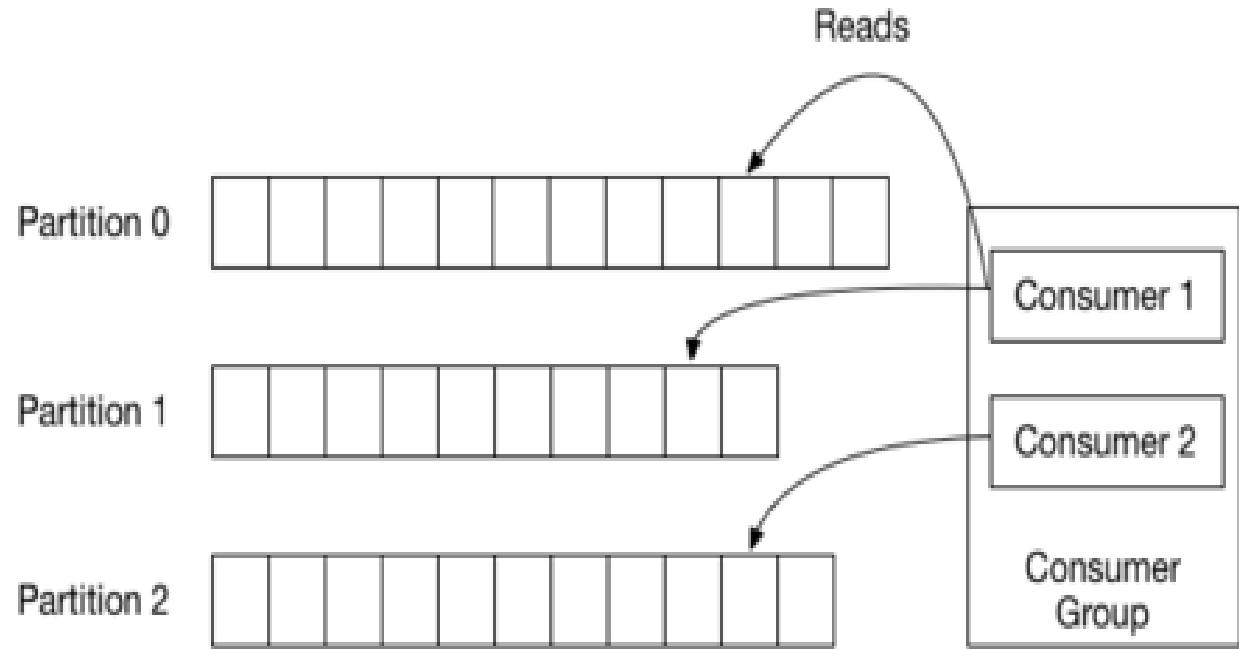
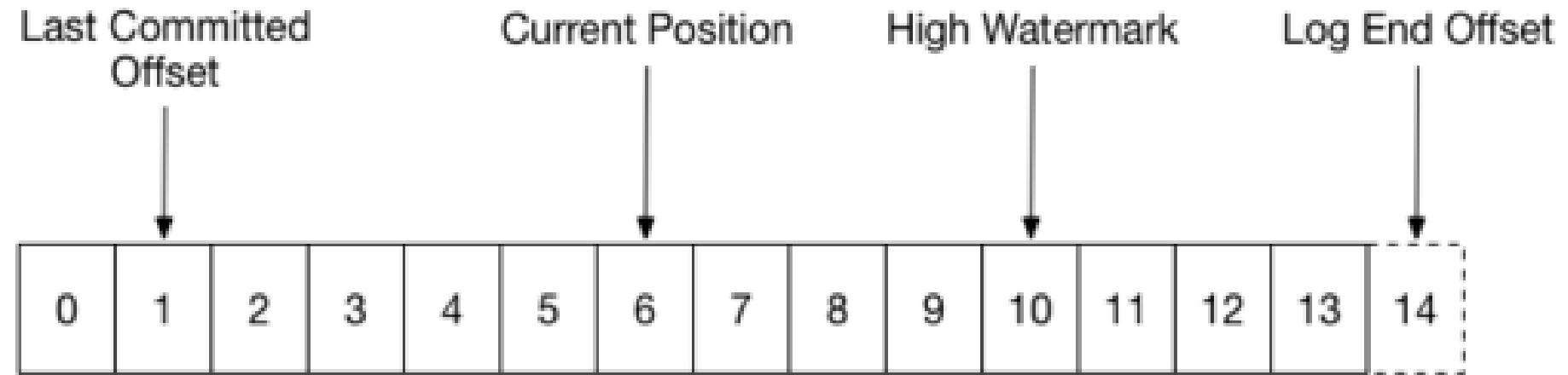


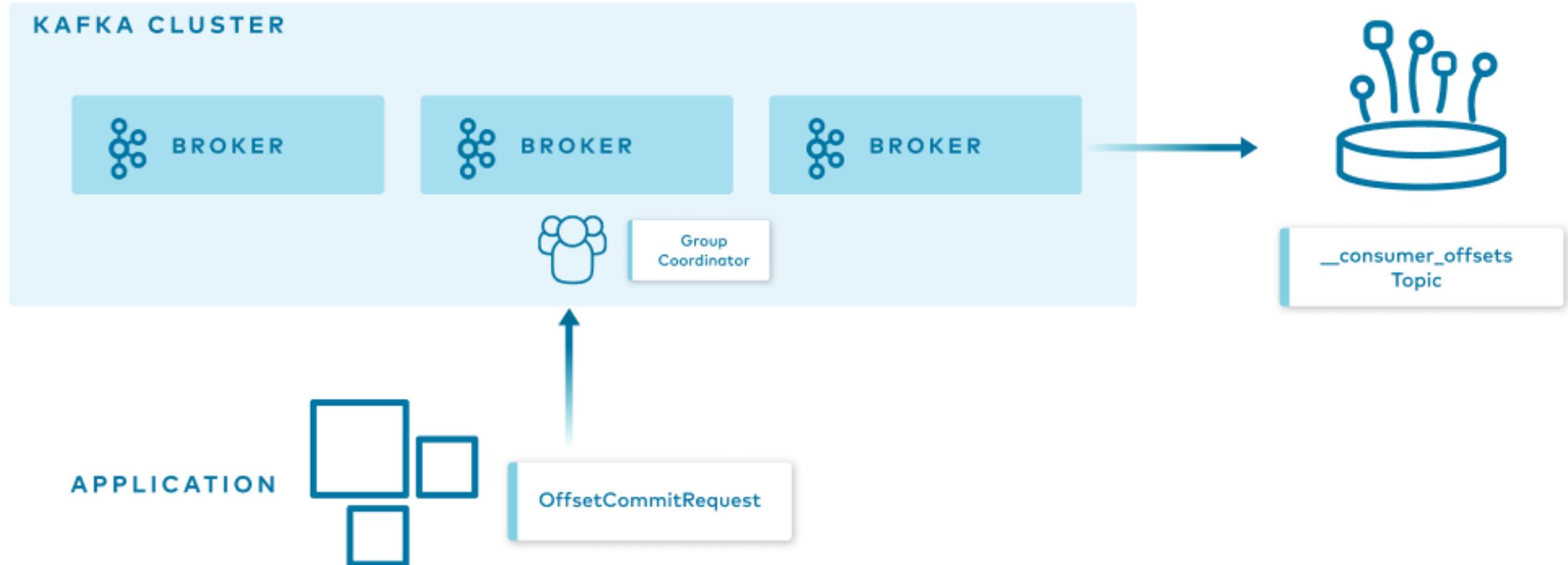
Figure 1: Consumer Group



# Consumer Position



# Consumer – Offset Storage



# Consumer – Describe using CLI

```
./kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group my-group

TOPIC      PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG    CONSUMER-ID                               HOST            CLIENT-ID
my-topic    0          2              4                2    consumer-1-029af89c-873c-4751-a720-cefd41a669d6  /127.0.0.1    consumer-1
my-topic    1          2              3                1    consumer-1-029af89c-873c-4751-a720-cefd41a669d6  /127.0.0.1    consumer-1
my-topic    2          2              3                1    consumer-2-42c1abd4-e3b2-425d-a8bb-e1ea49b29bb2  /127.0.0.1    consumer-2
```



# Consumer – Lag

- Key Performance Indicator - KPI
- Delta between last committed to last produces
- Keep lag to minimum



- Kafka Persistence is based on retention

- Lag persists

- Lose data at some point in time



# Consumer Lag Explore

- ❖ Kafka Lag Exporter
  - ❖ native Kubernetes support
  - ❖ contained time-based lag monitoring

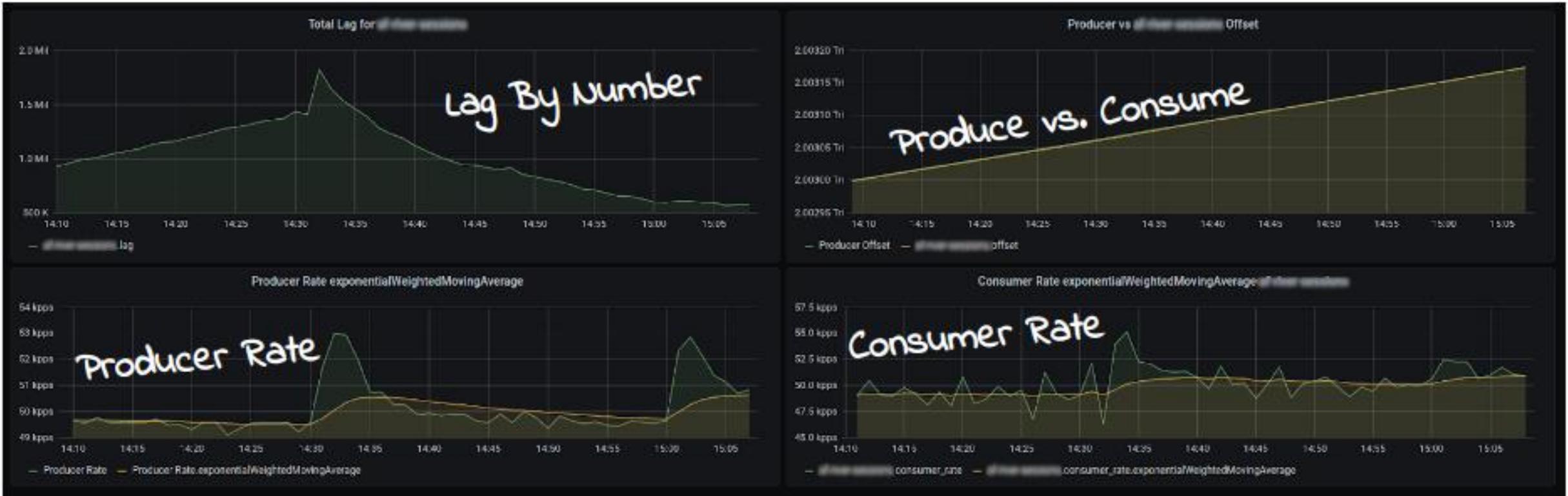
- ❖ Burrow
  - ❖ Burrow is an active LinkedIn project
  - ❖ Has an active community



- ❖ Remora
  - ❖ Was created after Zalando spent some time using Burrow.
  - ❖ Has Datadog and CloudWatch integration



# Burrow dashboards



# More Important – Time-Based

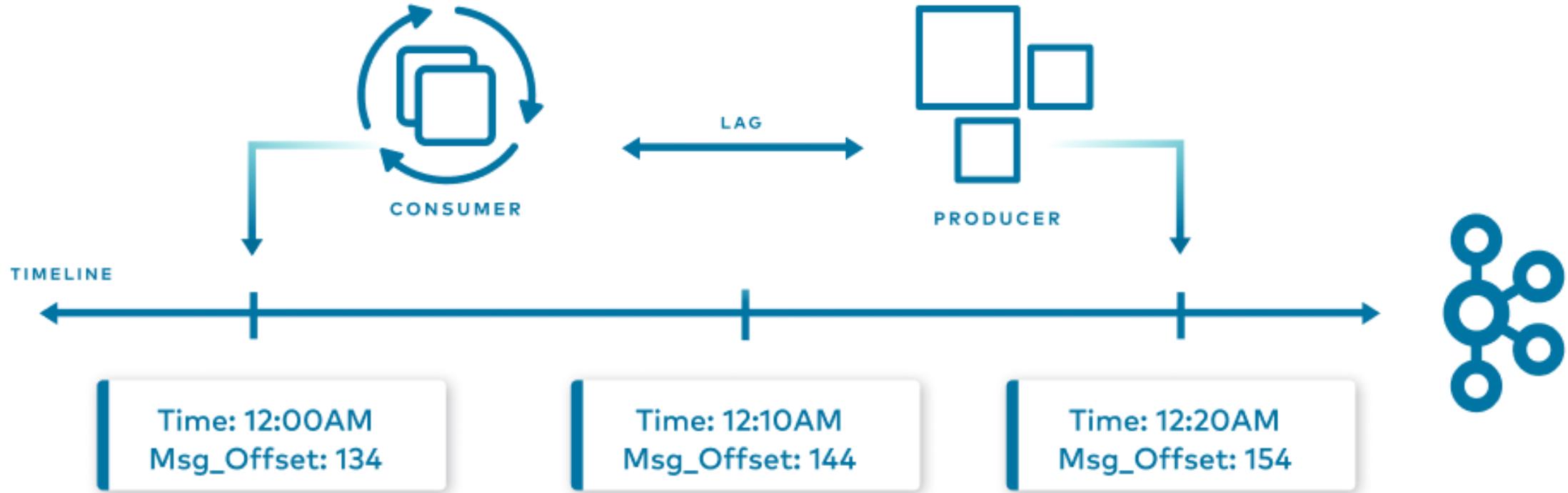


Diff (Last\_Consumed, Last\_Produced)

Producer Rate

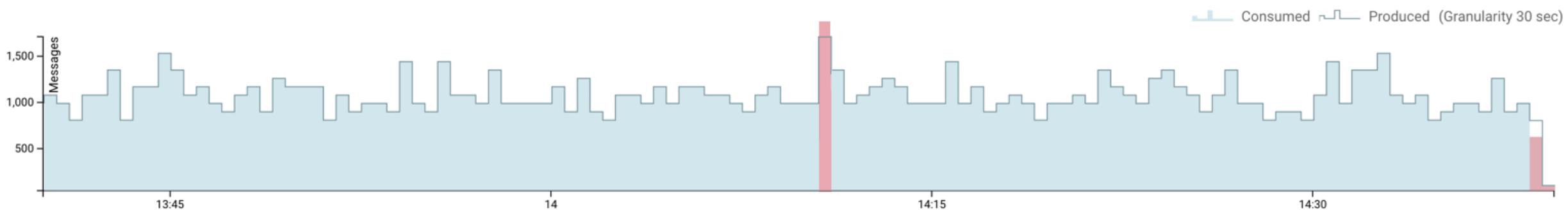


# Consumer Lag – Time Lag - Visual



# Consumer - Over and Under Consumption

Messages Consumed



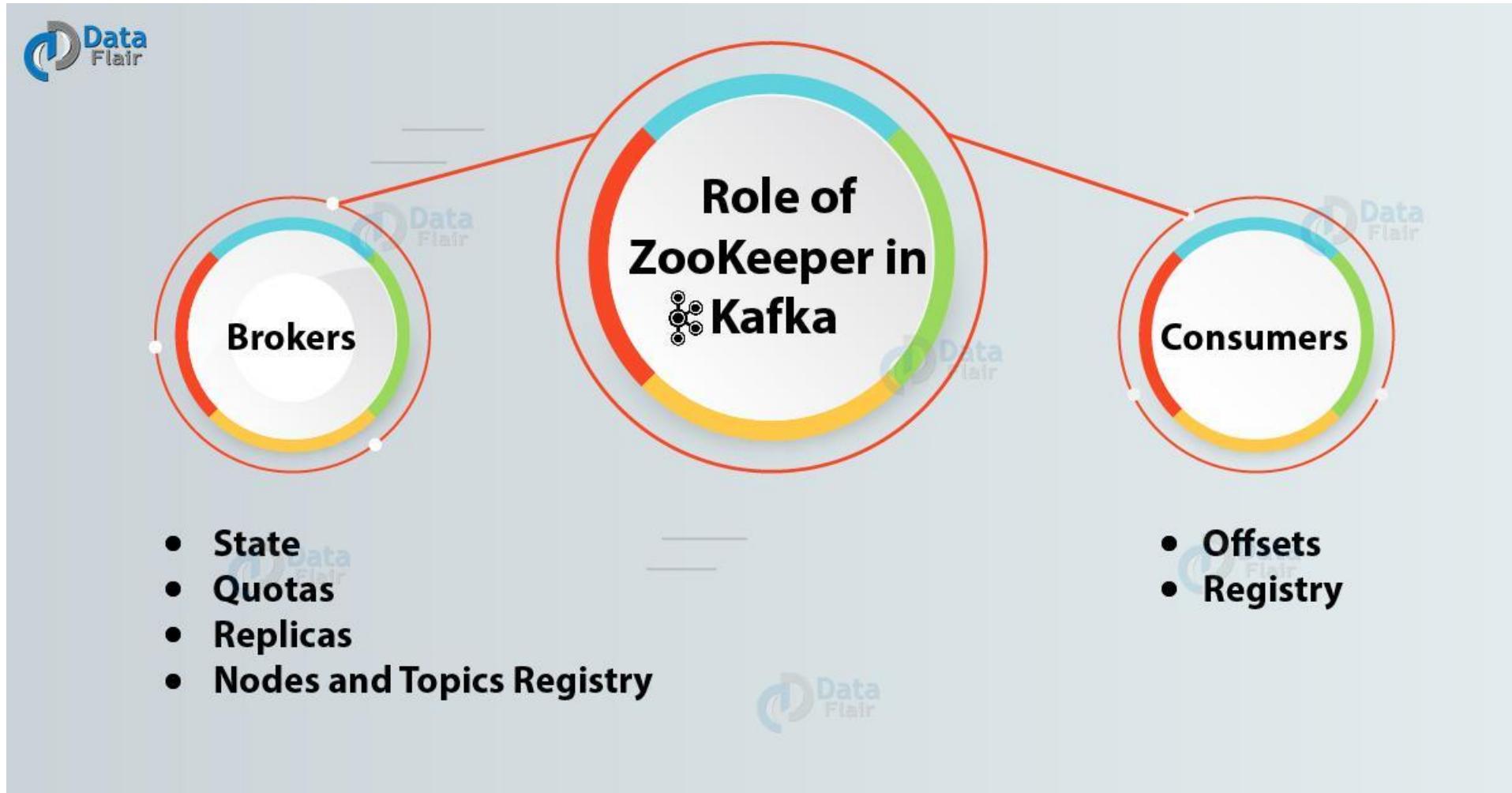


شرکت خدمات انفورماتیک  
informatics services corporation

## 6 - Configuration and Dependency



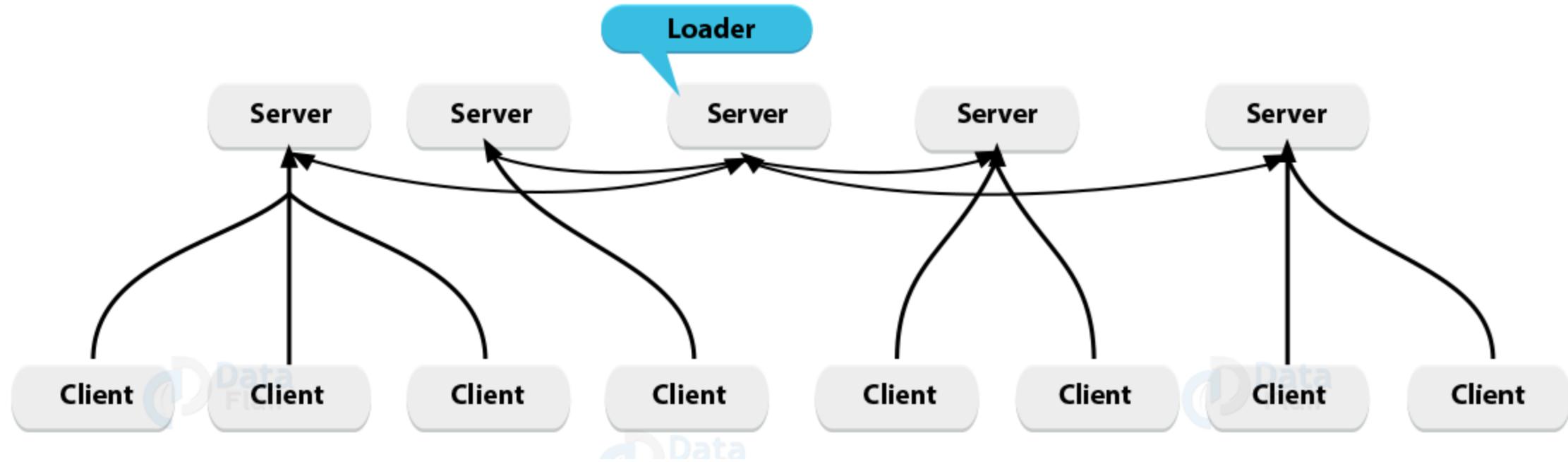
# Dependency - Zookeeper



# Zookeeper Architecture



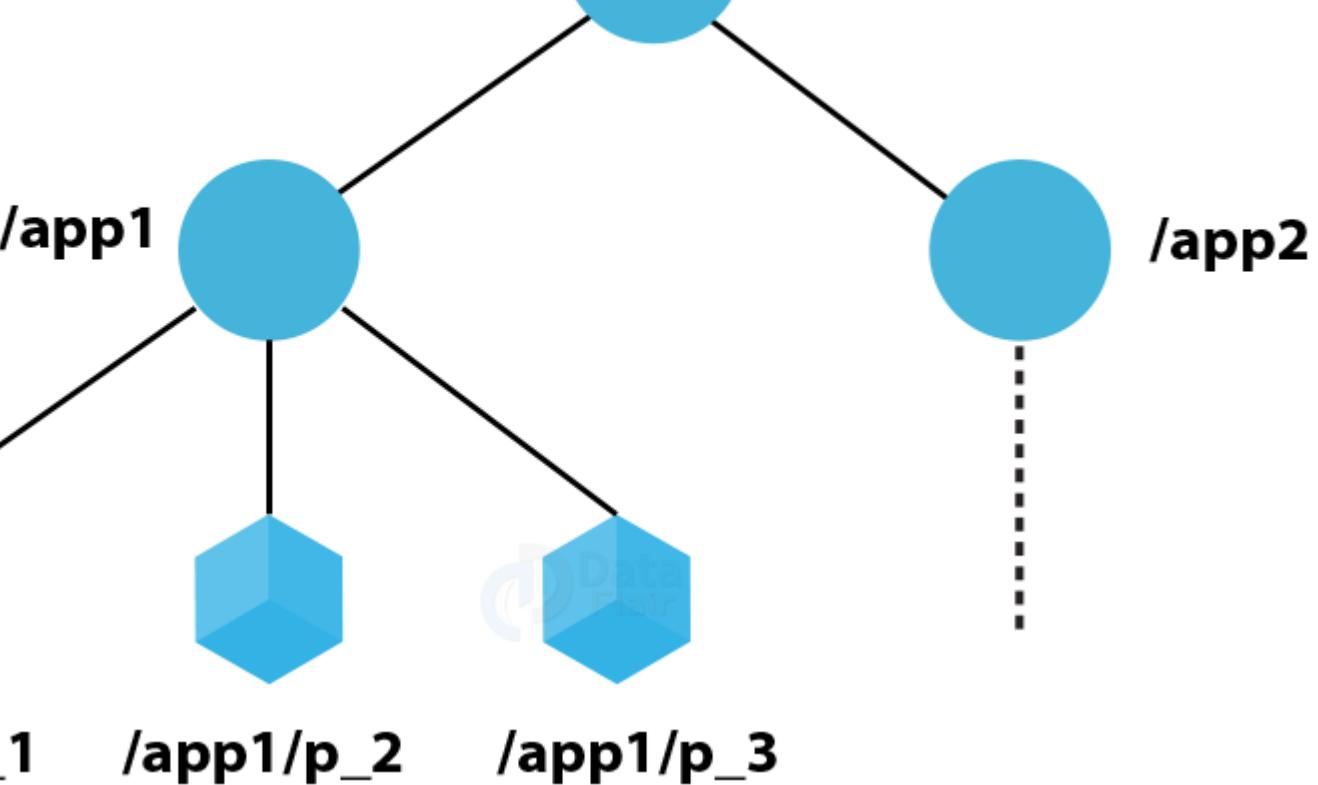
# Zookeeper – Client / Server Architecture

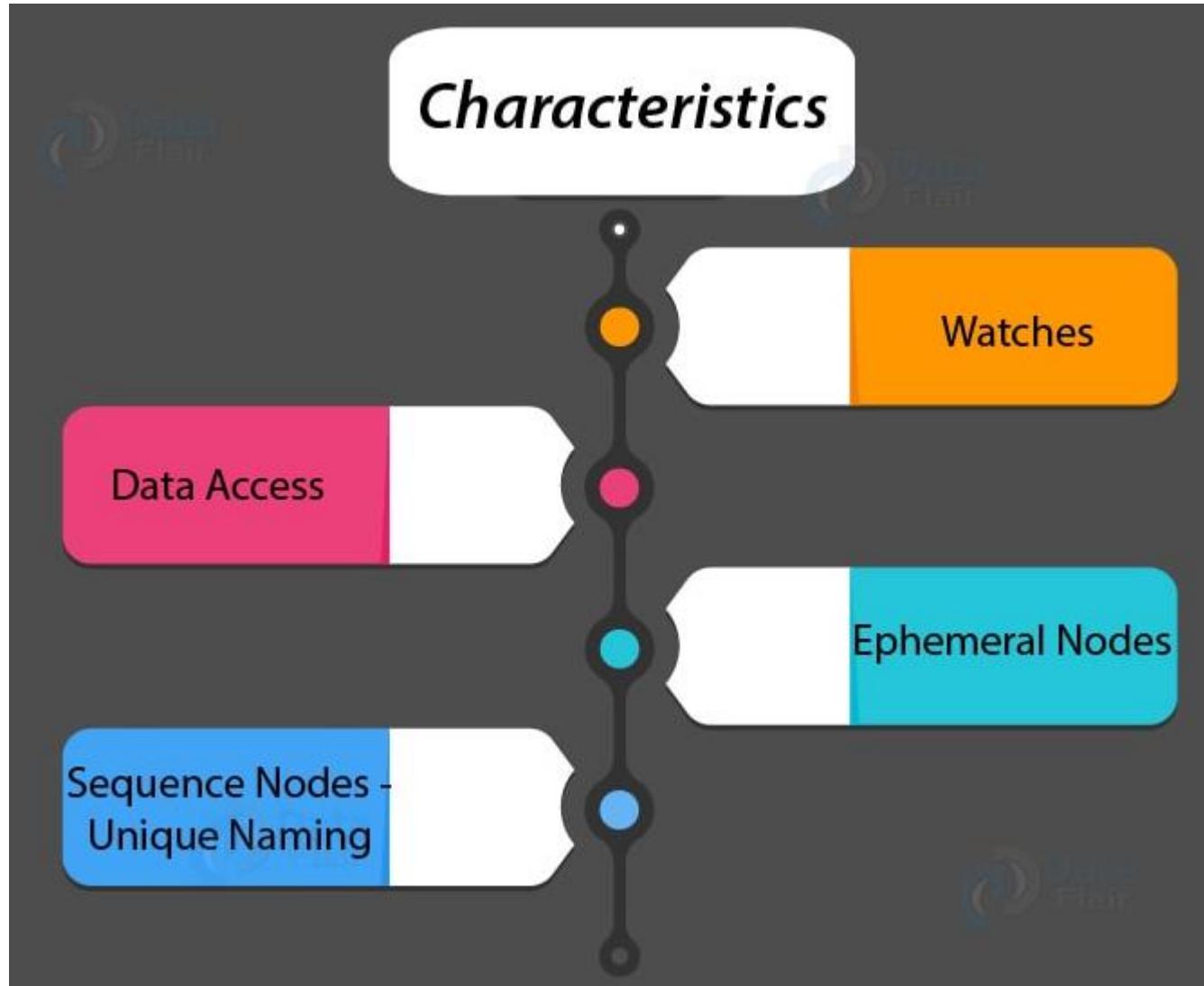


# Design Goals of Zookeeper Architecture



# Data Model in ZooKeeper





# Zookeeper – Versions

```
[zk: localhost:2181(CONNECTED) 16] get /newznode
new znode
cZxid = 0x10
ctime = Wed Jul 02 23:48:15 EDT 2014
mZxid = 0x4d
mtime = Sun Jul 06 23:33:53 EDT 2014
pZxid = 0x10
cversion = 0
dataVersion = 5 ←
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 9
numChildren = 0
[zk: localhost:2181(CONNECTED) 17]
```

```
[zk: localhost:2181(CONNECTED) 18] get /newznode
new znode
cZxid = 0x10
ctime = Wed Jul 02 23:48:15 EDT 2014
mZxid = 0x56
mtime = Sun Jul 06 23:47:01 EDT 2014
pZxid = 0x10
cversion = 0
dataVersion = 6 ←
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 9
numChildren = 0
[zk: localhost:2181(CONNECTED) 19]
```



```
26      zkc.connect("localhost");
27
28      ZooKeeper zk = zkc.getZooKeeper();
29      Stat stat = zk.exists("/newznode", true);
30      zk.setData("/newznode", "new znode".getBytes(), 8);
```

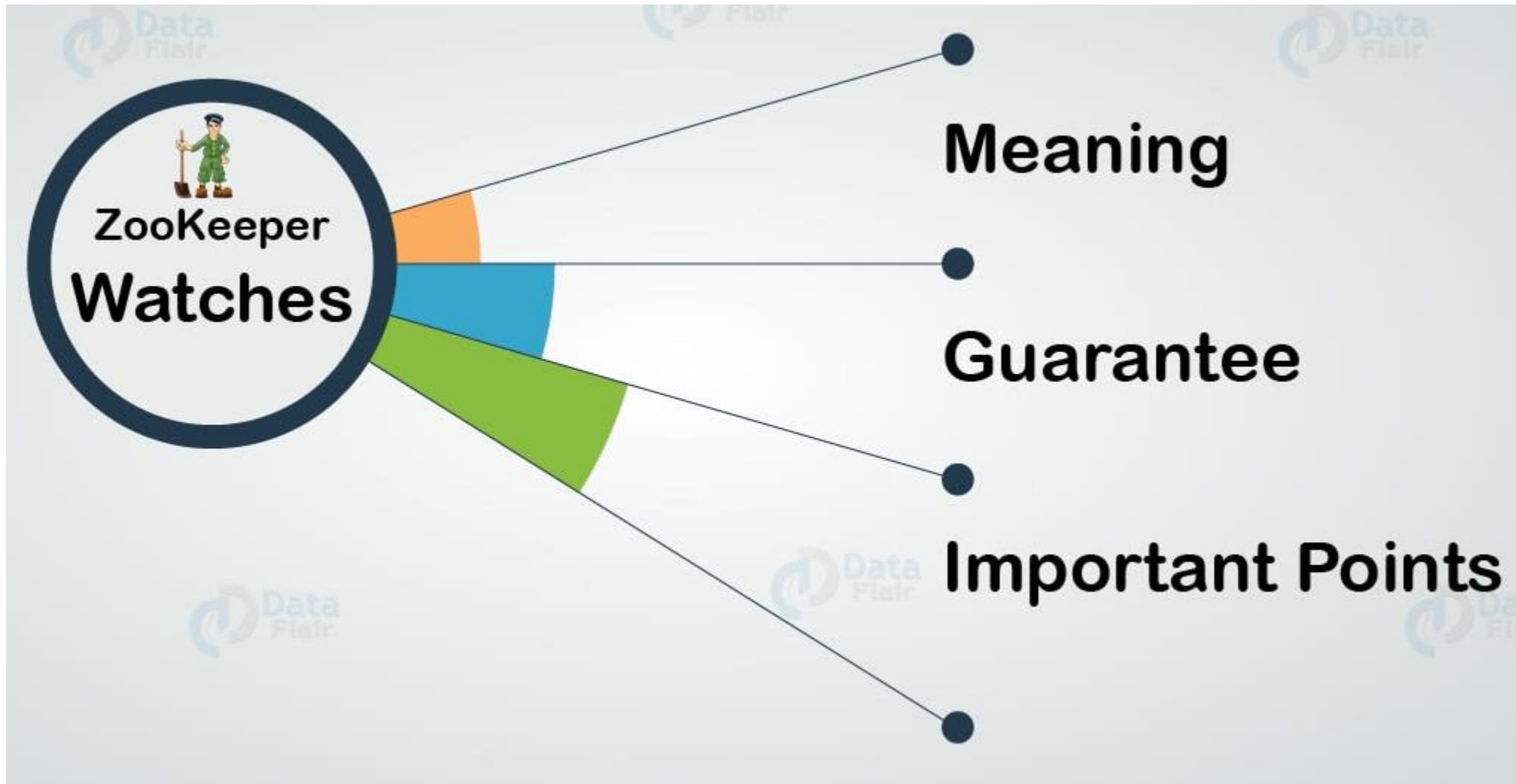
Markers Properties Data Source Explorer Snippets Console

<terminated> App [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Jul 6, 2014, 11:51:50 PM)

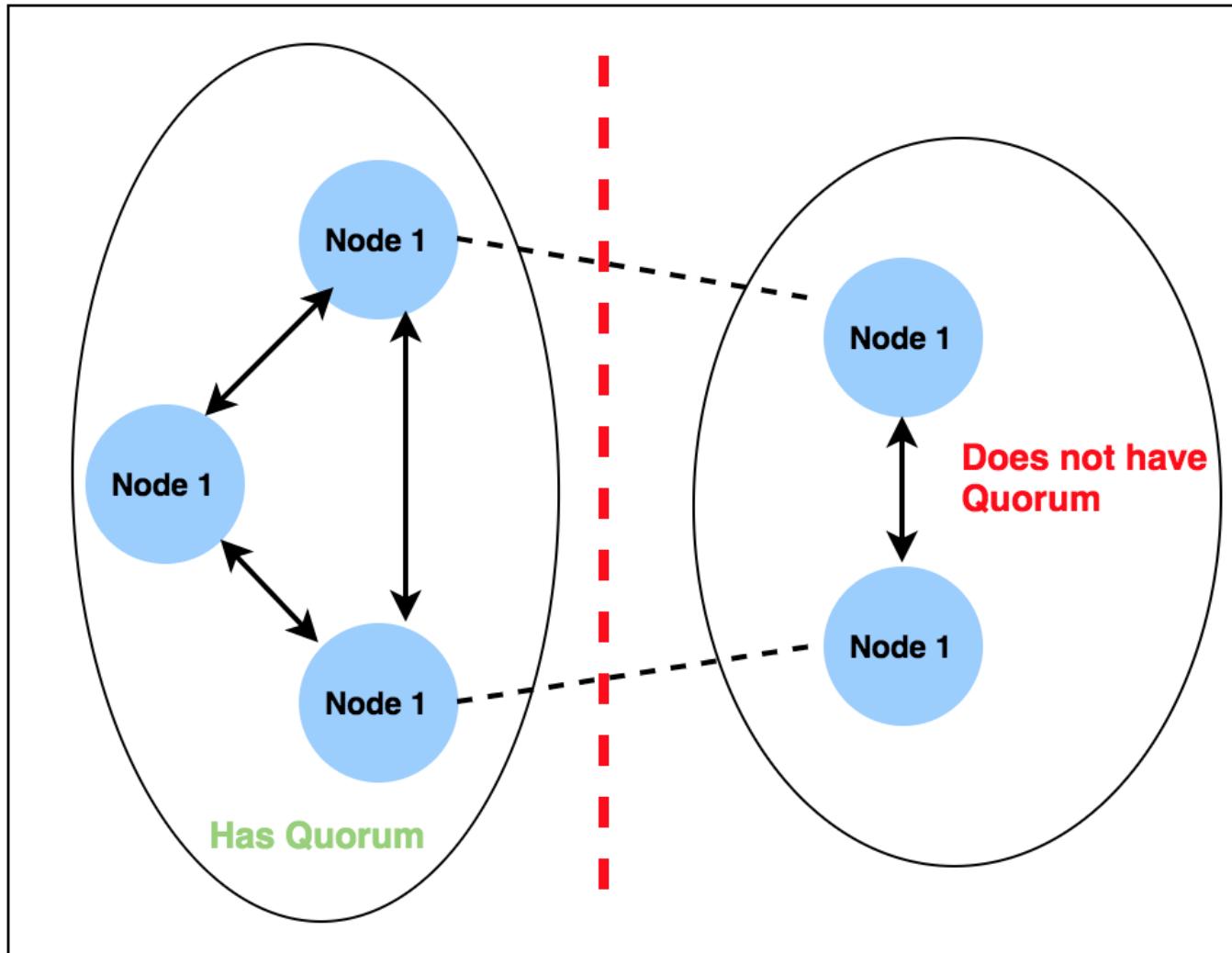
log4j:WARN See <http://logging.apache.org/log4j/1.2/faq.html#noconfig> for more info.

Exception in thread "main" org.apache.zookeeper.KeeperException\$BadVersionException: KeeperErrorCode = BadVersion for /newznode  
at org.apache.zookeeper.KeeperException.create(KeeperException.java:115)  
at org.apache.zookeeper.KeeperException.create(KeeperException.java:51)  
at org.apache.zookeeper.ZooKeeper.setData(ZooKeeper.java:1270)  
at com.zook.app.App.main(App.java:29)

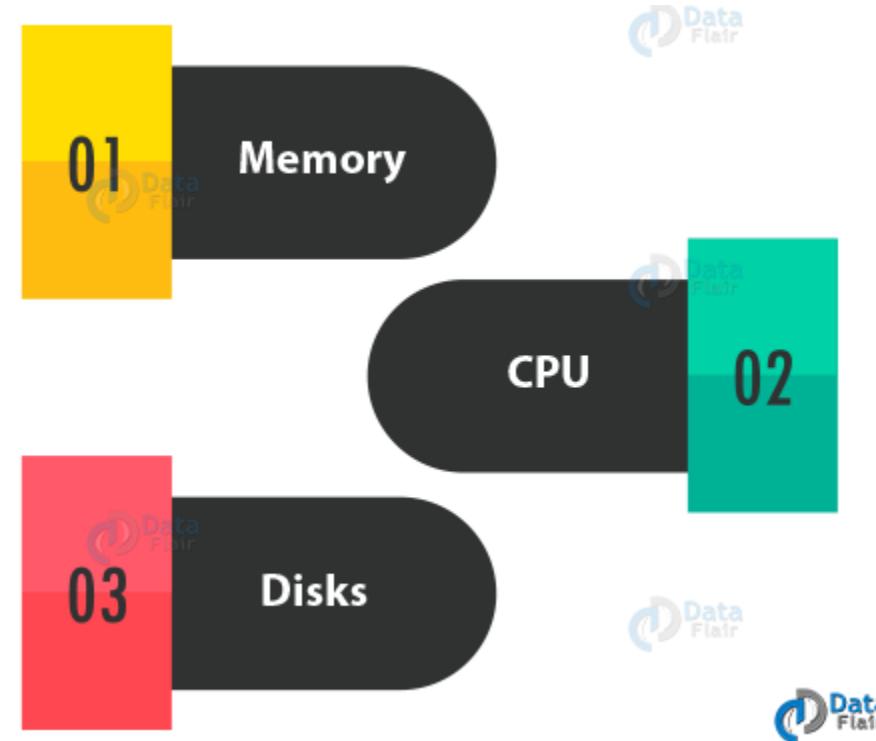




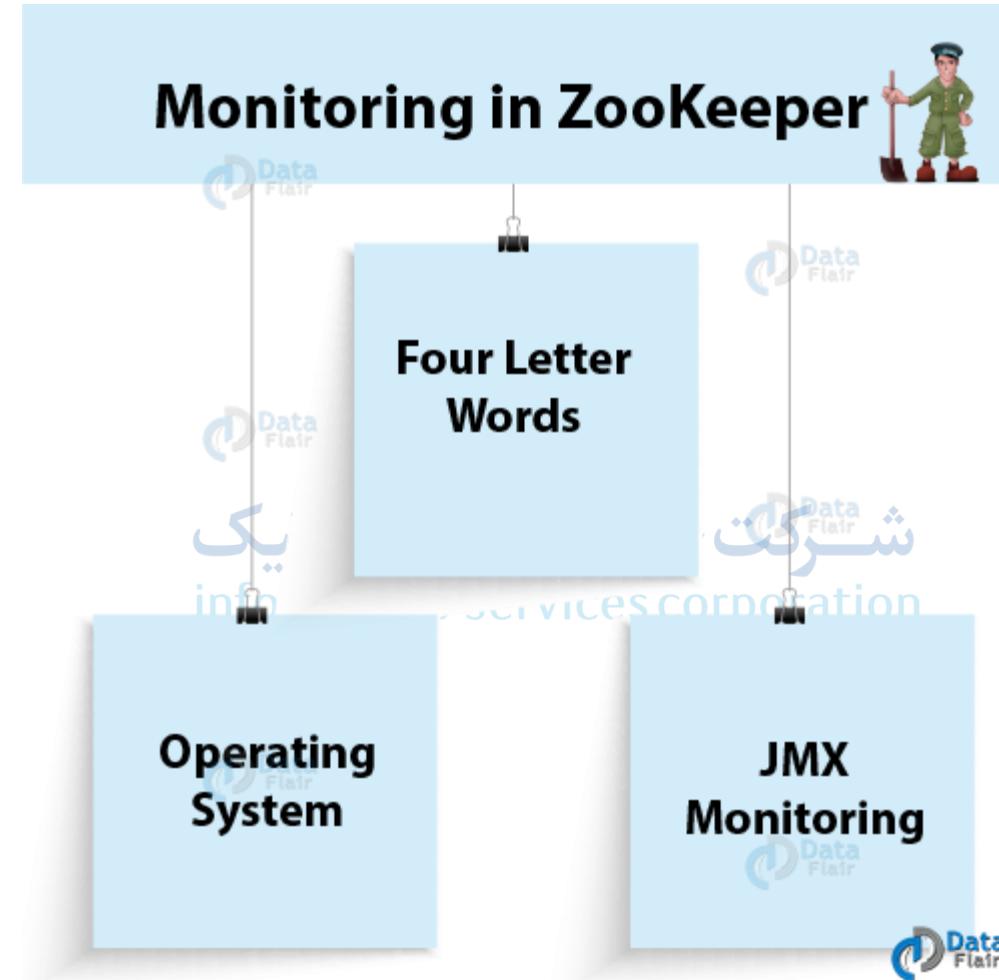
# Zookeeper - Quorums



# Hardware of Zookeeper



# Zookeeper - Monitoring



# Kafka Requirement



شرکت خ  
corporation



# Kafka – Important Configuration

## Level

Operation System – OS

## Important Configuration

- ❖ File descriptor limits
- ❖ Max socket buffer
- ❖ Max number of memory map

Producer

- ❖ acks
- ❖ compression
- ❖ batch size

Consumer

fetch size

Java

8 and 11



# Kafka – Linkedin busiest Cluster

## Cluster Status

- 60 brokers
- 50k partitions (replication factor 2)
- 800k messages/sec in
- 300 MB/sec inbound
- 1 GB/sec outbound

## JVM Argument

- -XX:+UseG1GC
- -XX:MaxGCPauseMillis=20
- -XX:InitiatingHeapOccupancyPercent=35
- -XX:G1HeapRegionSize=16M
- -XX:MinMetaspaceFreeRatio=50
- -XX:MaxMetaspaceFreeRatio=80
- -XX:+ExplicitGCIInvokesConcurrent



## GC Result

- 90% GC pause time of about 21ms
- less than 1% young GC per second



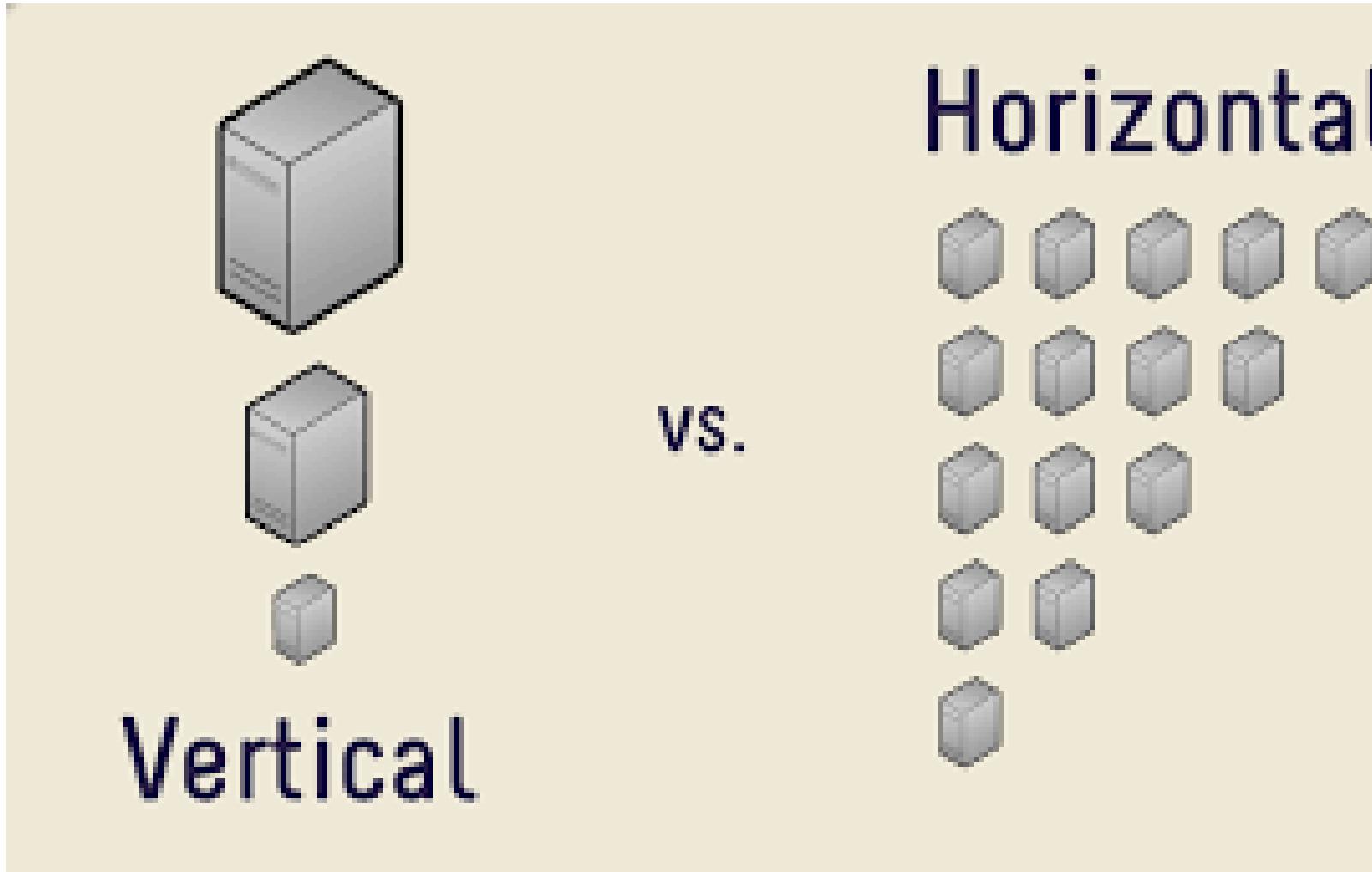


شرکت خدمات انفورماتیک  
informatics services corporation

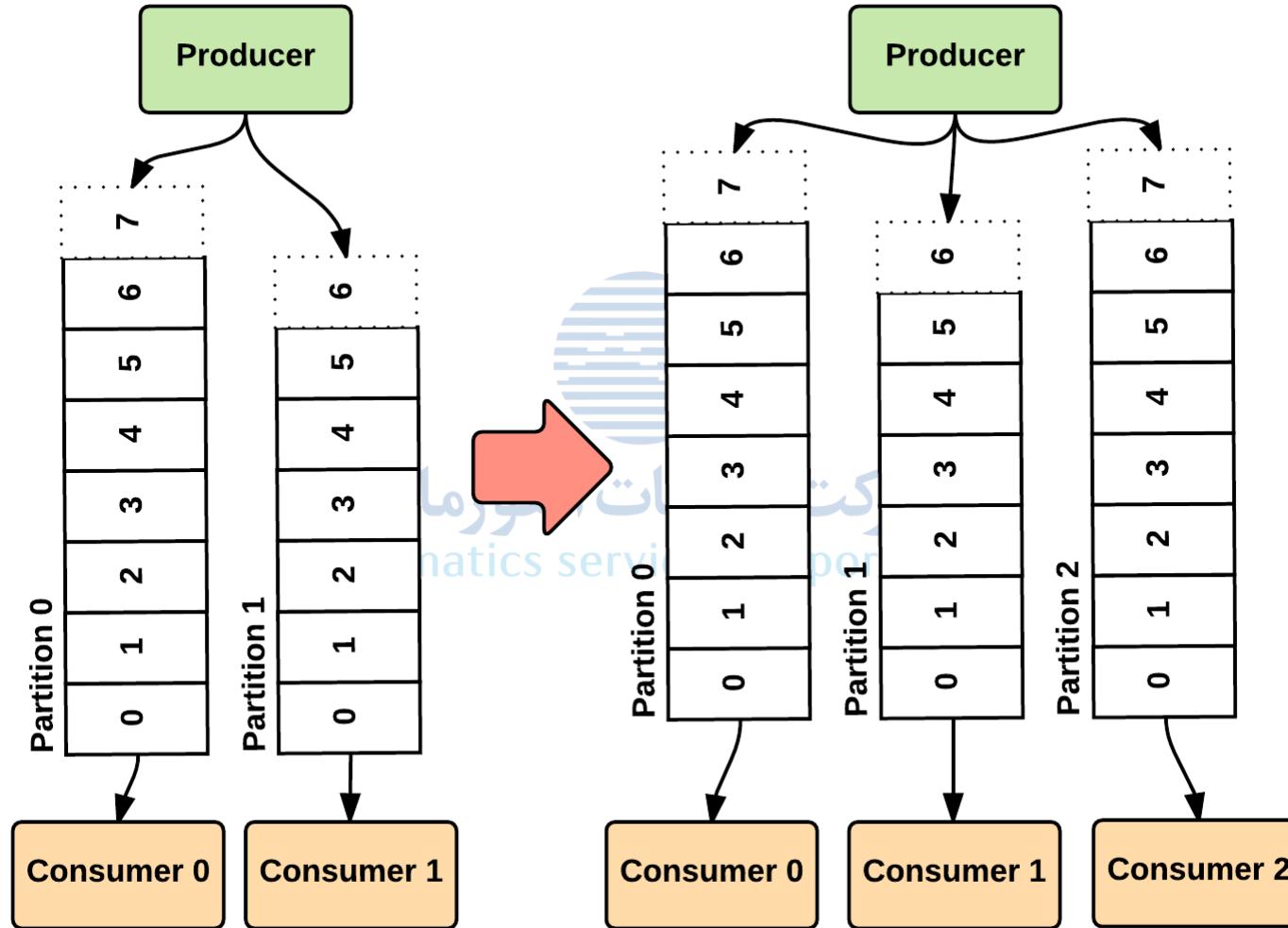
## 7 - Clustering – Scalability, Fault Tolerant and High Availability



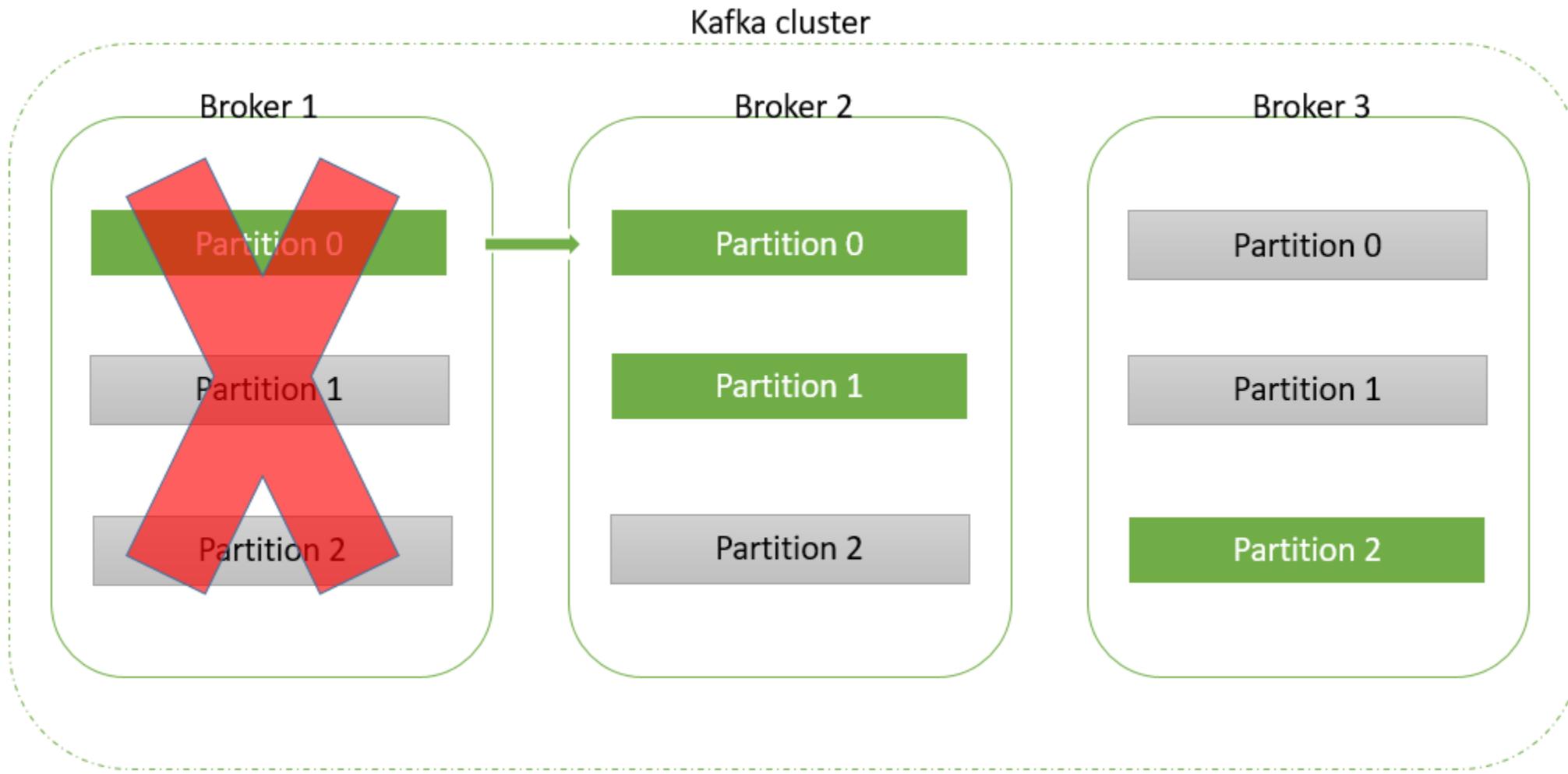
# Scaling – Vertical vs Horizontal



# Scale-out architecture with Kafka



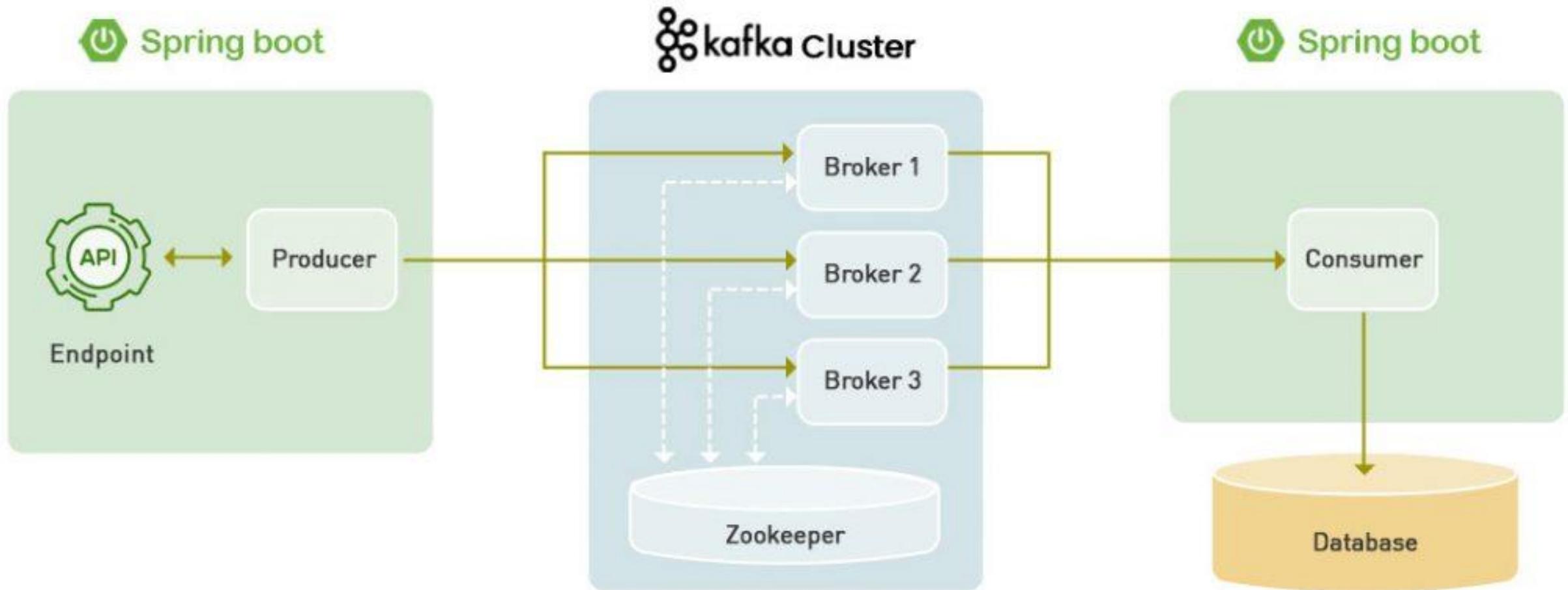
# Fault Tolerance



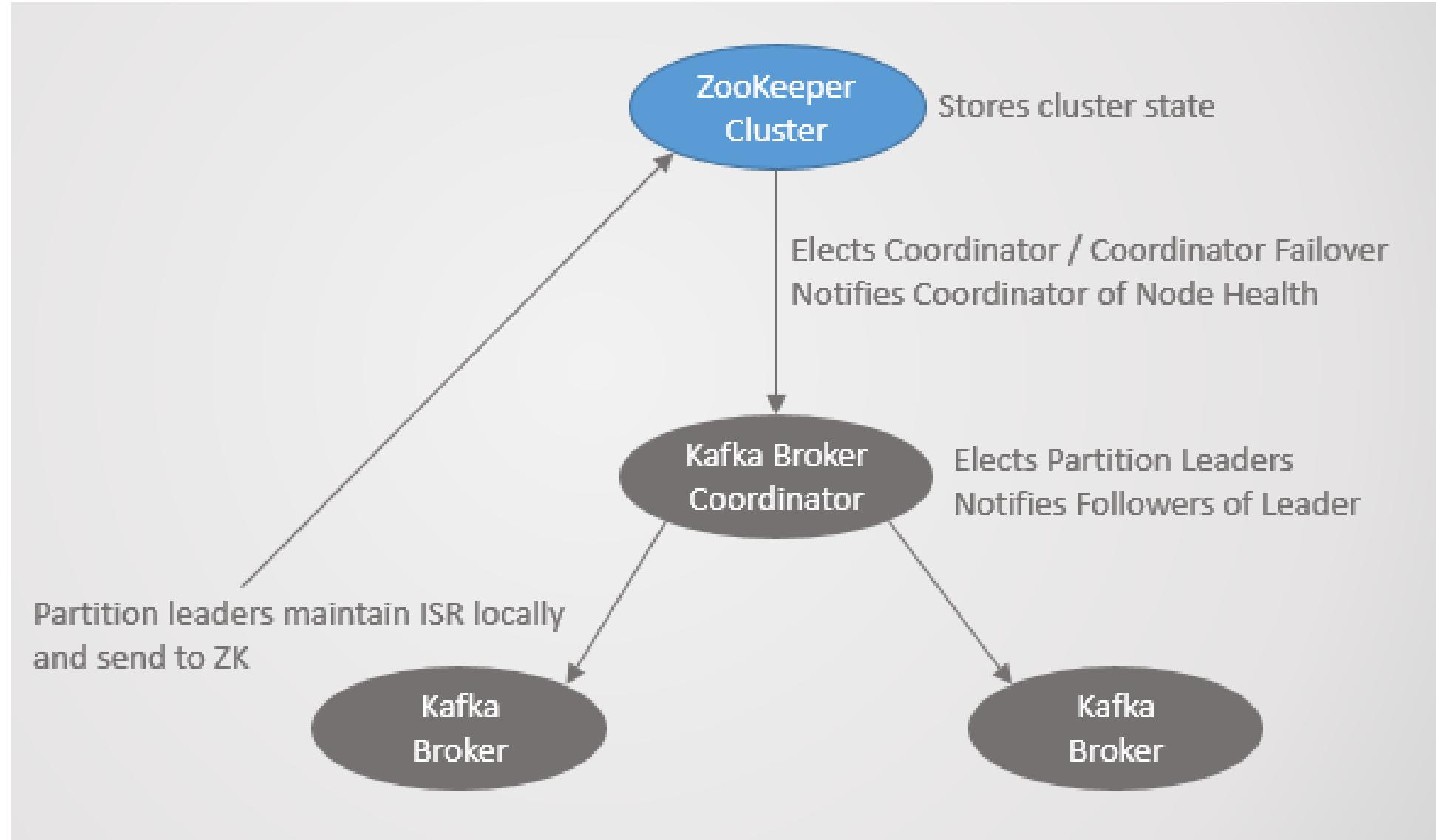
# Availability



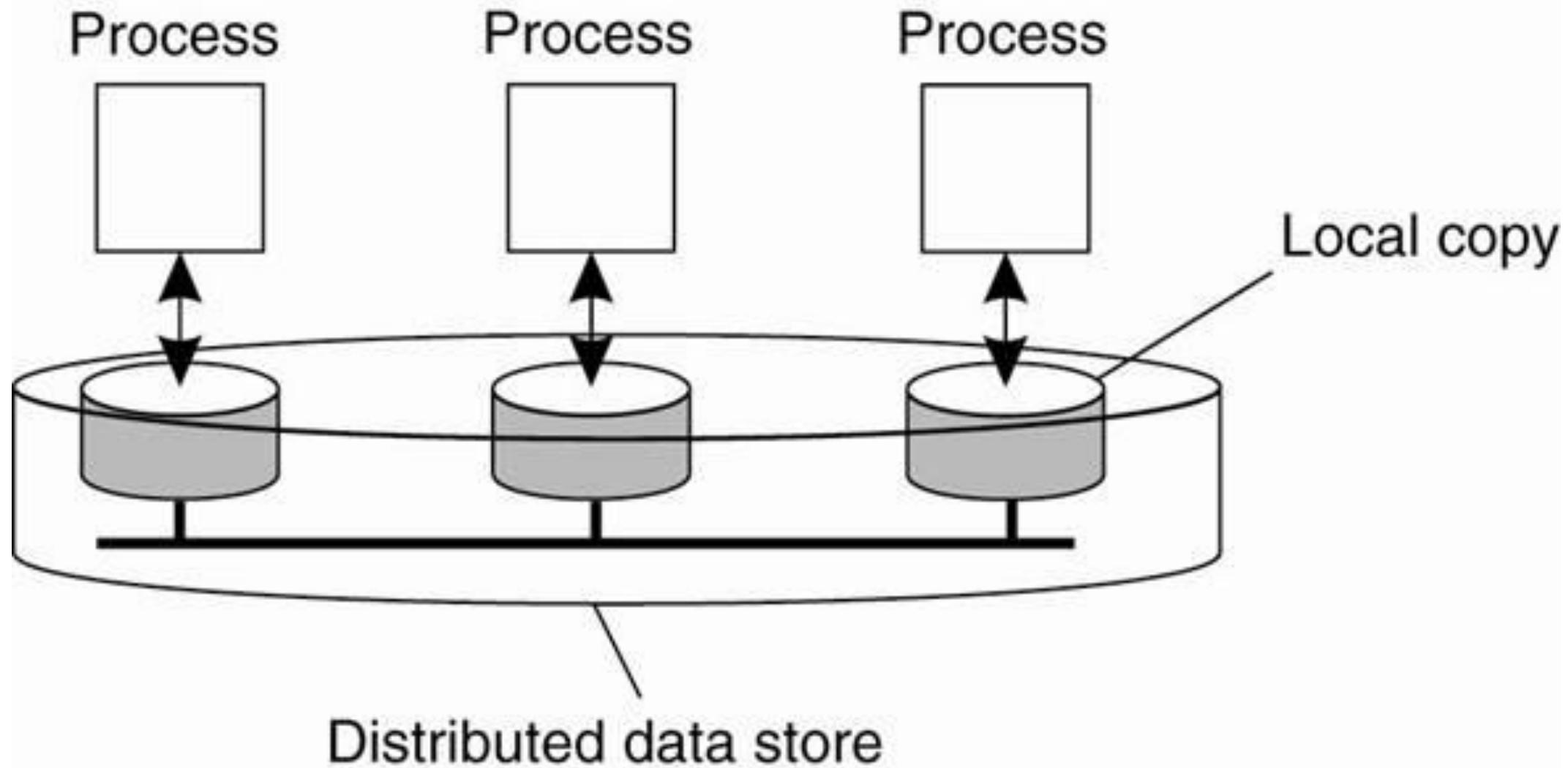
# Ensuring Client Connectivity



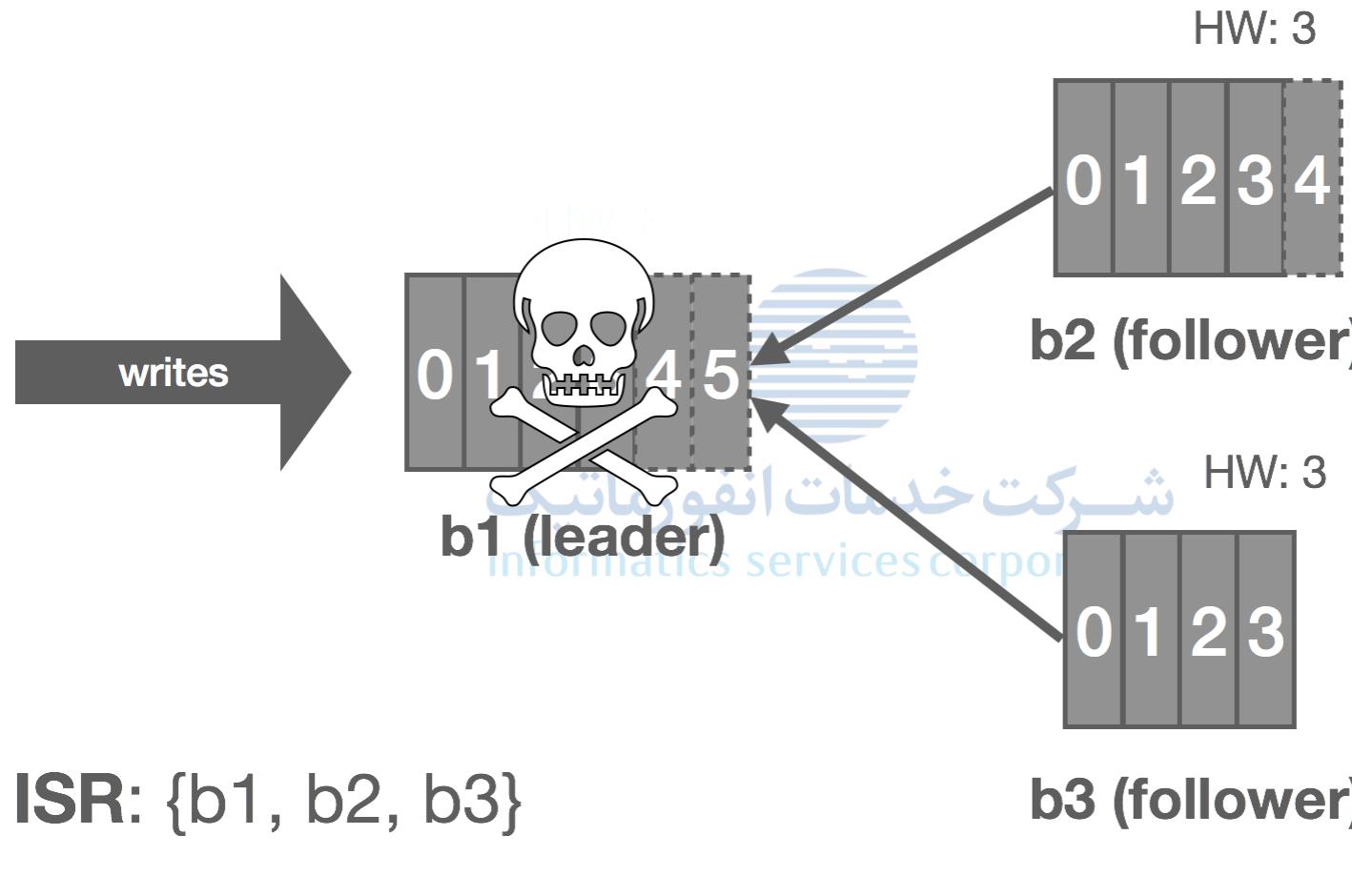
# The Kafka Consensus Architecture



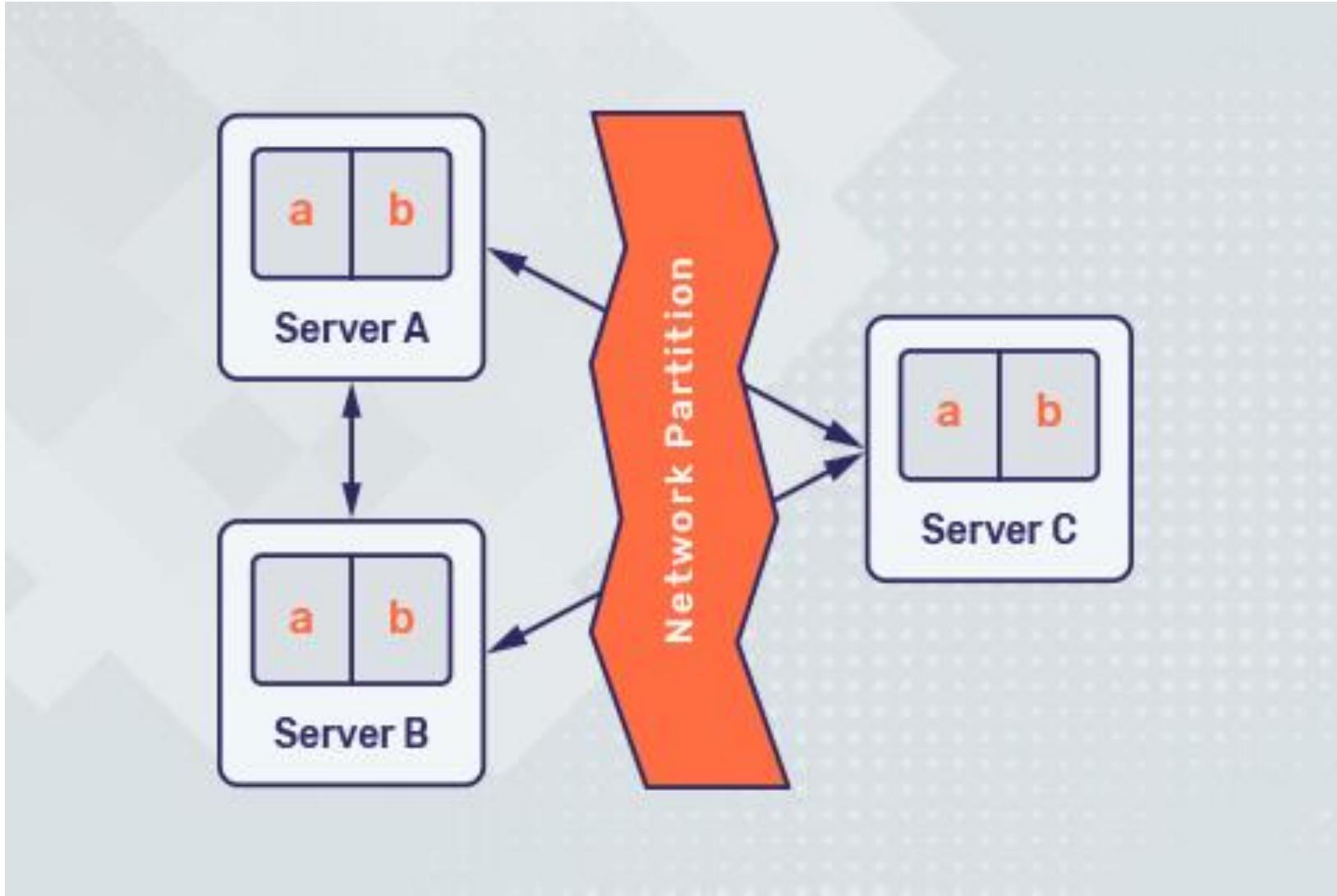
# The Replication Protocol



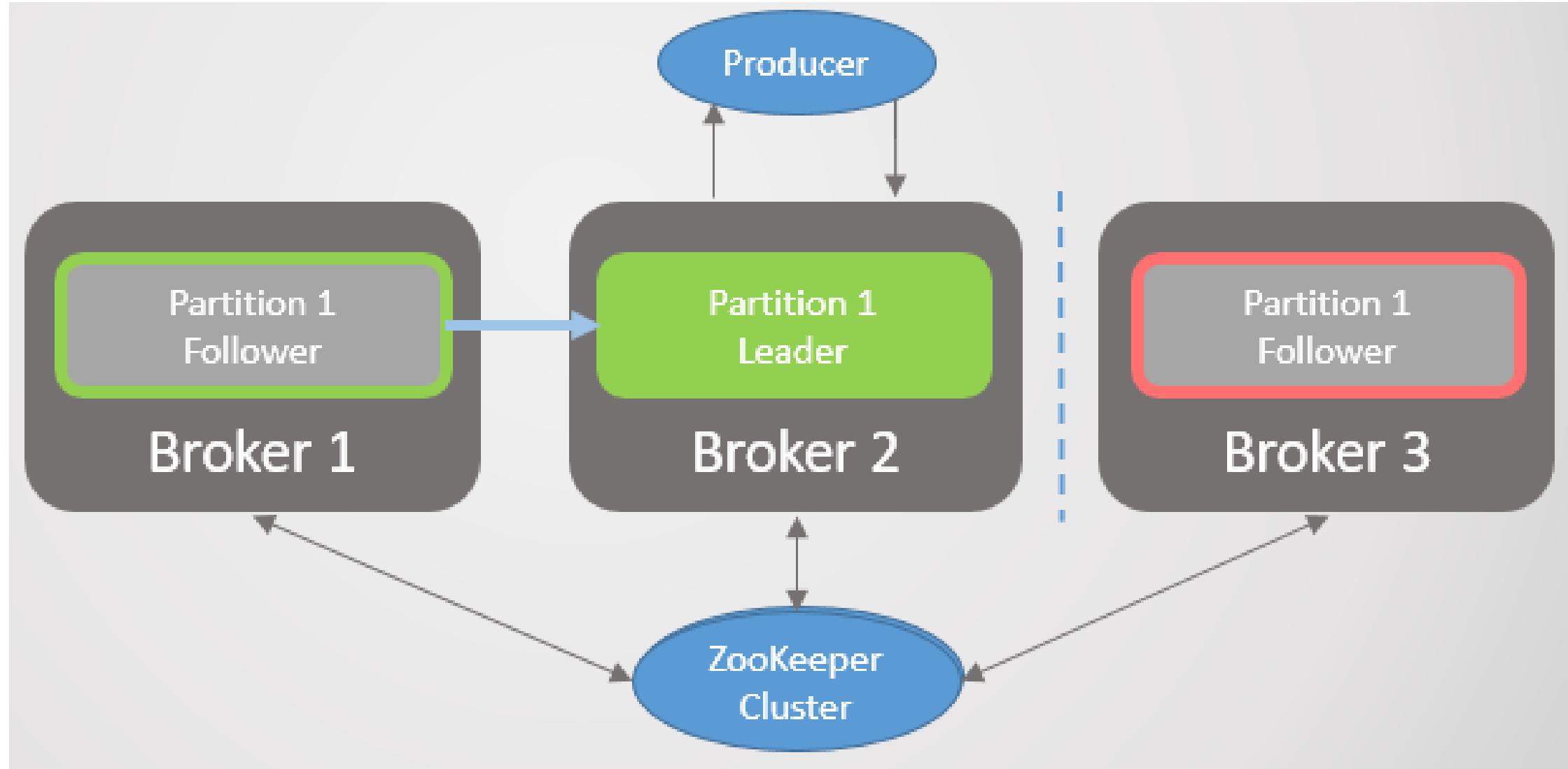
# Leader Fail-Over



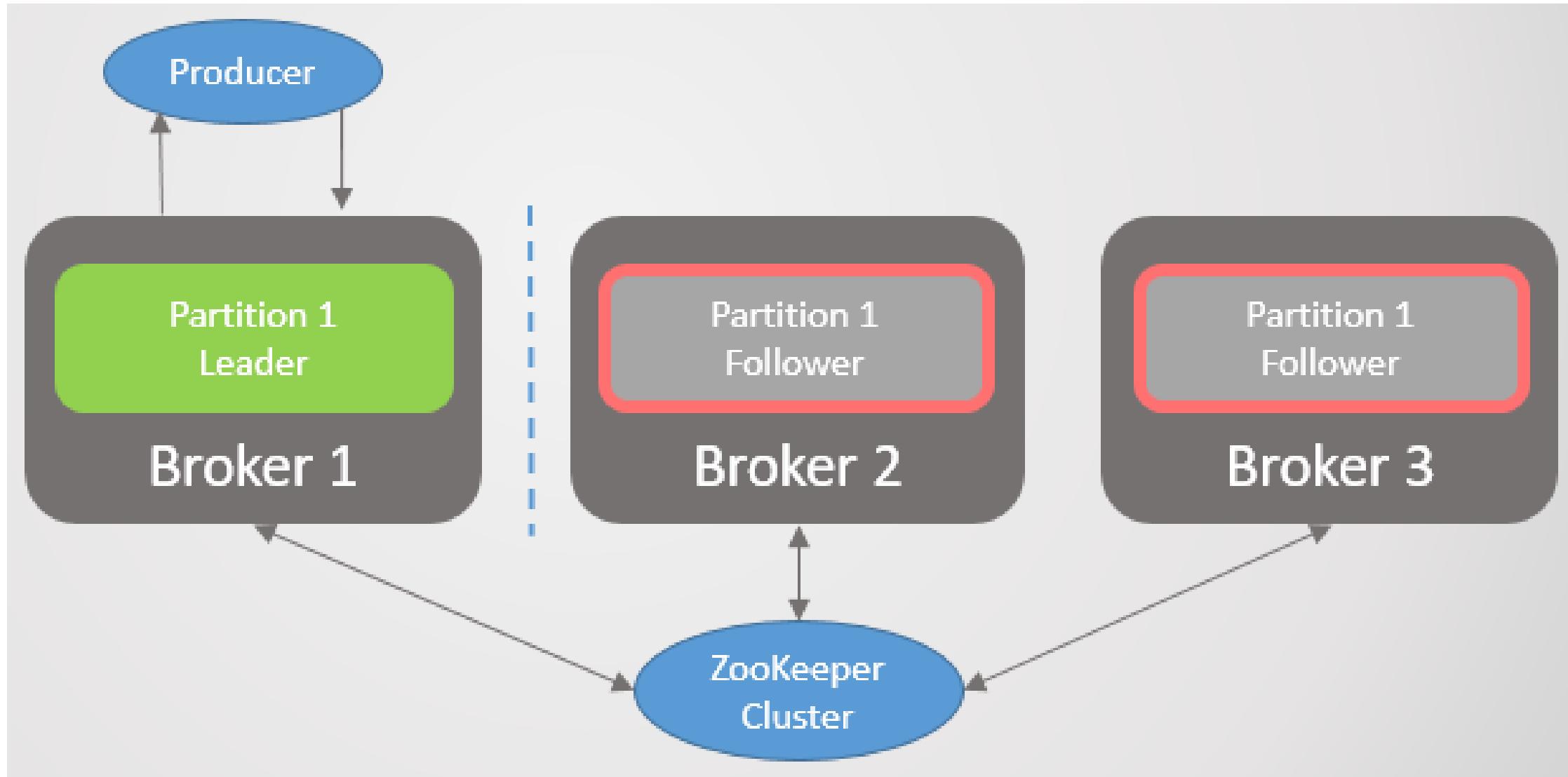
# Network Partitions



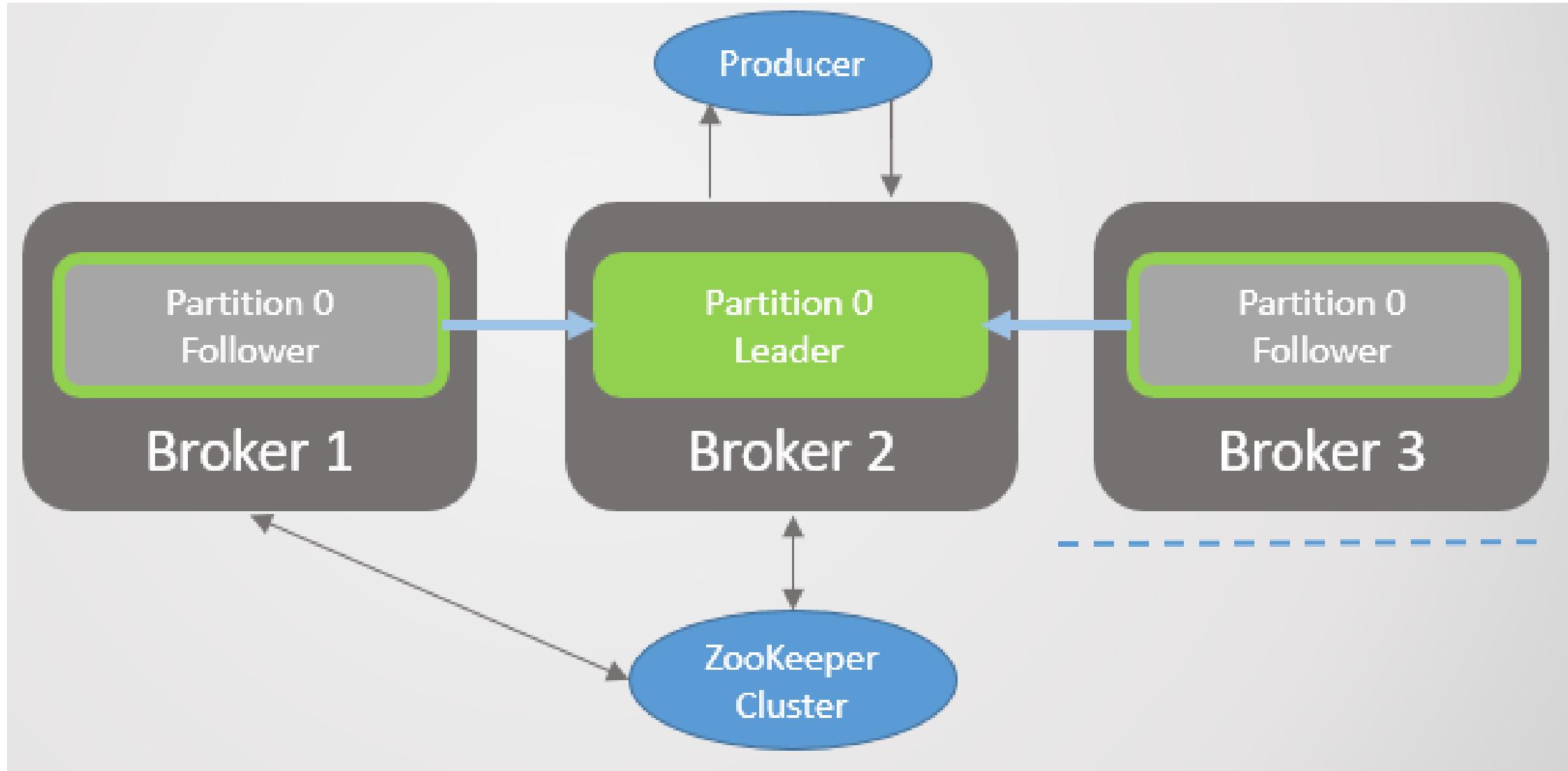
# Scenario 1: A follower cannot see the leader, but can still see Zookeeper



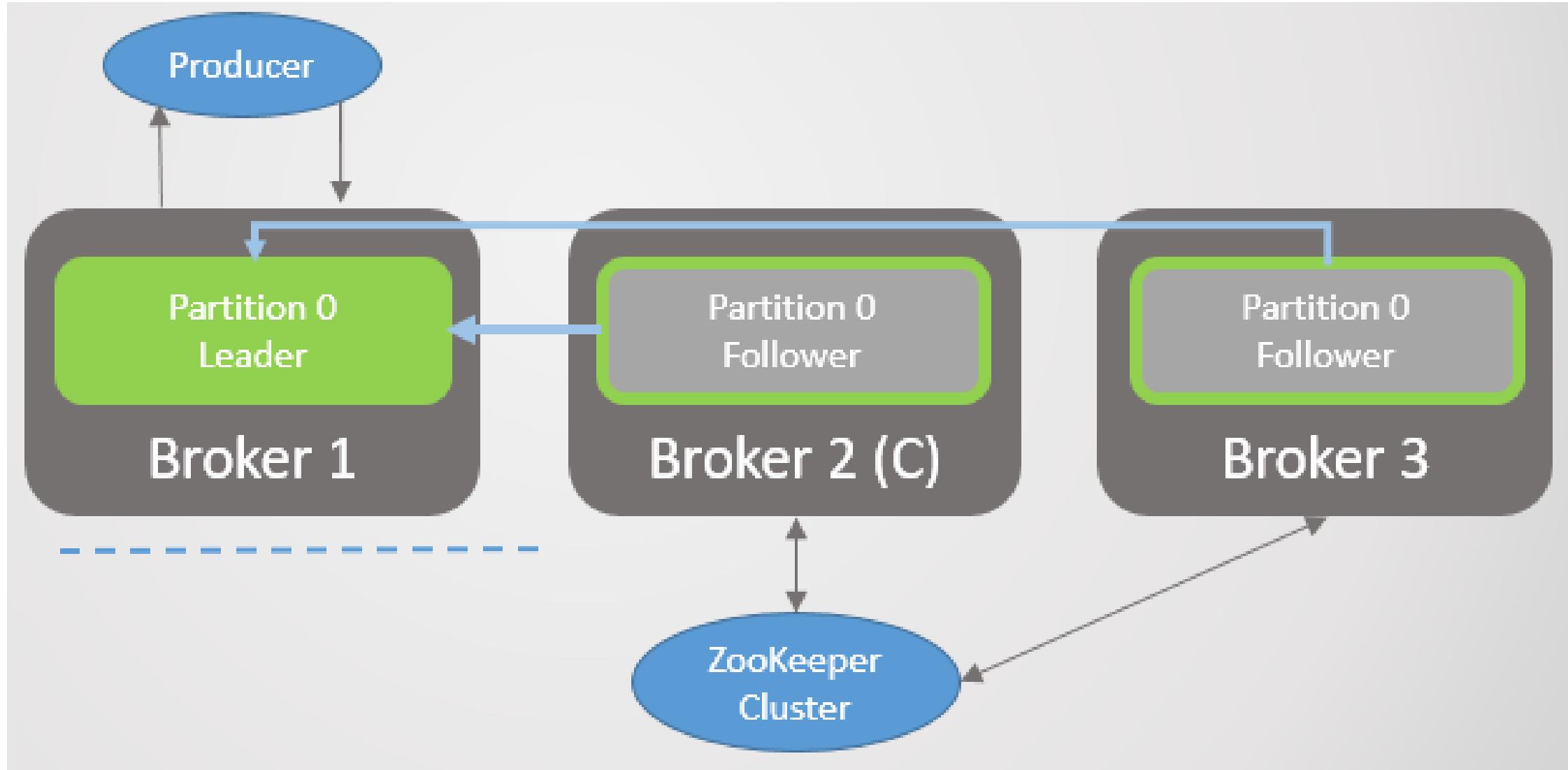
## Scenario 2: A leader cannot see any of its followers, but can still see Zookeeper



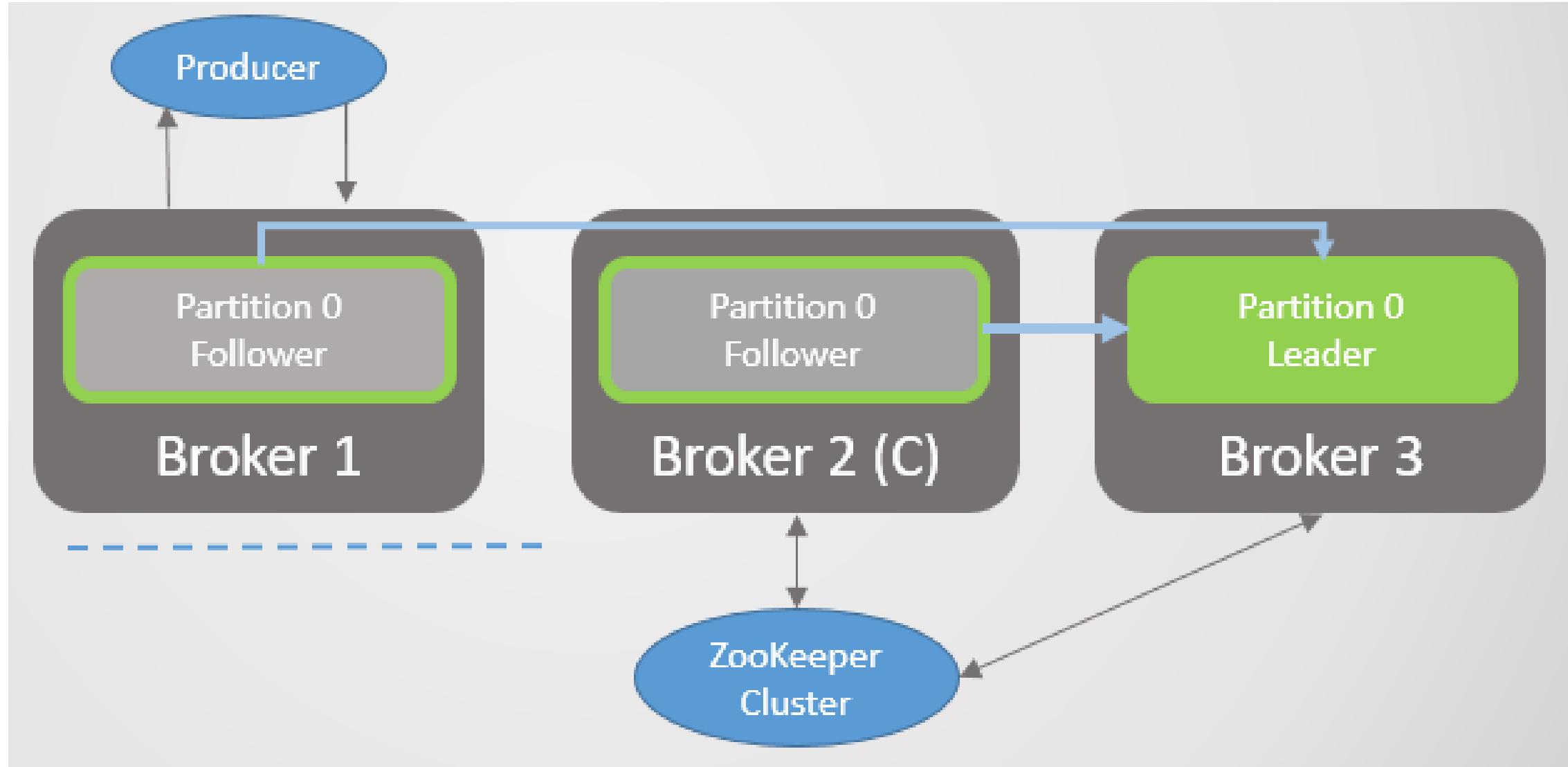
## Scenario 3: A follower can see the leader, but cannot see Zookeeper



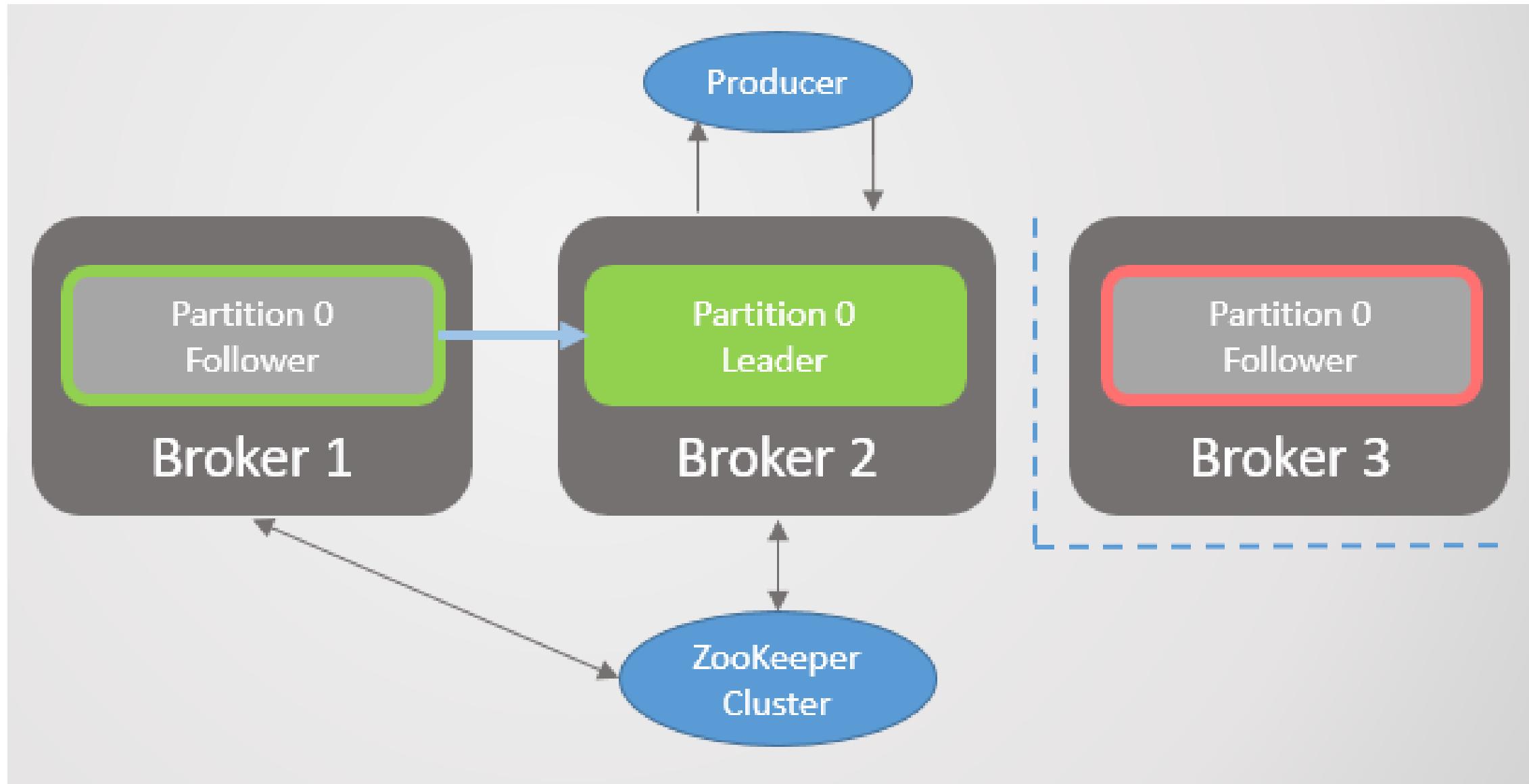
## Scenario 4: A leader can see its followers, but cannot see Zookeeper



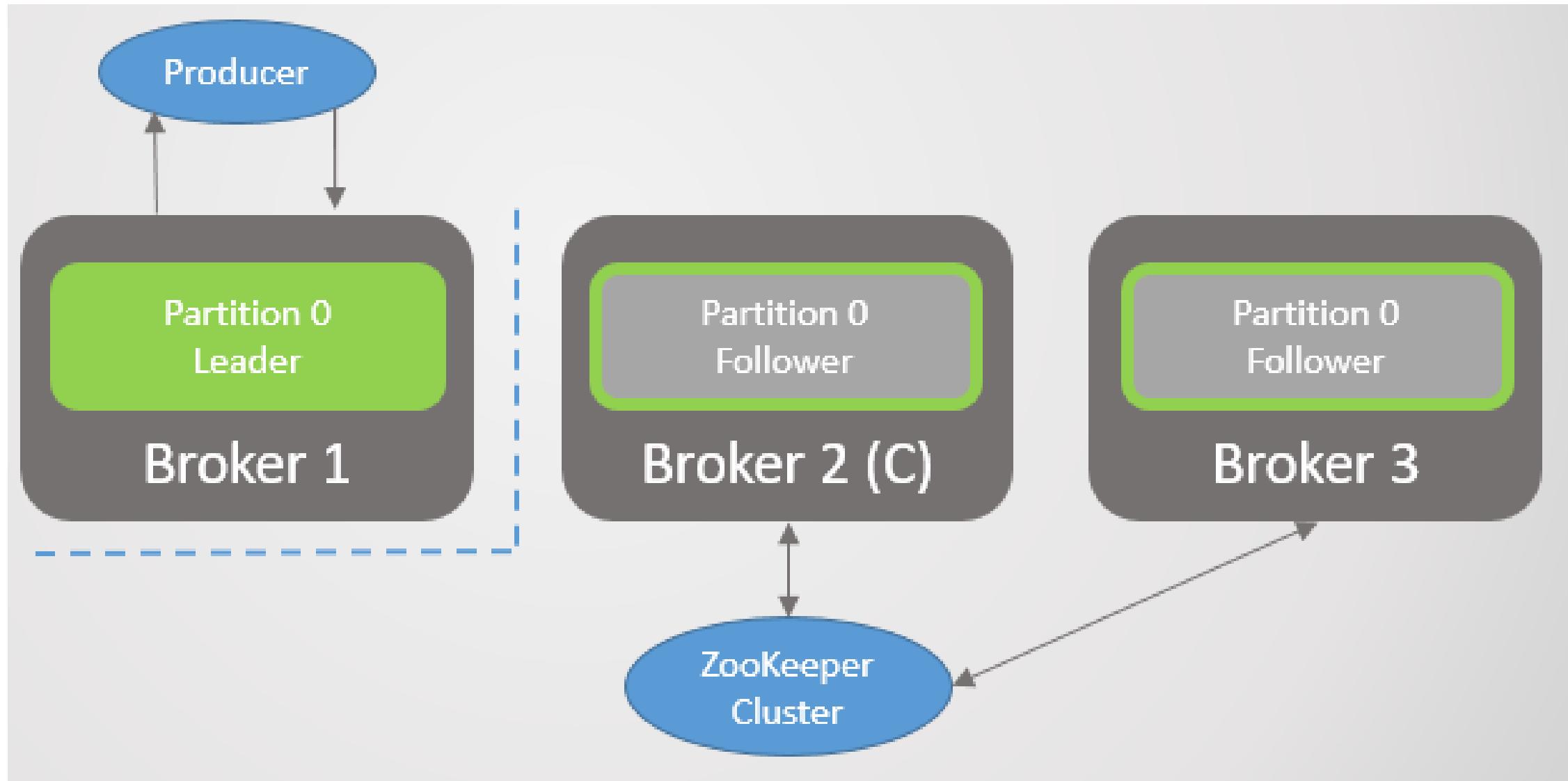
## Scenario 4. The leader on Broker 1 becomes a follower after the network partition is resolved



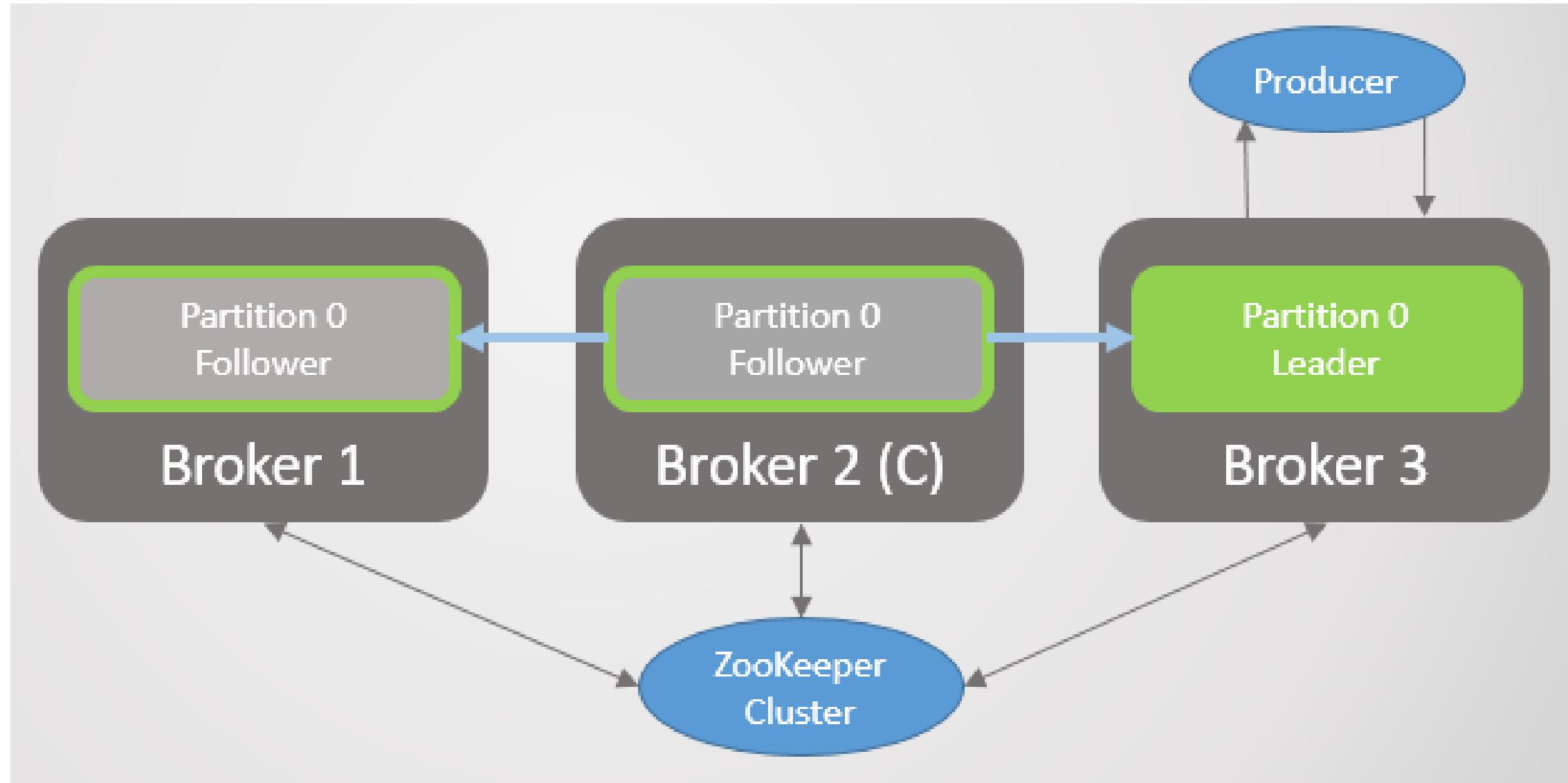
## Scenario 5: A follower is completely partitioned from both the other Kafka nodes and Zookeeper



## Scenario 6: A leader is completely partitioned from both the other Kafka nodes and Zookeeper



## Scenario 6: The original leader becomes a follower after the network partition is resolved



# Scenarios Conclusions - Message Loss

- \* **leader fail-over where acks=1**
- \* **unclean fail-over with acks=all**
- \* **A leader isolation from Zookeeper with acks=1**
- \* **A fully isolated leader, even acks=all if min.insync.replicas=1**
- \* **Simultaneous failures of all nodes of a partition.**
  - \* **messages are acknowledged once in memory, some messages may not yet have been written to disk.**



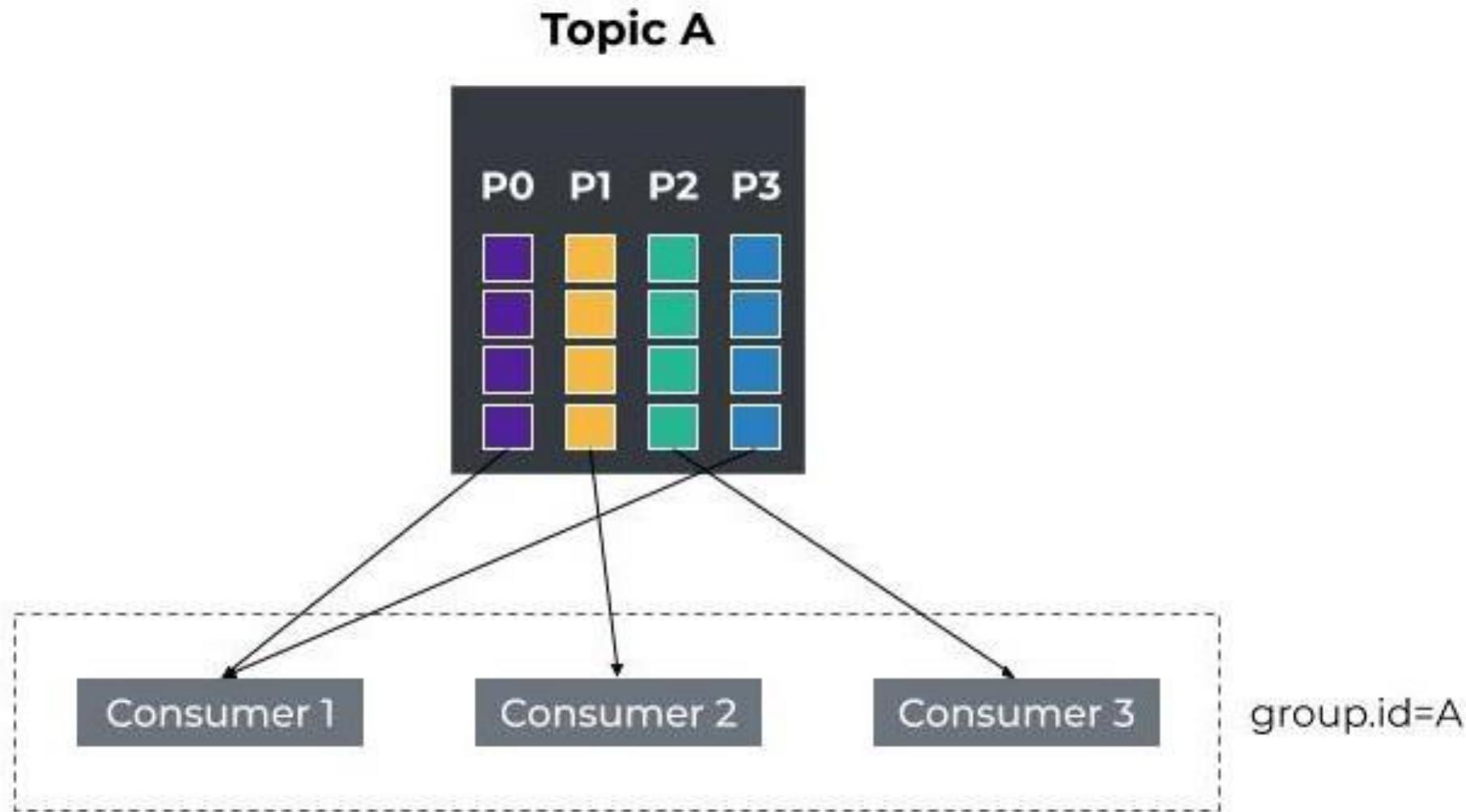


شرکت خدمات انفورماتیک  
informatics services corporation

## 8 - Clustering – Partition Assignment and Rebalancing



# Let's go back to some basics



*Rebalance/Rebalancing: the procedure that is followed by a number of distributed processes that use **Kafka clients** and/or the **Kafka coordinator** to form a common group and distribute a set of **resources** among the **members** of the group.*



شرکت خدمات اینفورماتیک

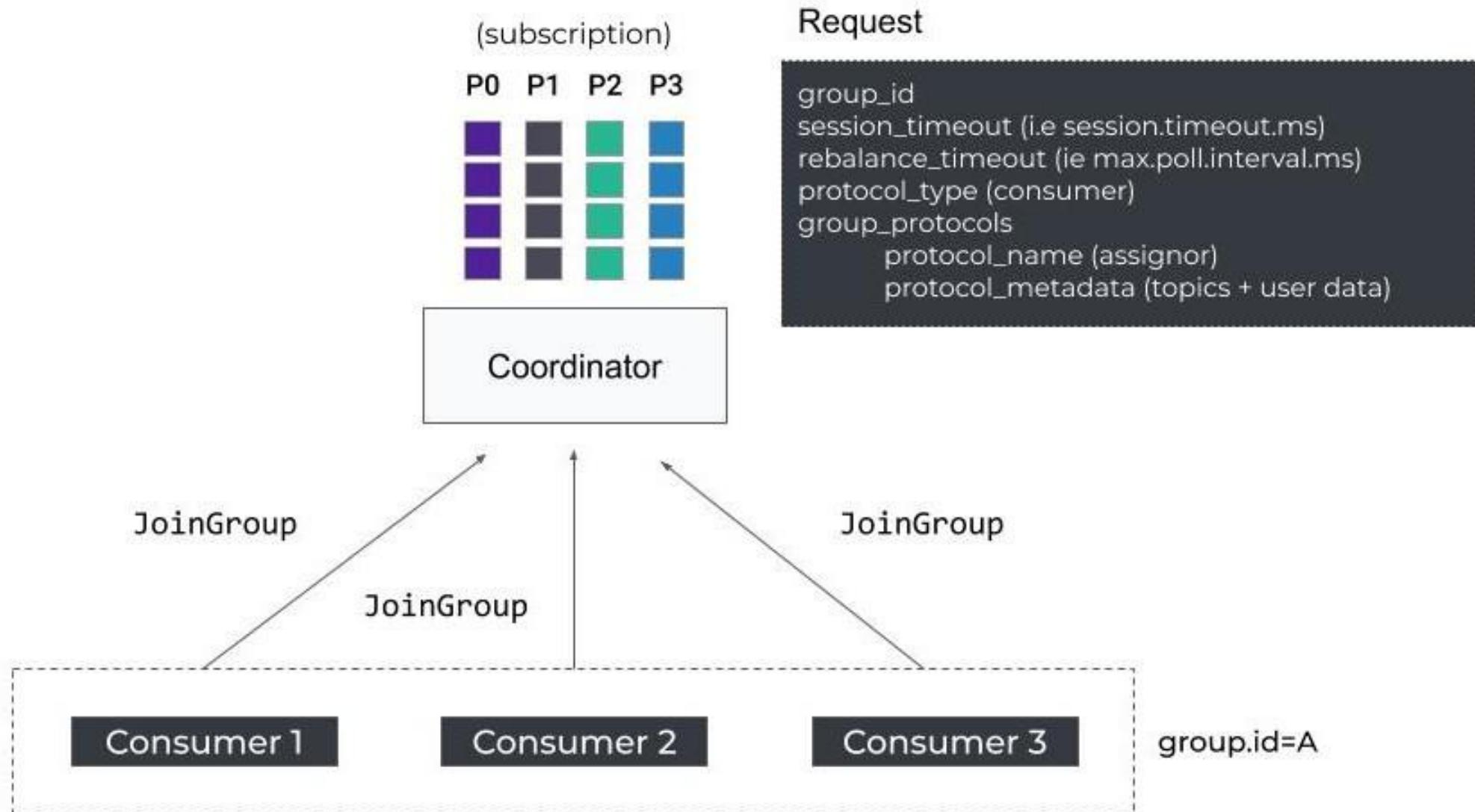
informatics services corporation



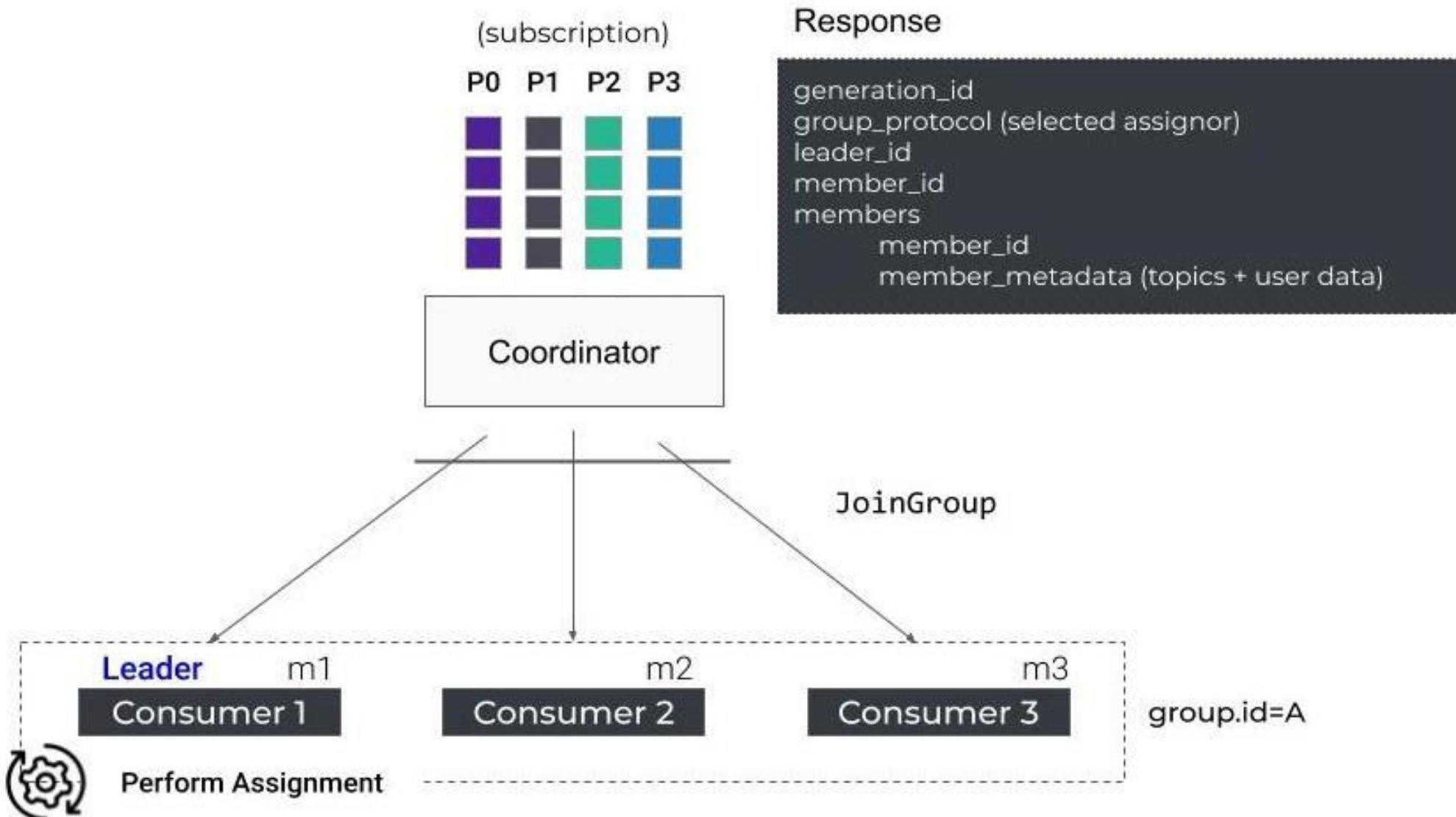
# Apache Kafka Rebalance Protocol and components



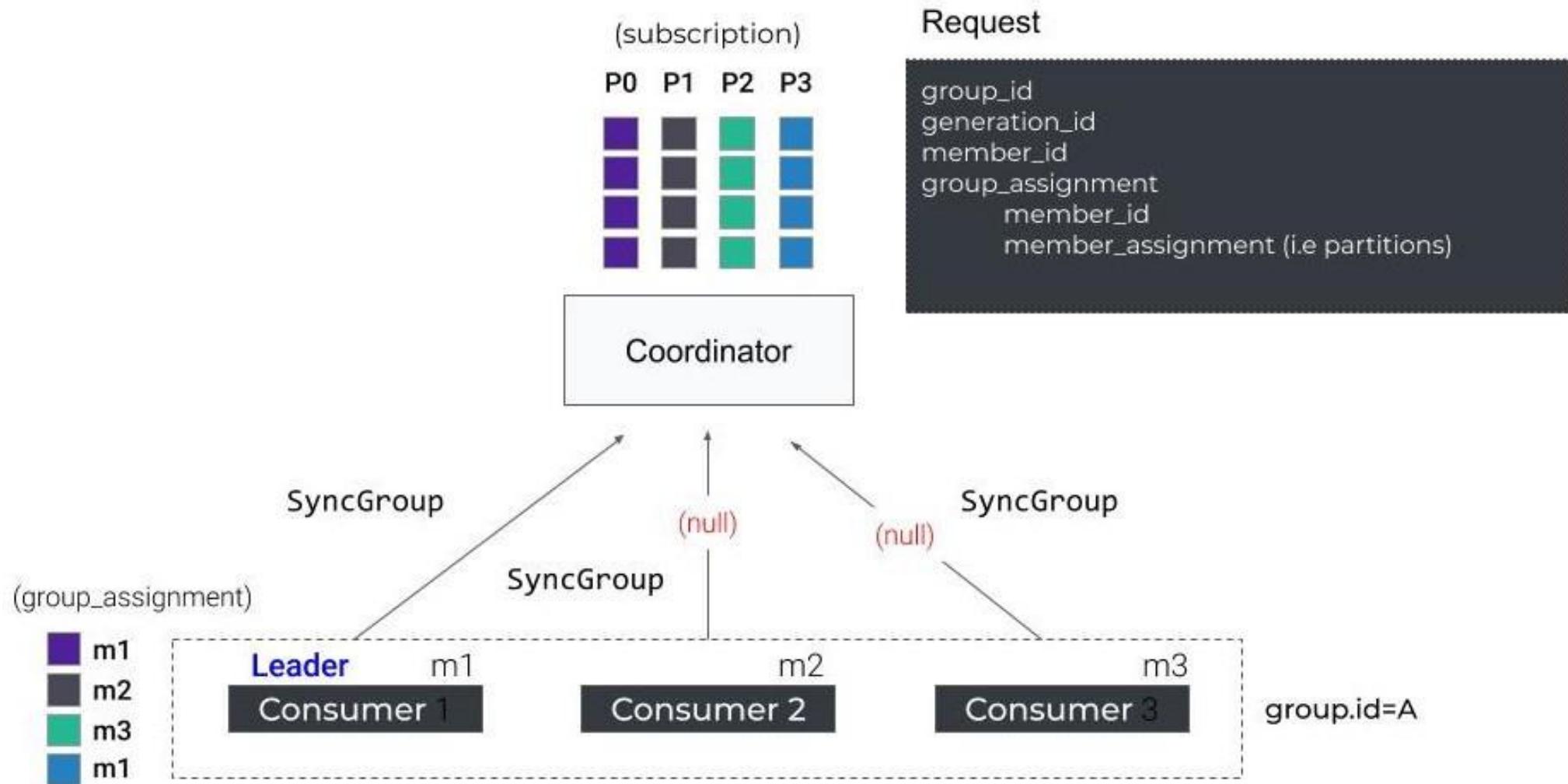
# Join Group Request



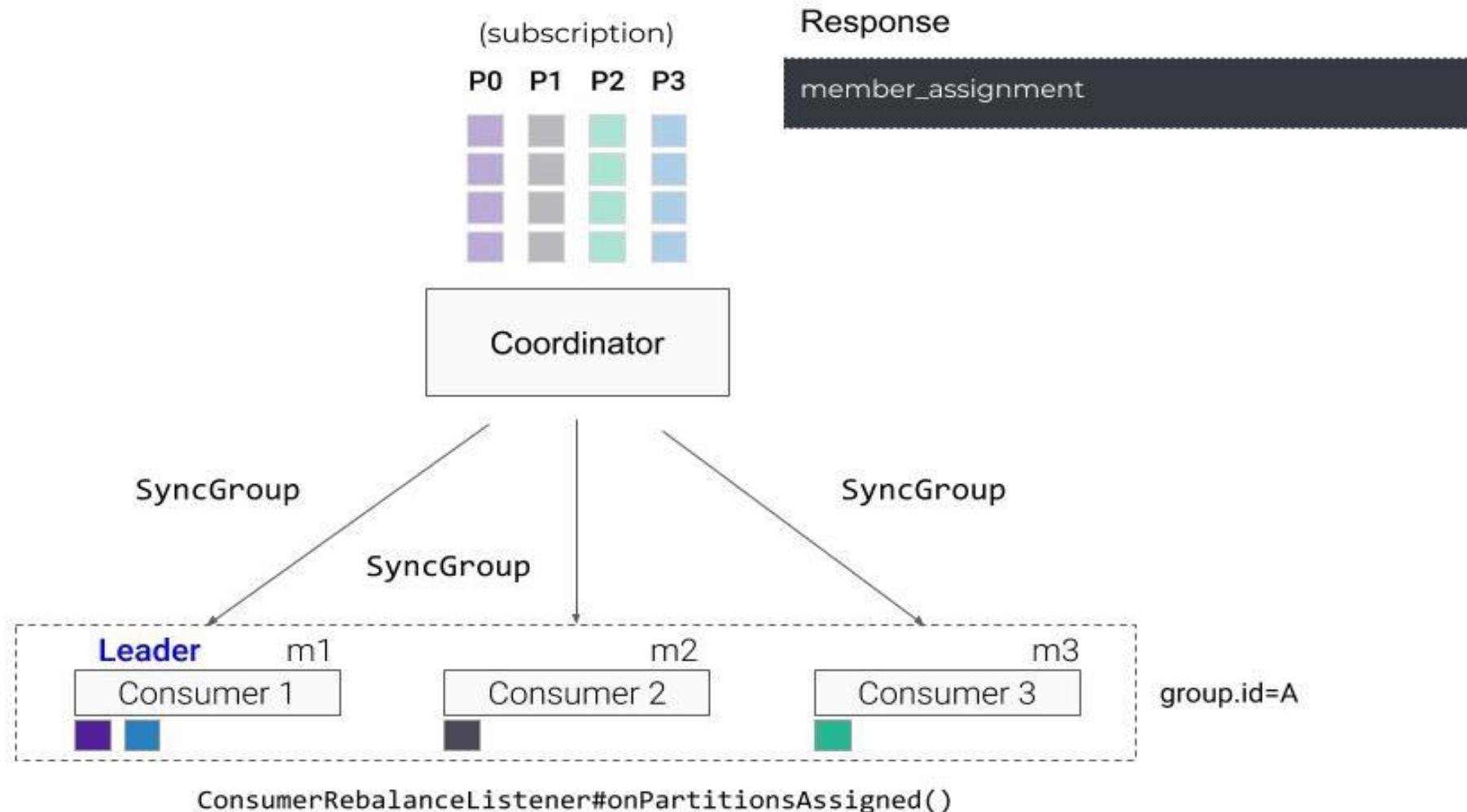
# Join Group Response



# Sync Group Request



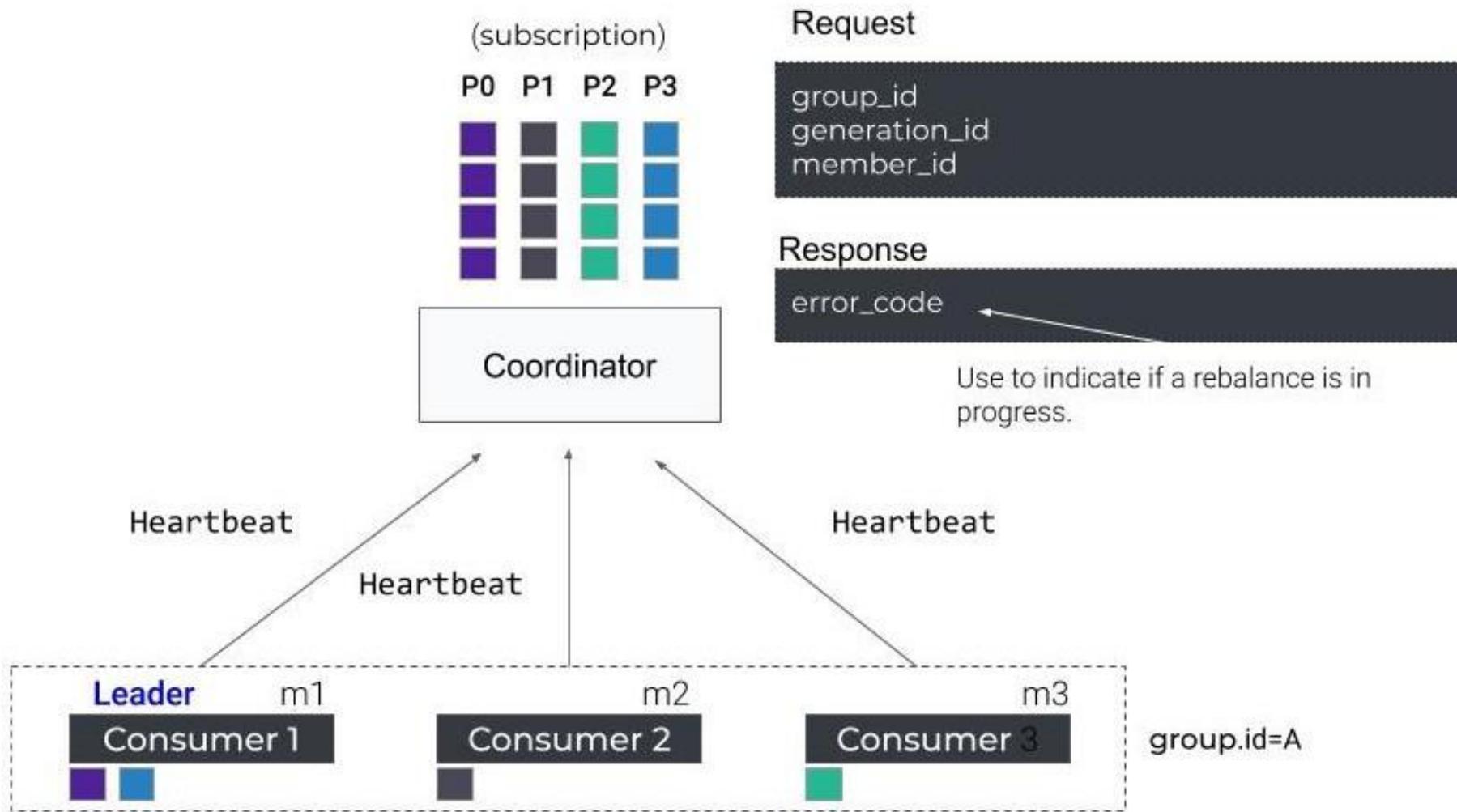
# Sync Group Response



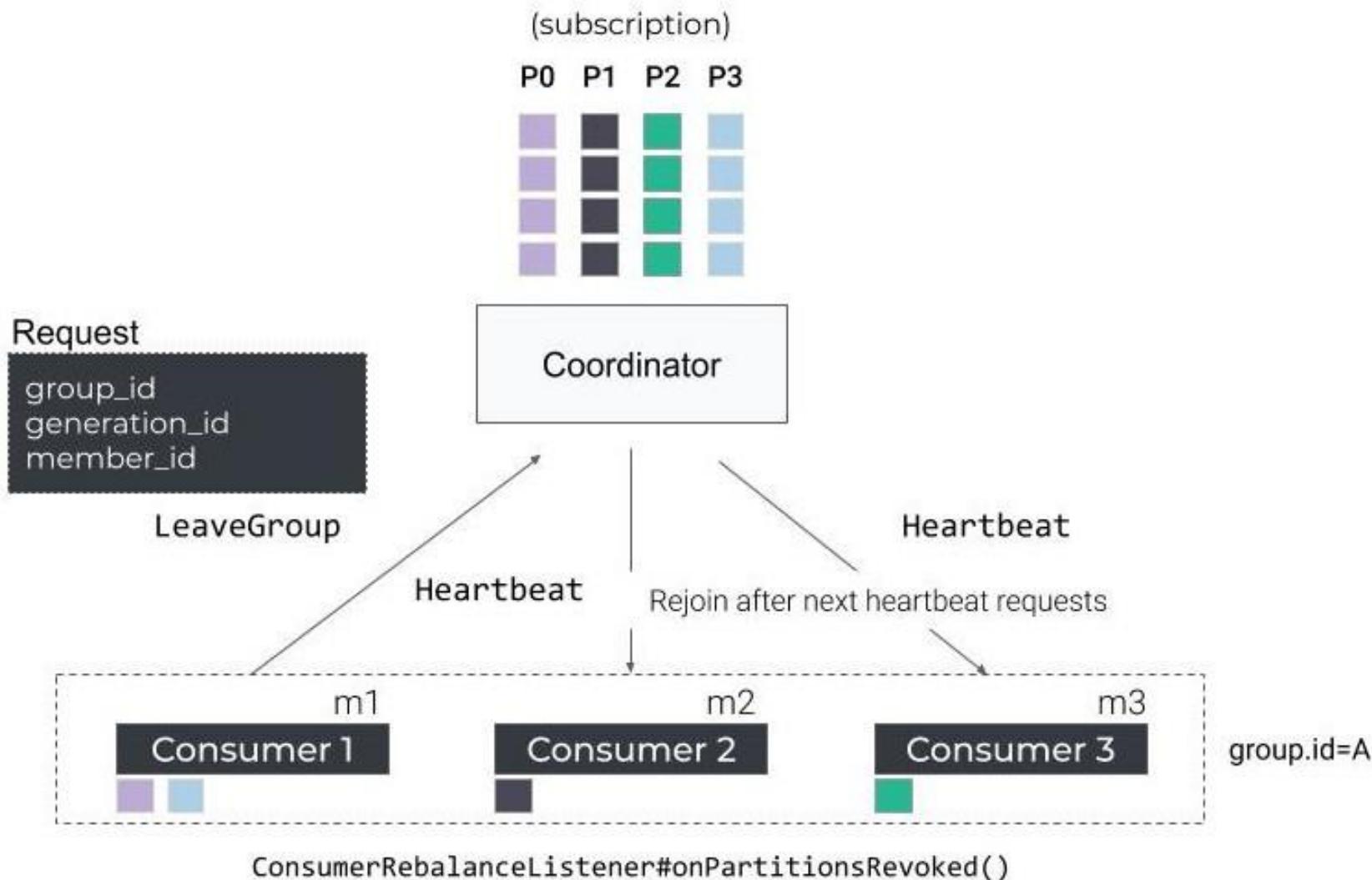
18



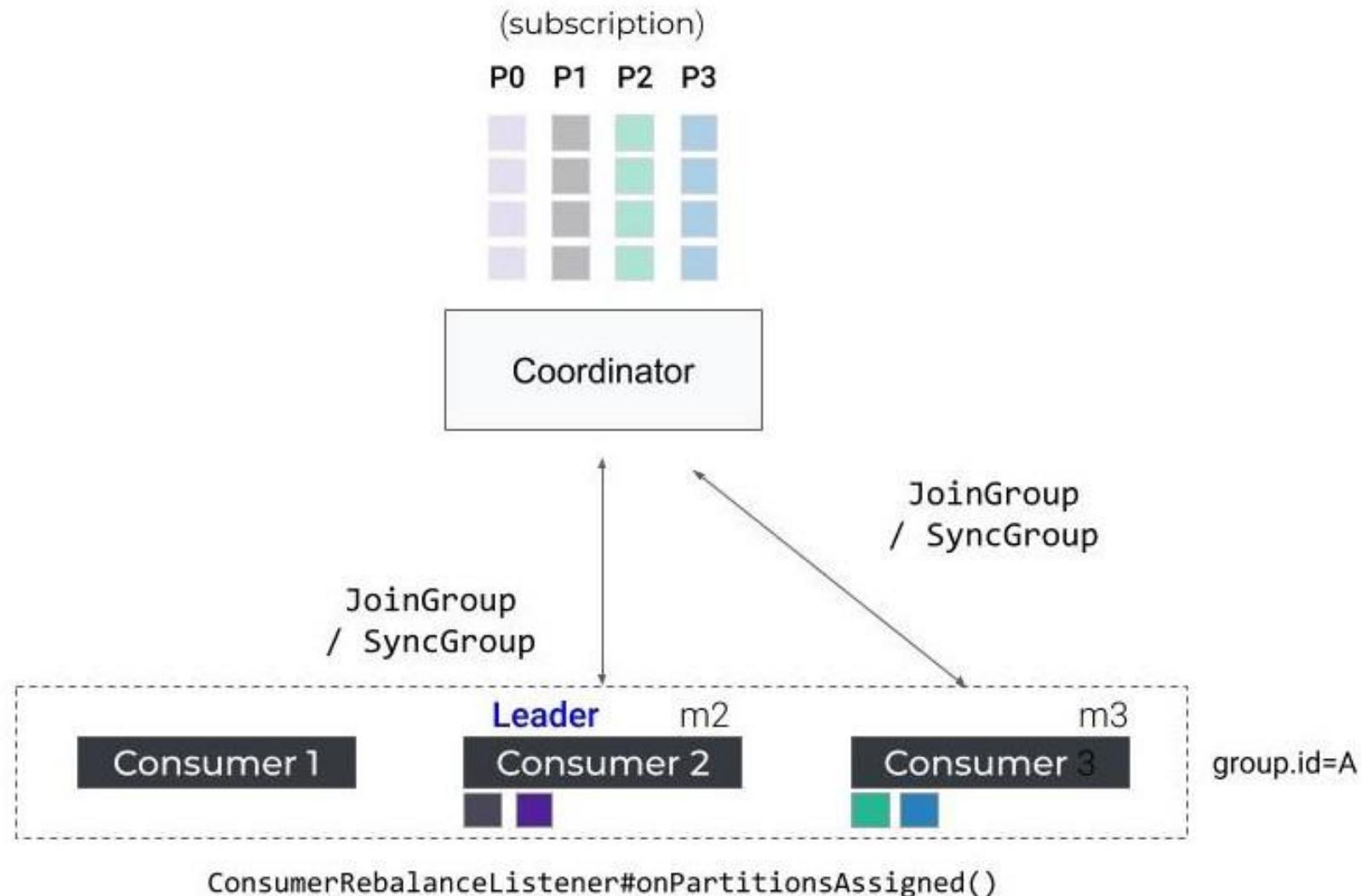
# Heartbeat



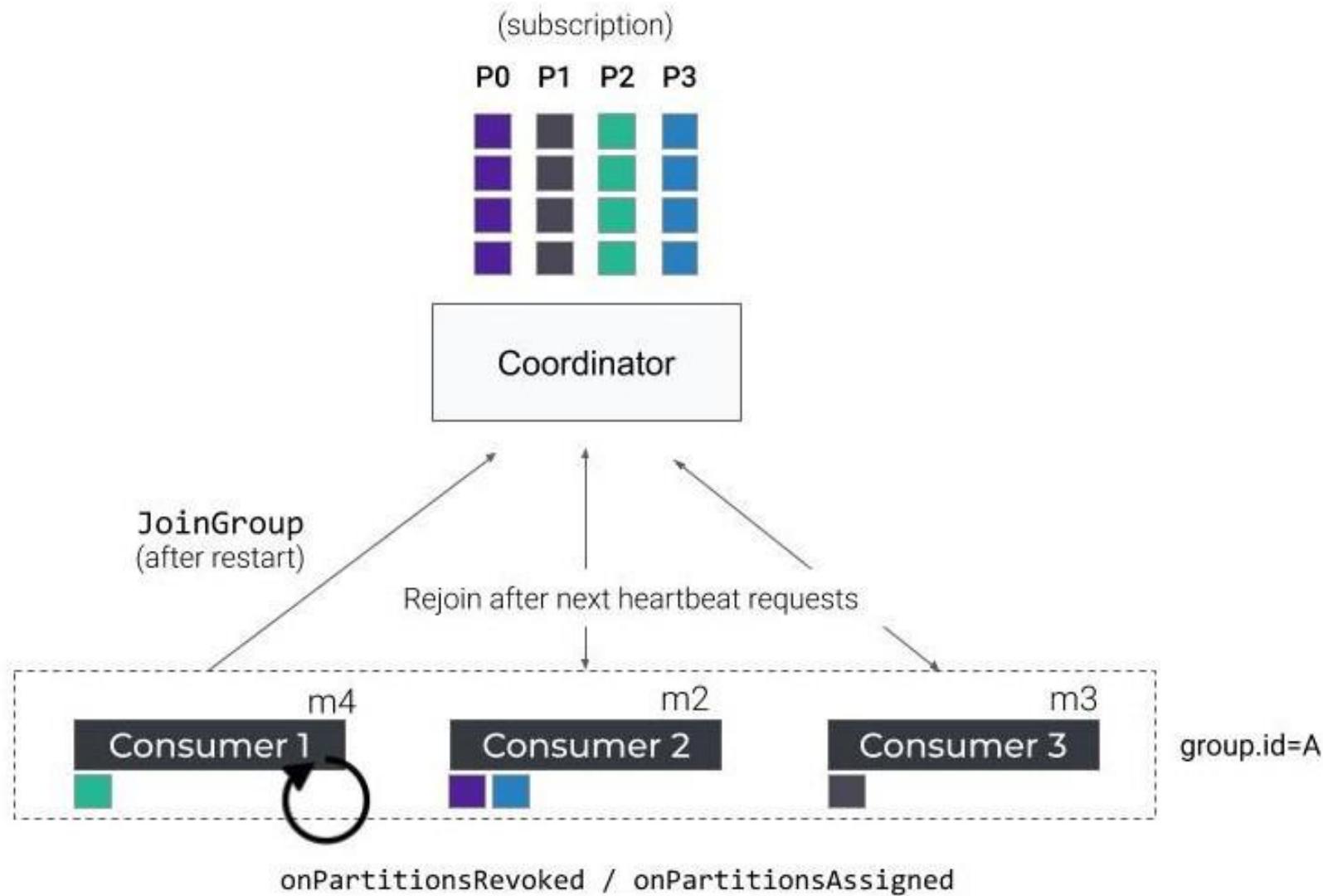
# Consumer — Rebalance Protocol — Leave Group



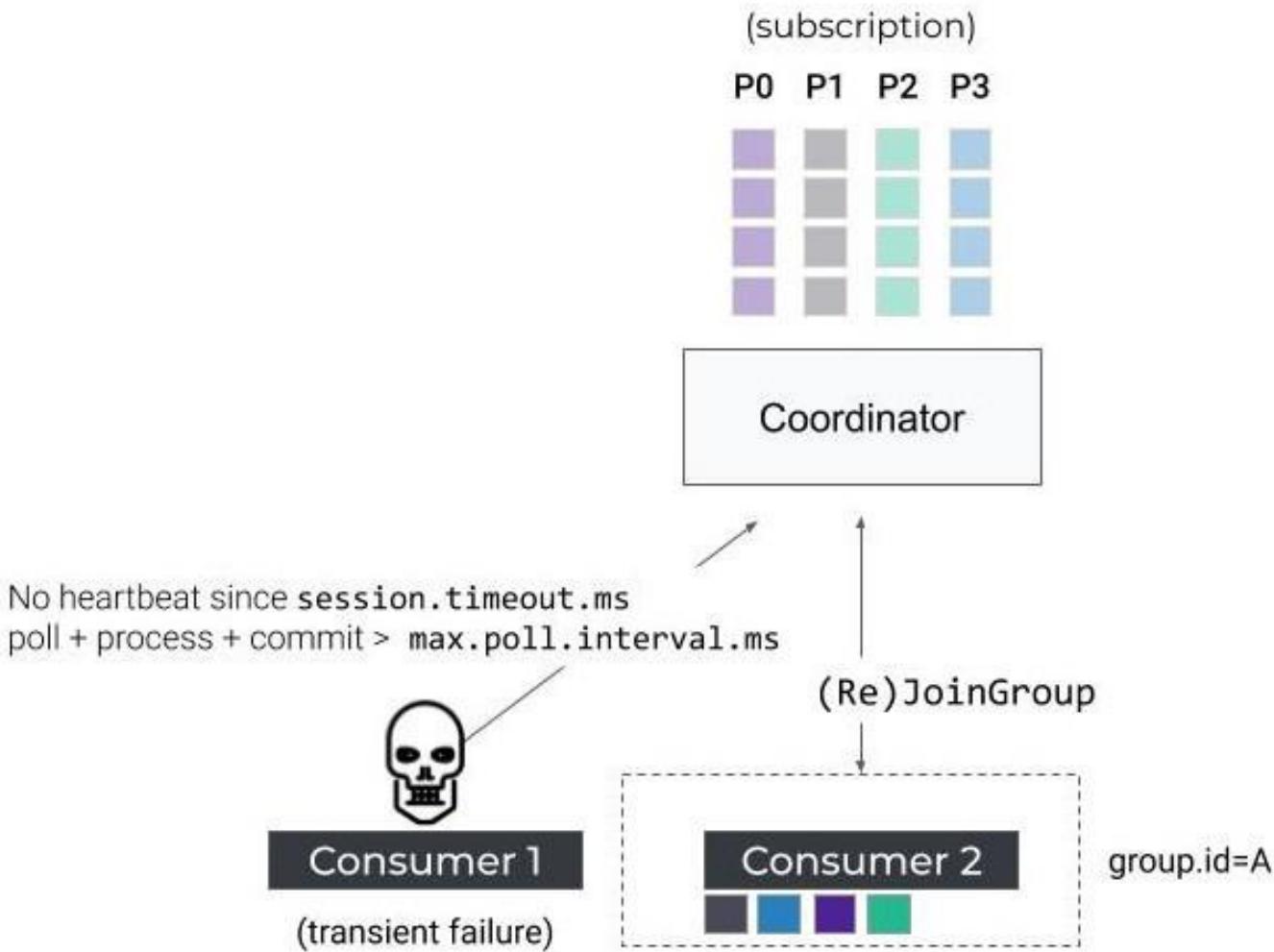
# Consumer — Rebalance Protocol — Rejoin



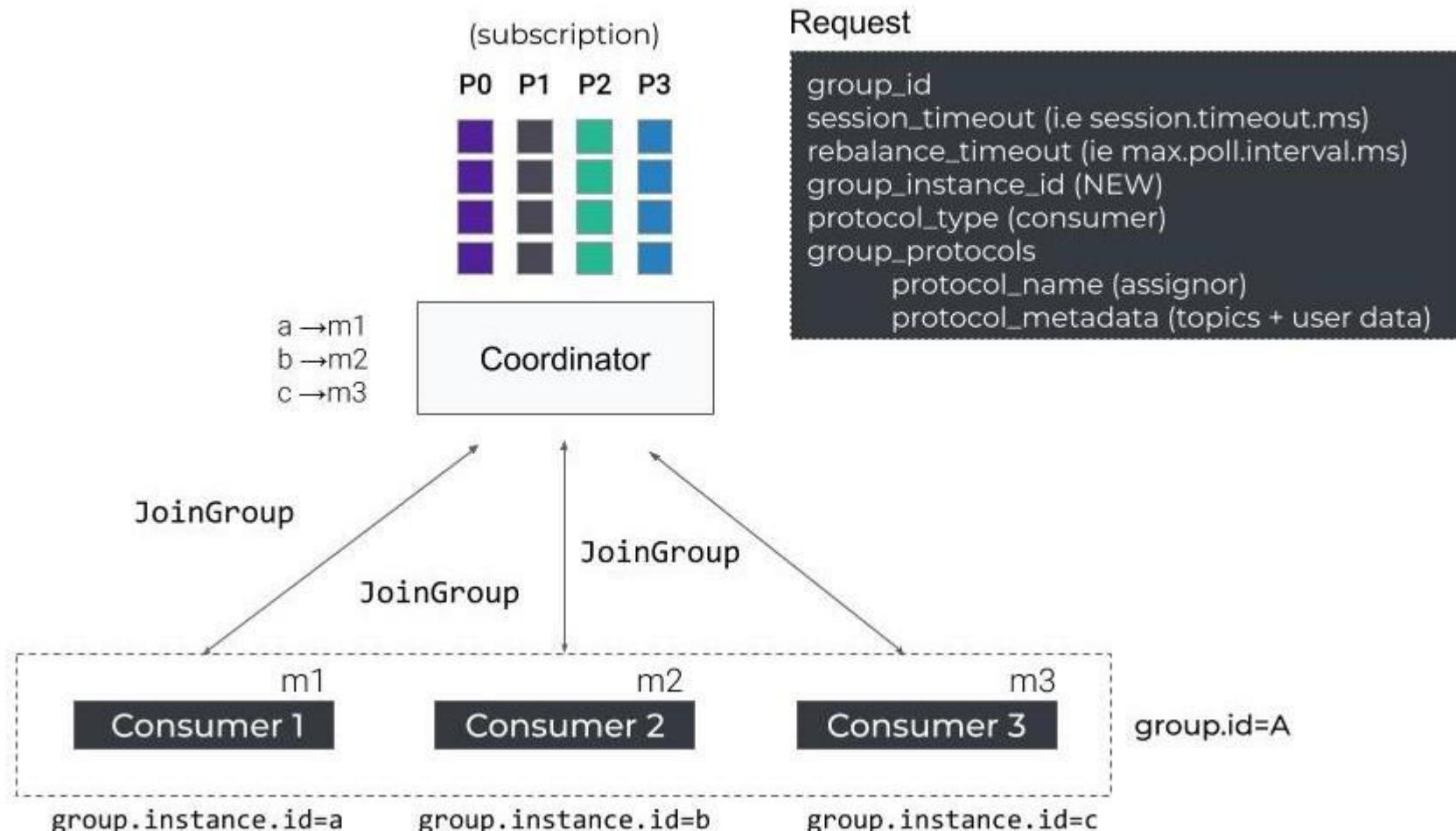
# Consumer — Rebalance Protocol — Restart



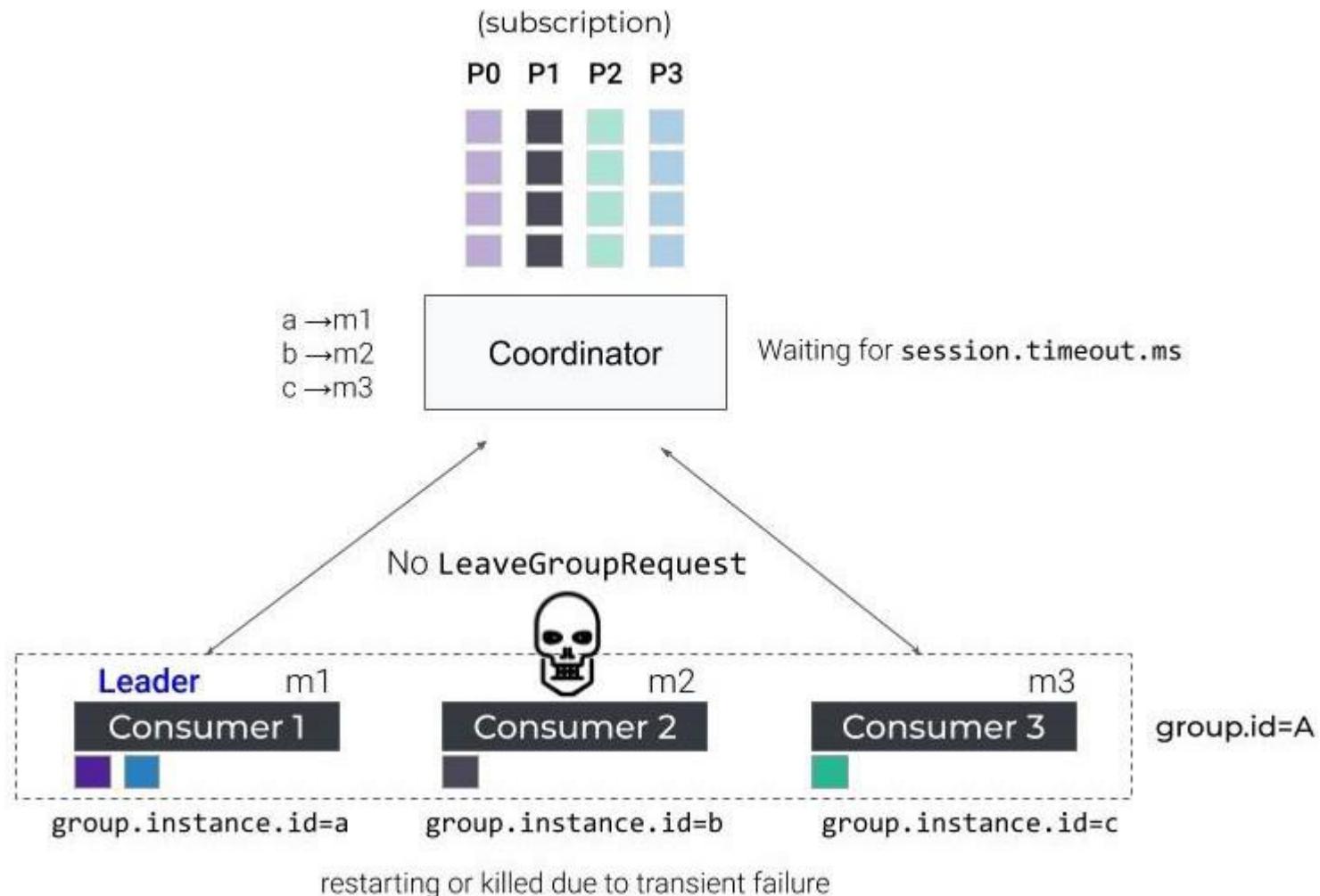
# Consumer — Rebalance Protocol — Timeout



# Static Membership



# Static Membership - Restarting or Killed due to transient failure



# Incremental Cooperative Rebalancing

- \* New embedded protocols
- \* Improve the resource availability of each member
- \* Minimizing stop-the-world effect
- \* The basic idea behind is perform rebalancing
  - \* Incrementally
  - \* In cooperation



# KIP - Rebalancing Protocols

- \* **KIP : Kafka Improvement Proposals**
- \* **KIP-345: Introduce static membership protocol to reduce consumer rebalances**
  - \* Released: 2.4.0
  - \* Last modified: Sep 09 2019
- \* **KIP-429: Kafka Consumer Incremental Rebalance Protocol**
  - \* Released: 2.4.0
  - \* Last modified: Mar 16 2021



# The Partition Assignor Strategies

- \* What is it?

Protocol to customize how partitions are assigned to the group members.

- \* How to configure?

Configurable through the consumer property

*partition.assignment.strategy*

- \* Built-in

- \* Range

- \* RoundRobin

- \* StickyAssignor



# Specify a Partition Assignor - code snippet

```
Properties props = new Properties();  
...  
props.put(ConsumerConfig.PARTITION_ASSIGNMENT_STRATEGY_CONFIG,  
StickyAssignor.class.getName());  
  
KafkaConsumer<String, String> consumer =  
    new KafkaConsumer<>(props);  
//...
```



شرکت خدمات اینفورماتیک

informatics services corporation



# Different strategy Issue

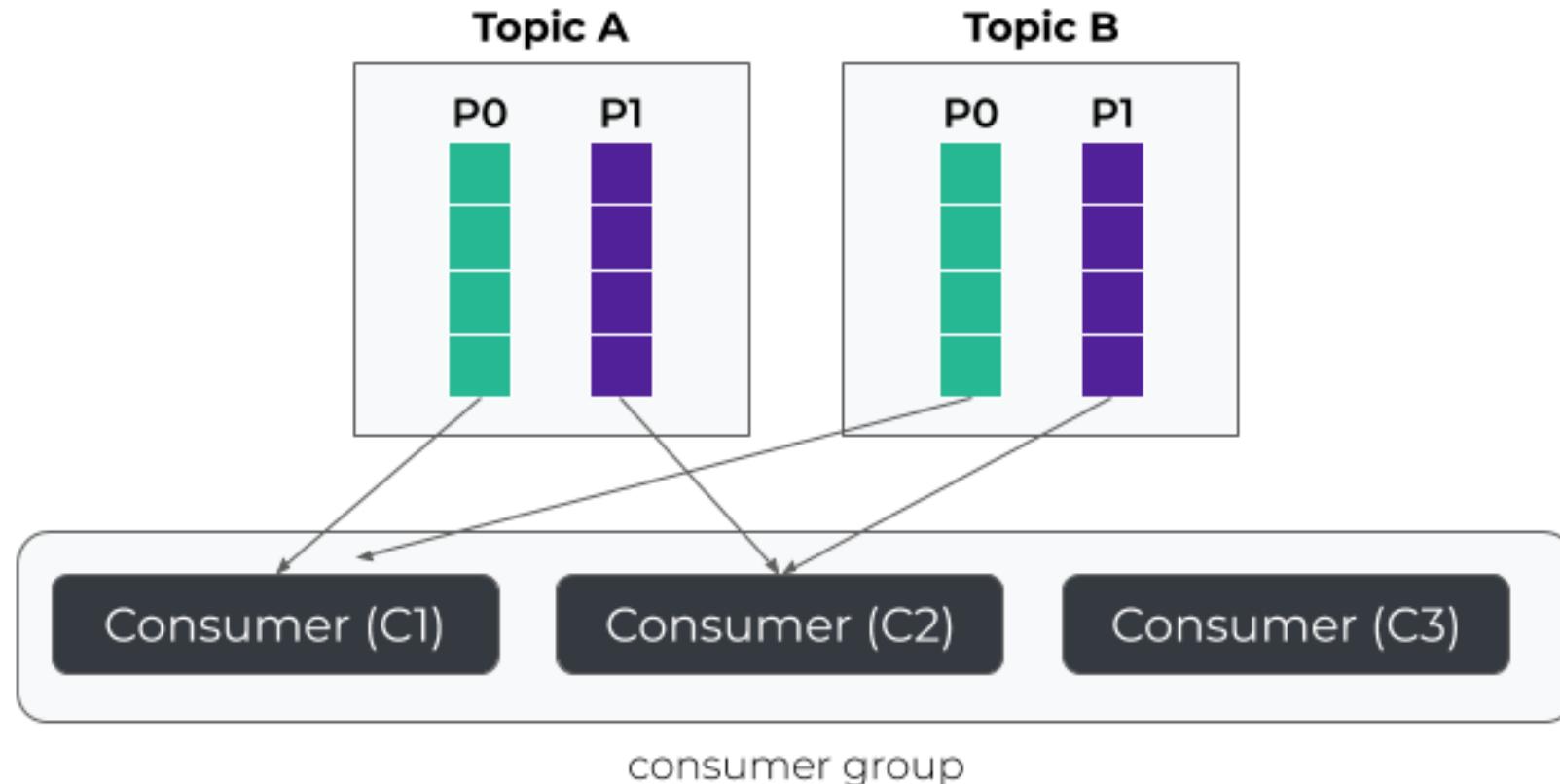
- \* All consumers which belong to the same group must have one common strategy declared

- \* An assignment configuration inconsistent with other group members

`org.apache.kafka.common.errors.InconsistentGroupProtocolException: The  
informatics services corporation  
group member's supported protocols are incompatible with those of  
existing members or first group member tried to join with empty  
protocol type or empty protocol list.`



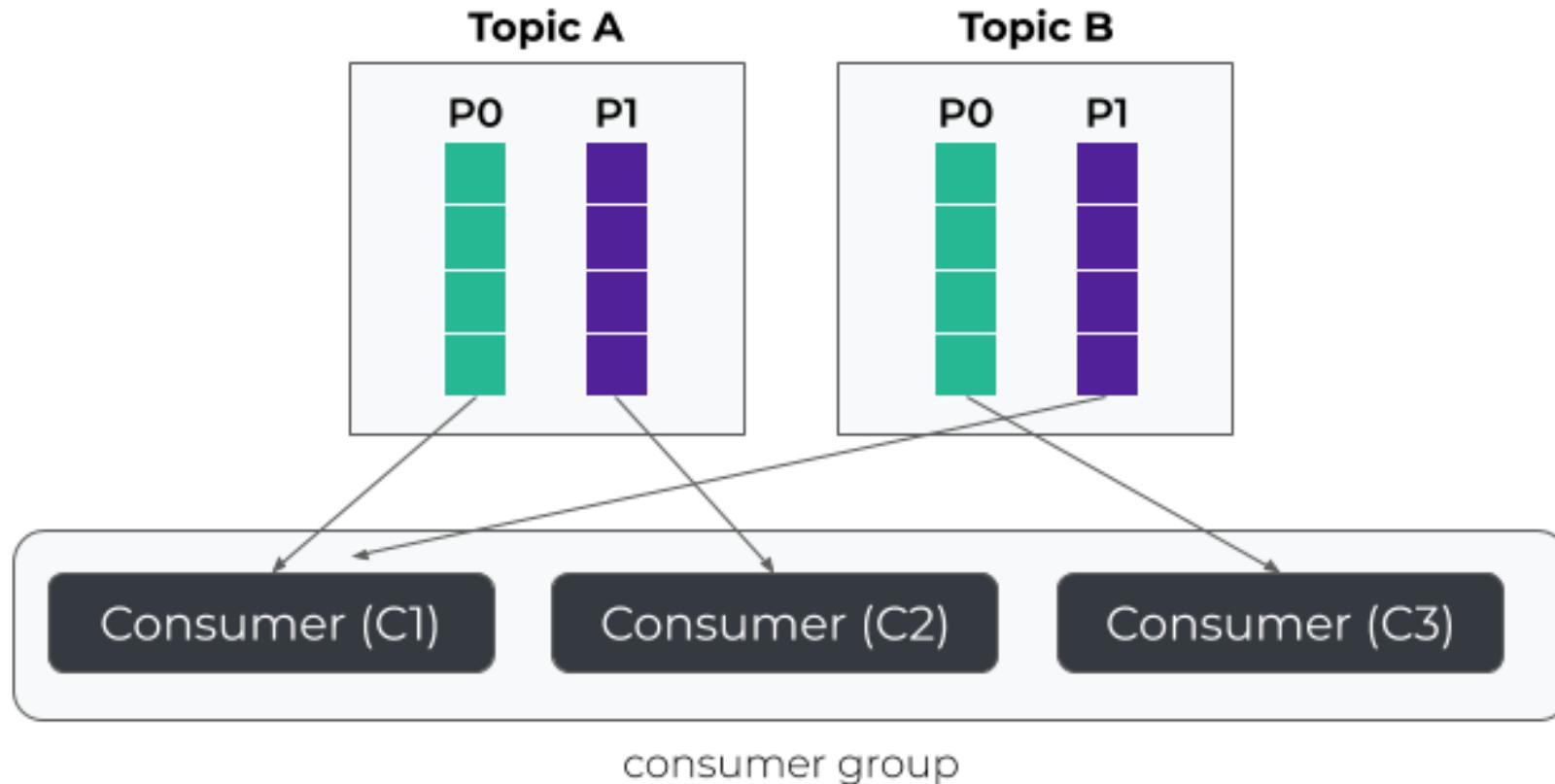
# Range Assignor



Assignment  $\rightarrow$  C1 = {A-0, B-0}, C2 = {A-1, B-1}, C3 = {}



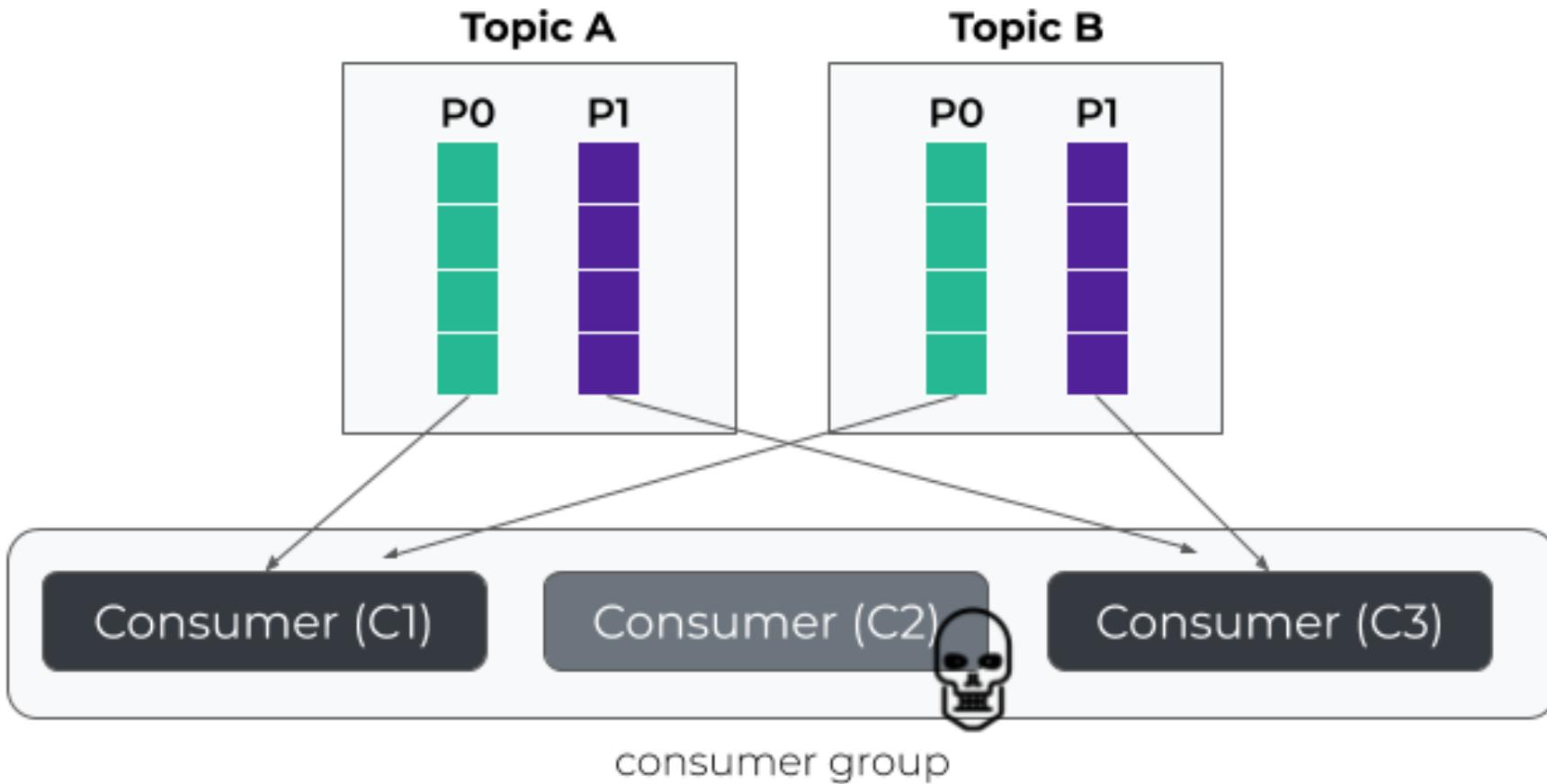
# Round Robin Assignor



Assignment  $\rightarrow$  C1 = {A-0, B-1}, C2 = {A-1}, C3 = {B-0}



# Round Robin Assignor with reassignment

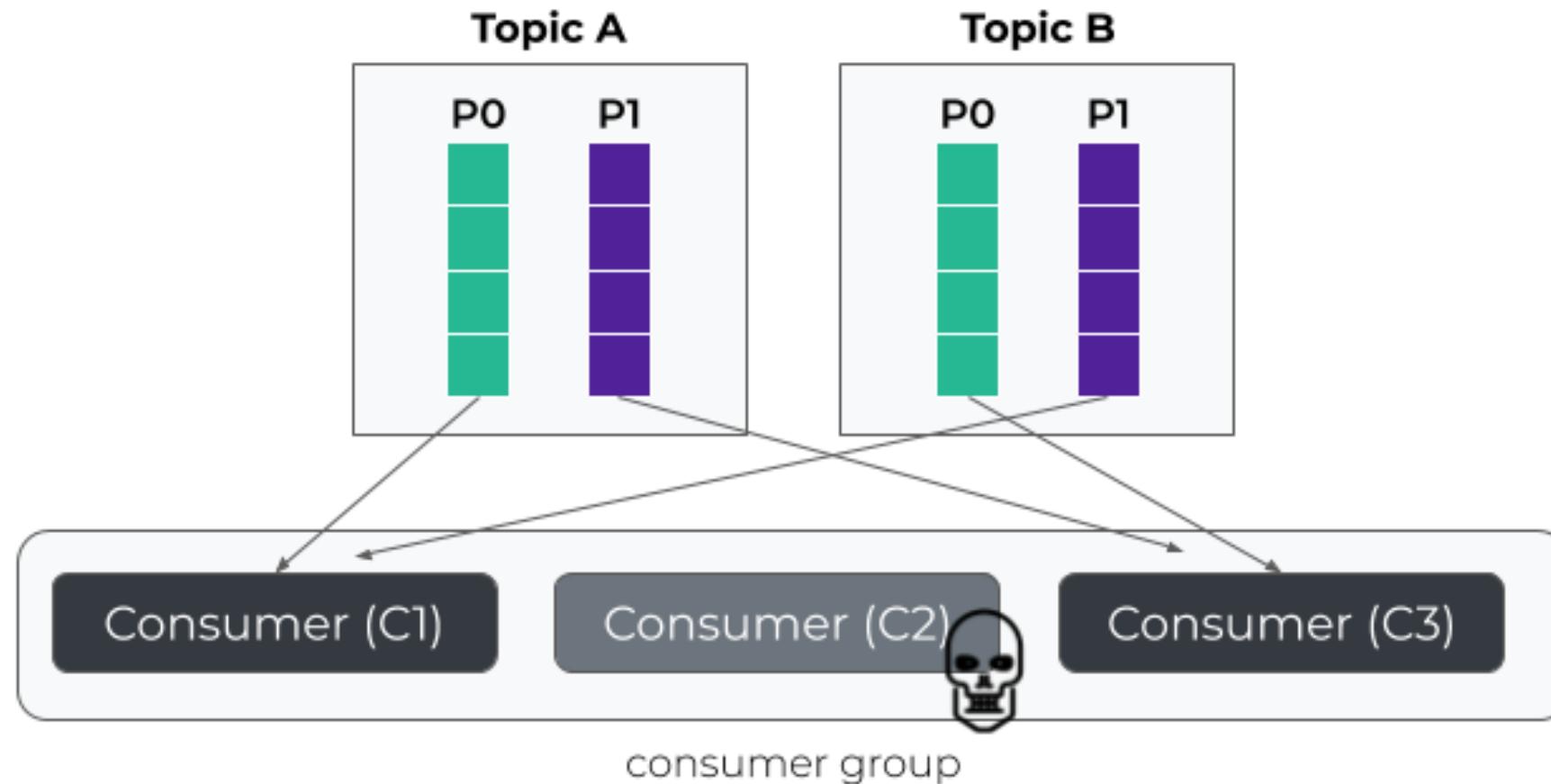


before —>  $C1 = \{A-0, B-1\}, C2 = \{A-1\}, C3 = \{B-0\}$

after —>  $C1 = \{A-0, B-0\}, C3 = \{A-1, B-1\}$



# Sticky Assignor

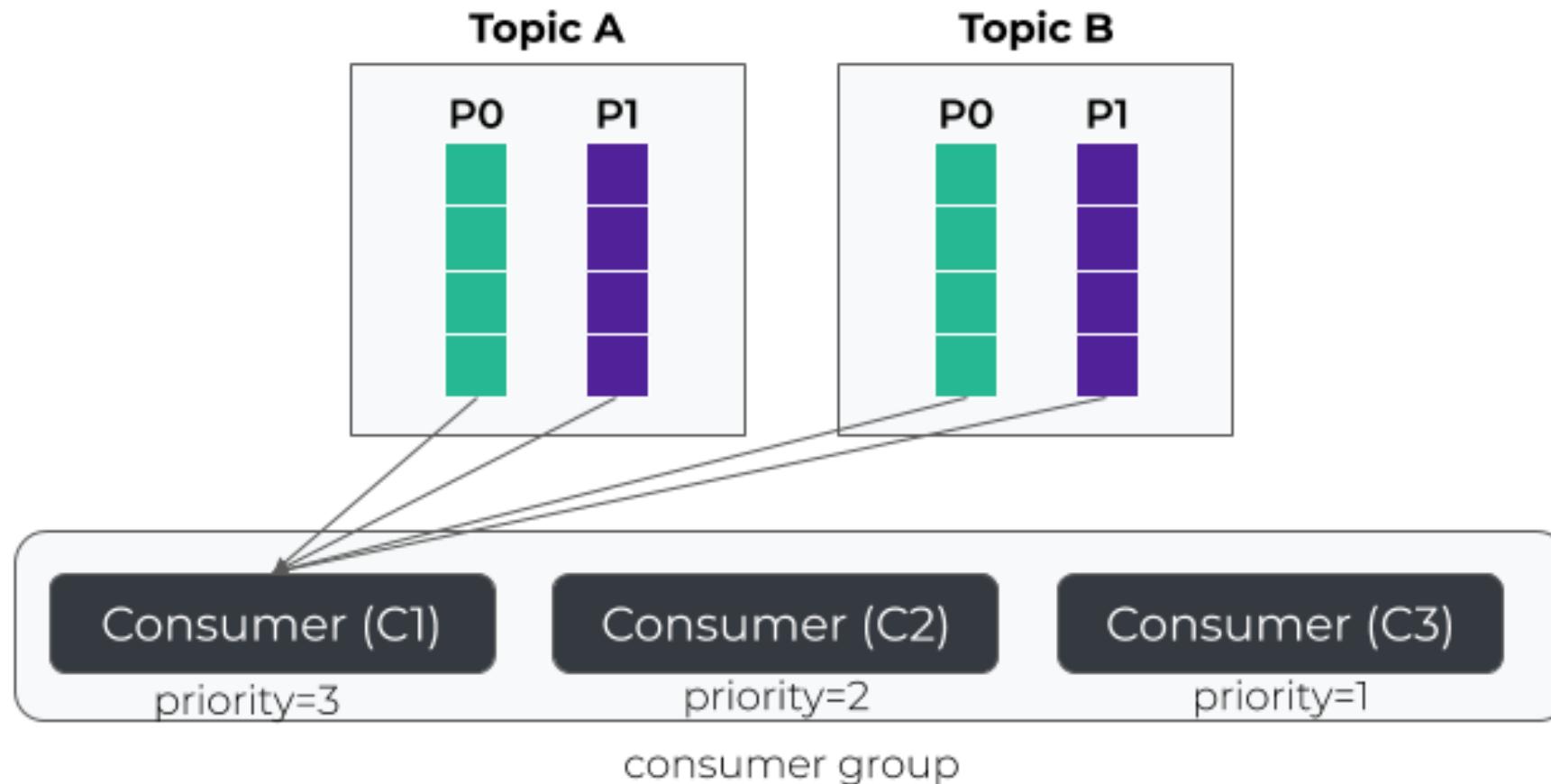


# Implementing a Custom Strategy

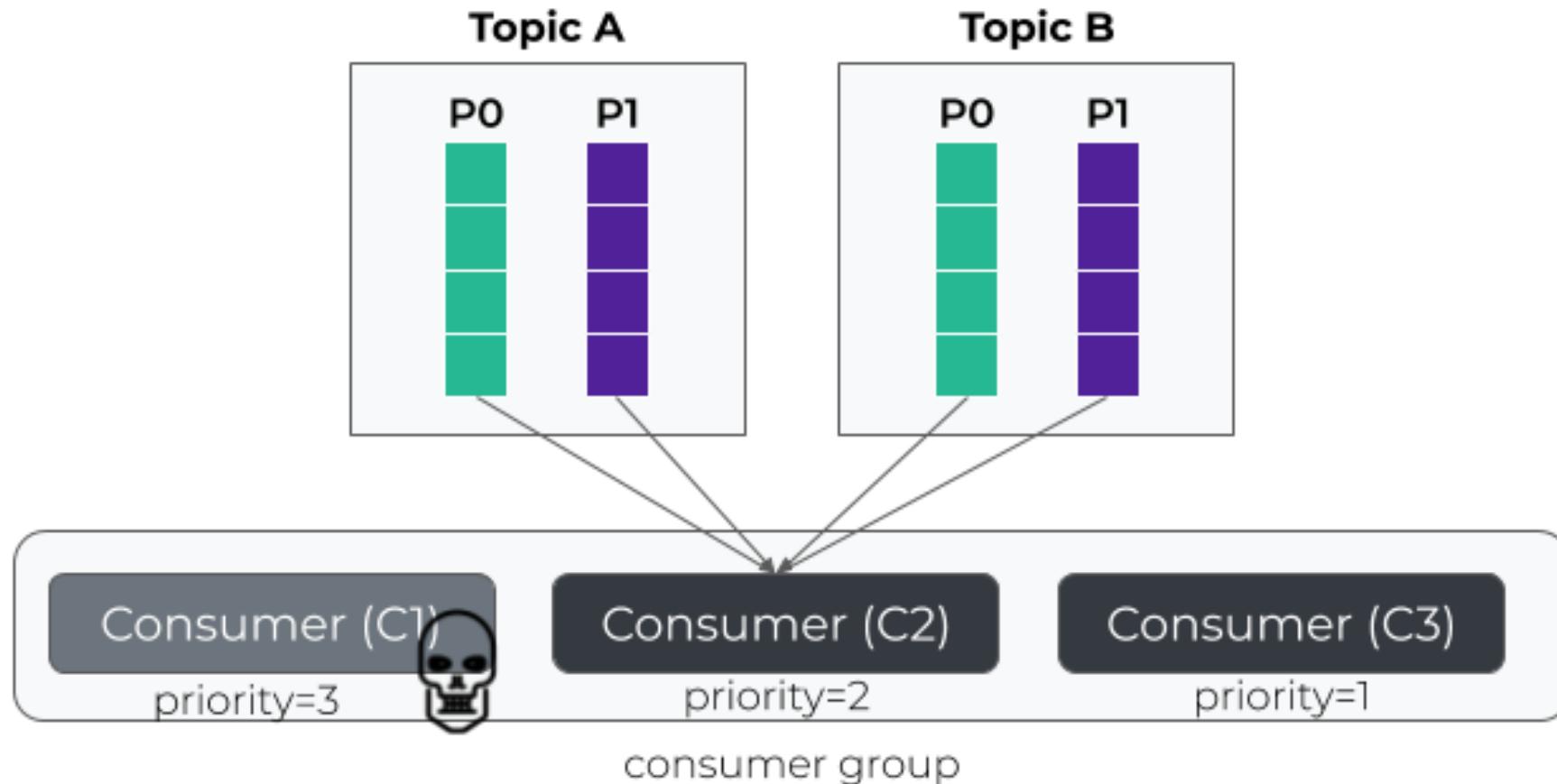
```
public interface PartitionAssignor {  
  
    Subscription subscription(Set<String> topics);  
  
    Map<String, Assignment> assign(  
        Cluster metadata,  
        Map<String, Subscription> subscriptions);  
  
    void onAssignment(Assignment assignment);  
  
    String name();  
}
```



# Simple Failover strategy



# Simple Failover strategy – C1 fails



Assignment  $\rightarrow$  C2 = {A-0, A-1, B-0, B-1}, C3 = {}



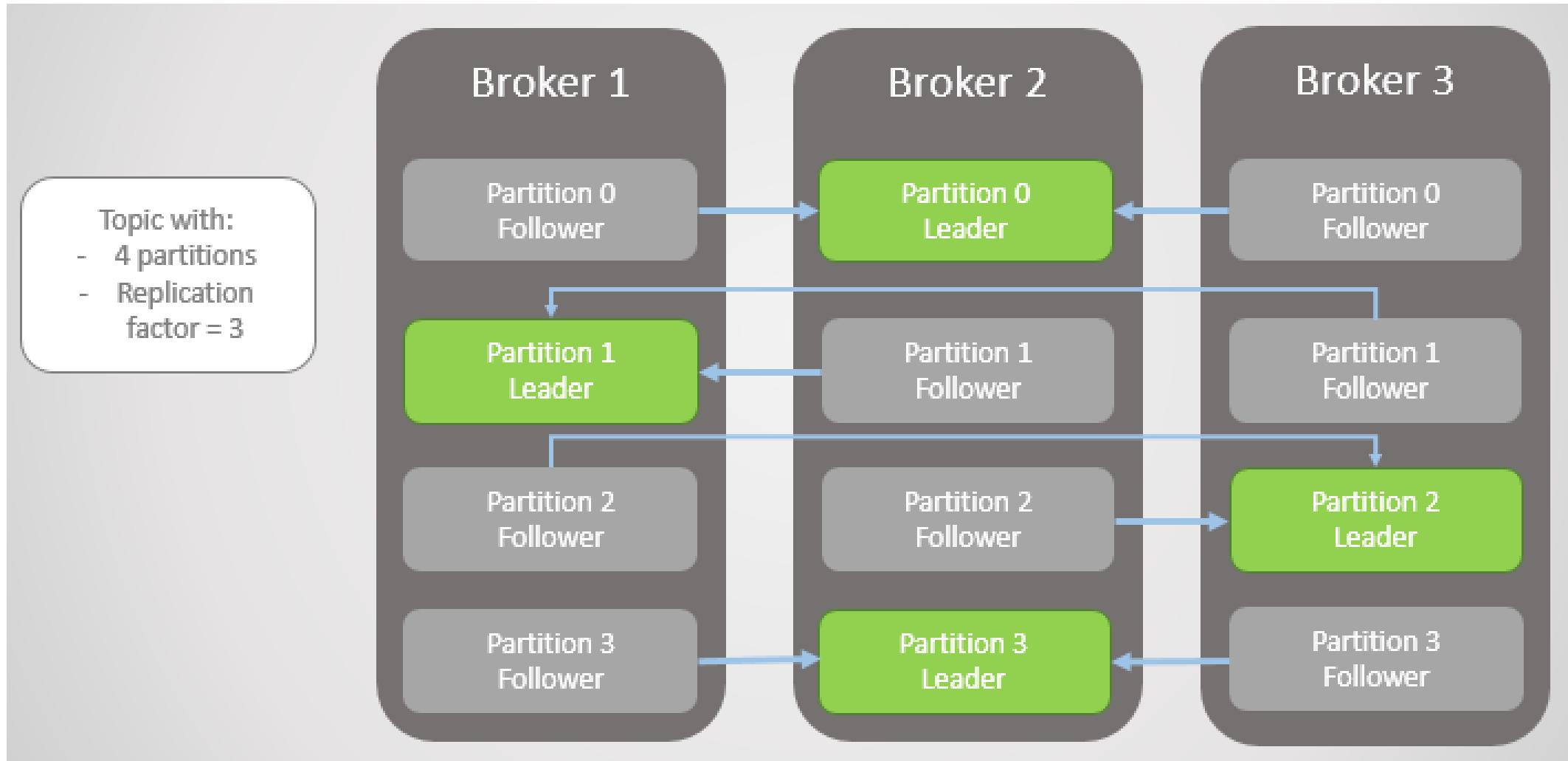


شرکت خدمات انفورماتیک  
informatics services corporation

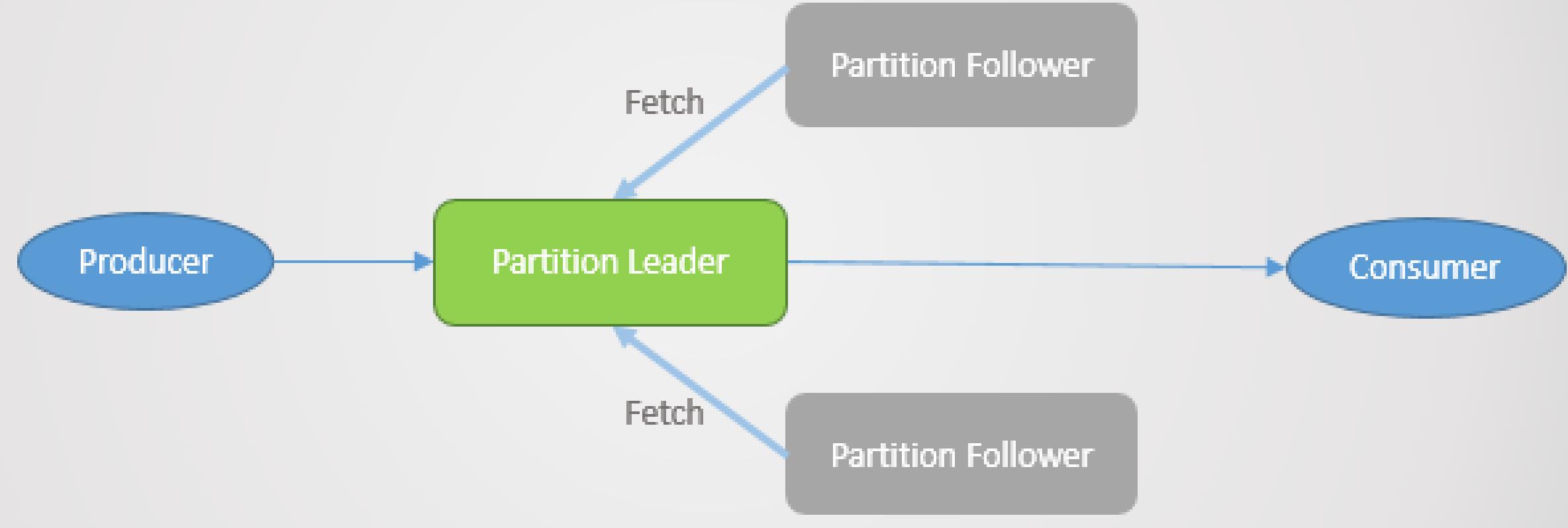
## 9 - Clustering - Replication, Leaders and Followers



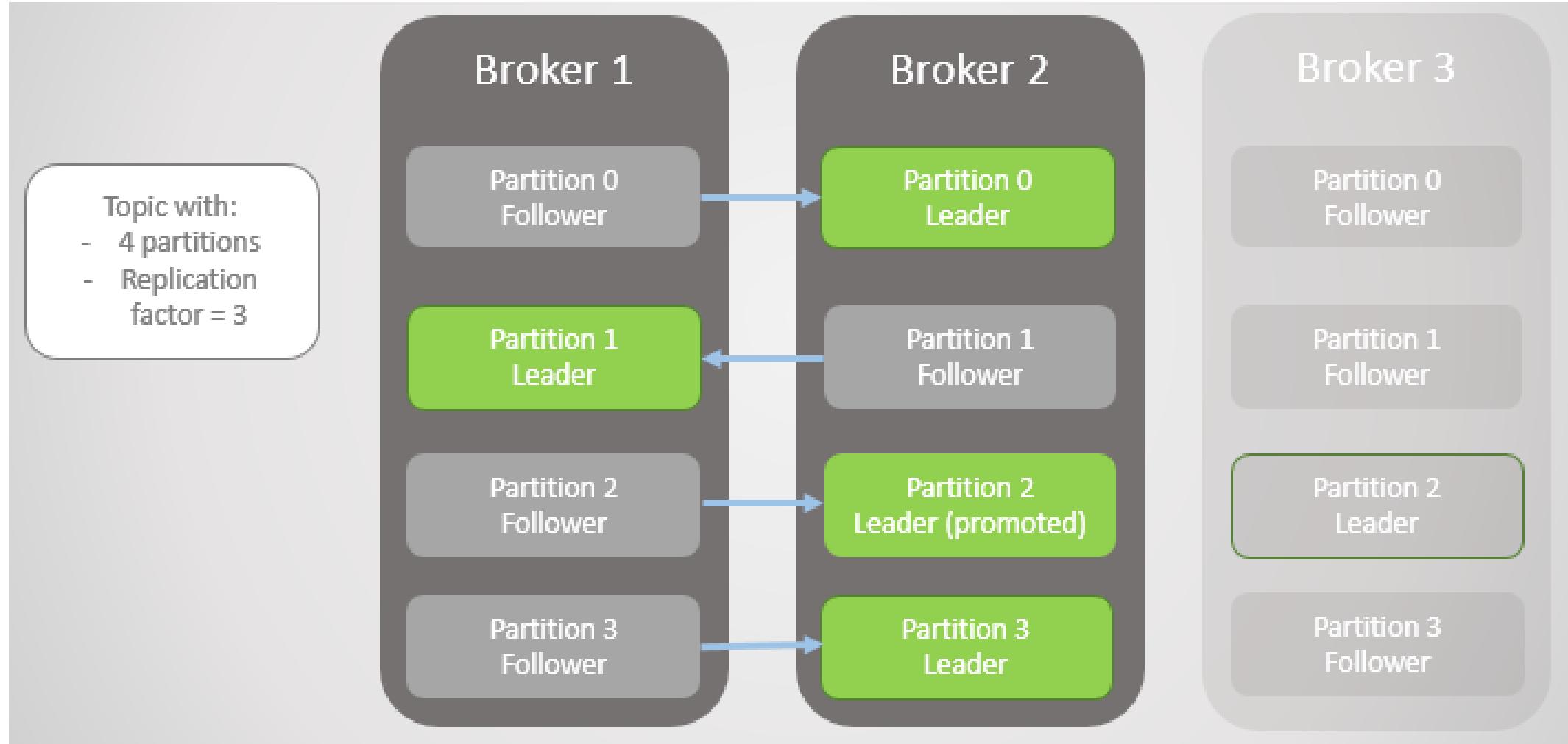
# Four partitions distributed across three brokers



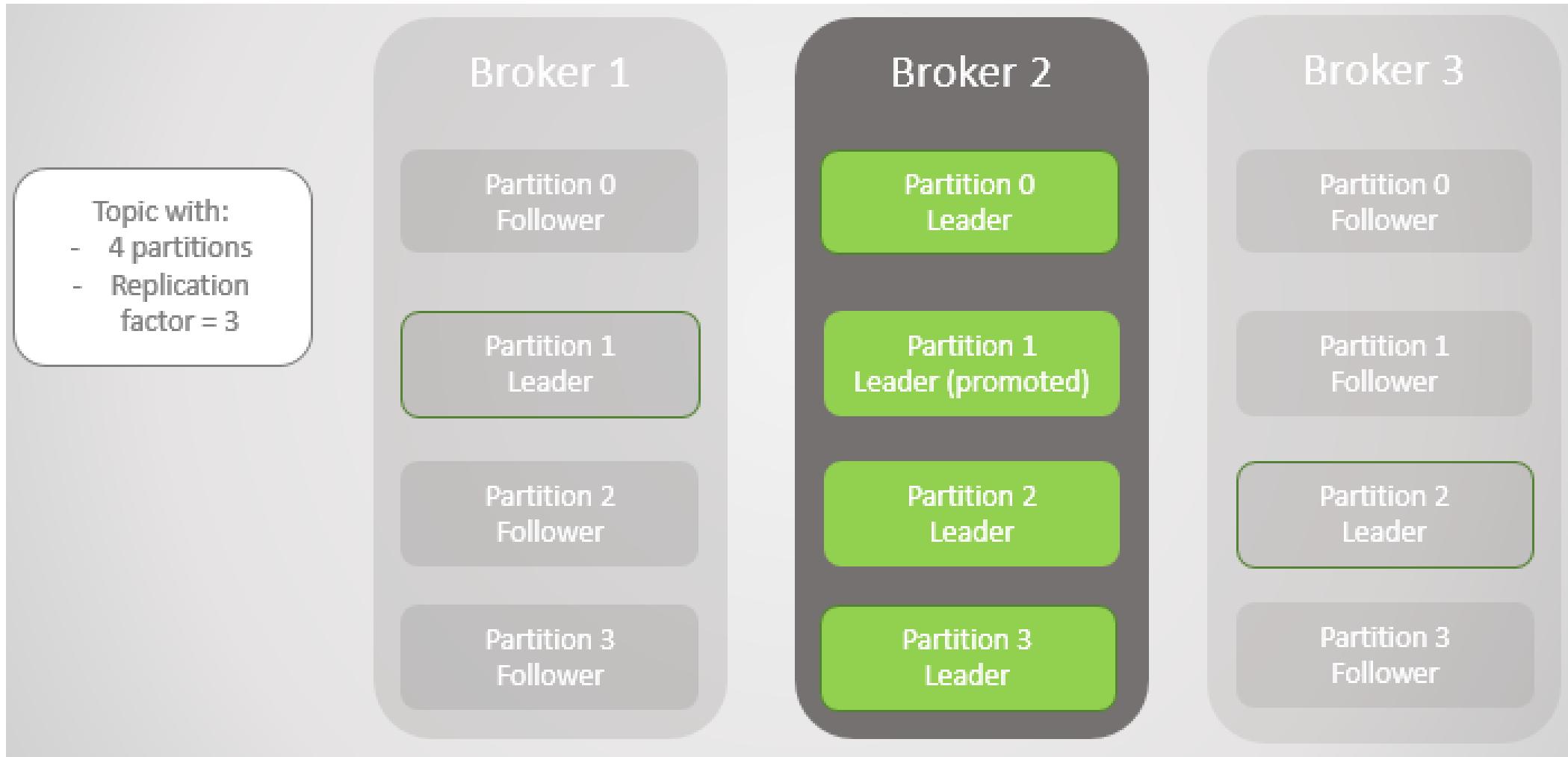
# Producer, Consumer And Leader



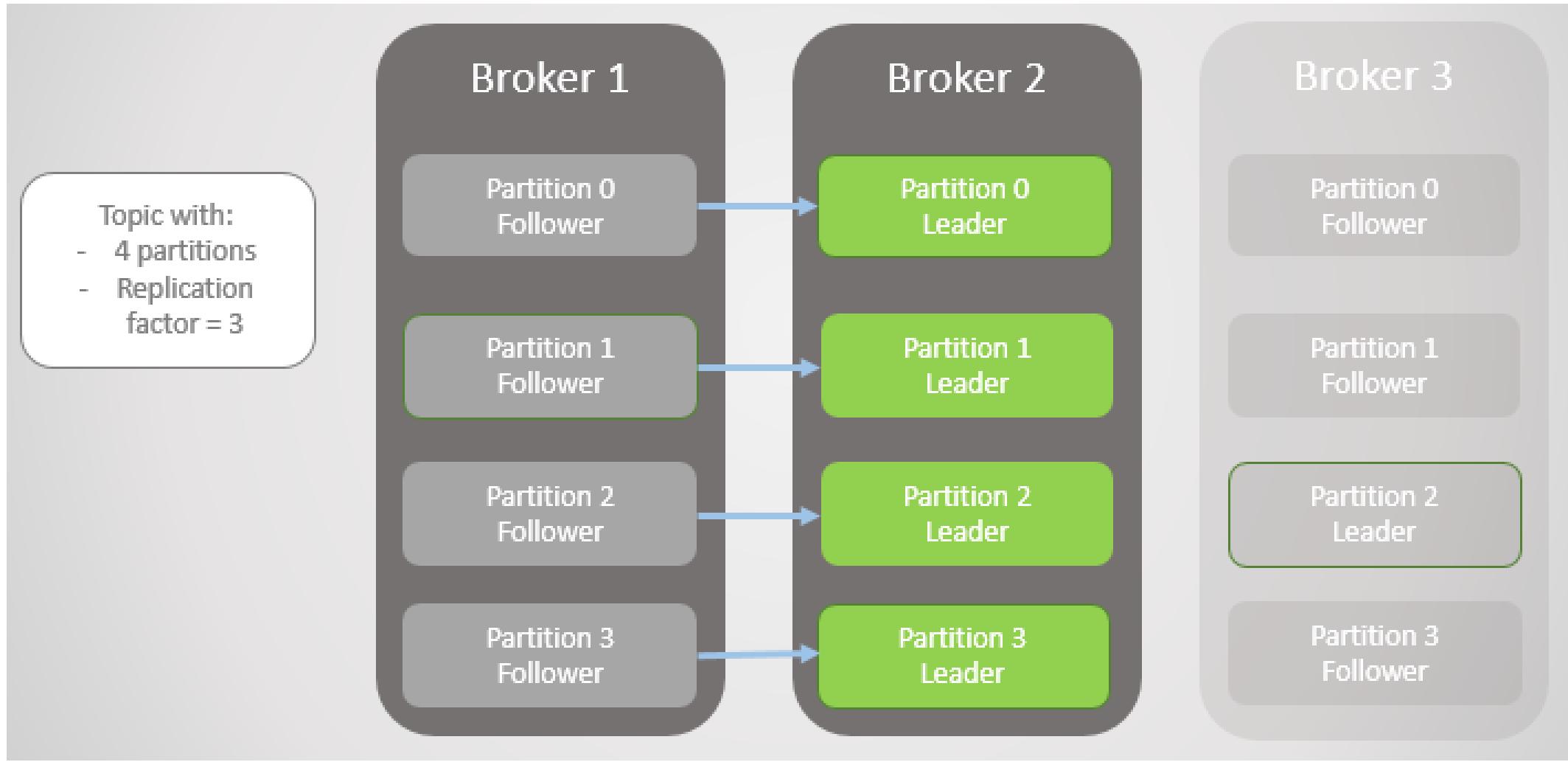
# Partition Fail-Over - Broker 3 dies



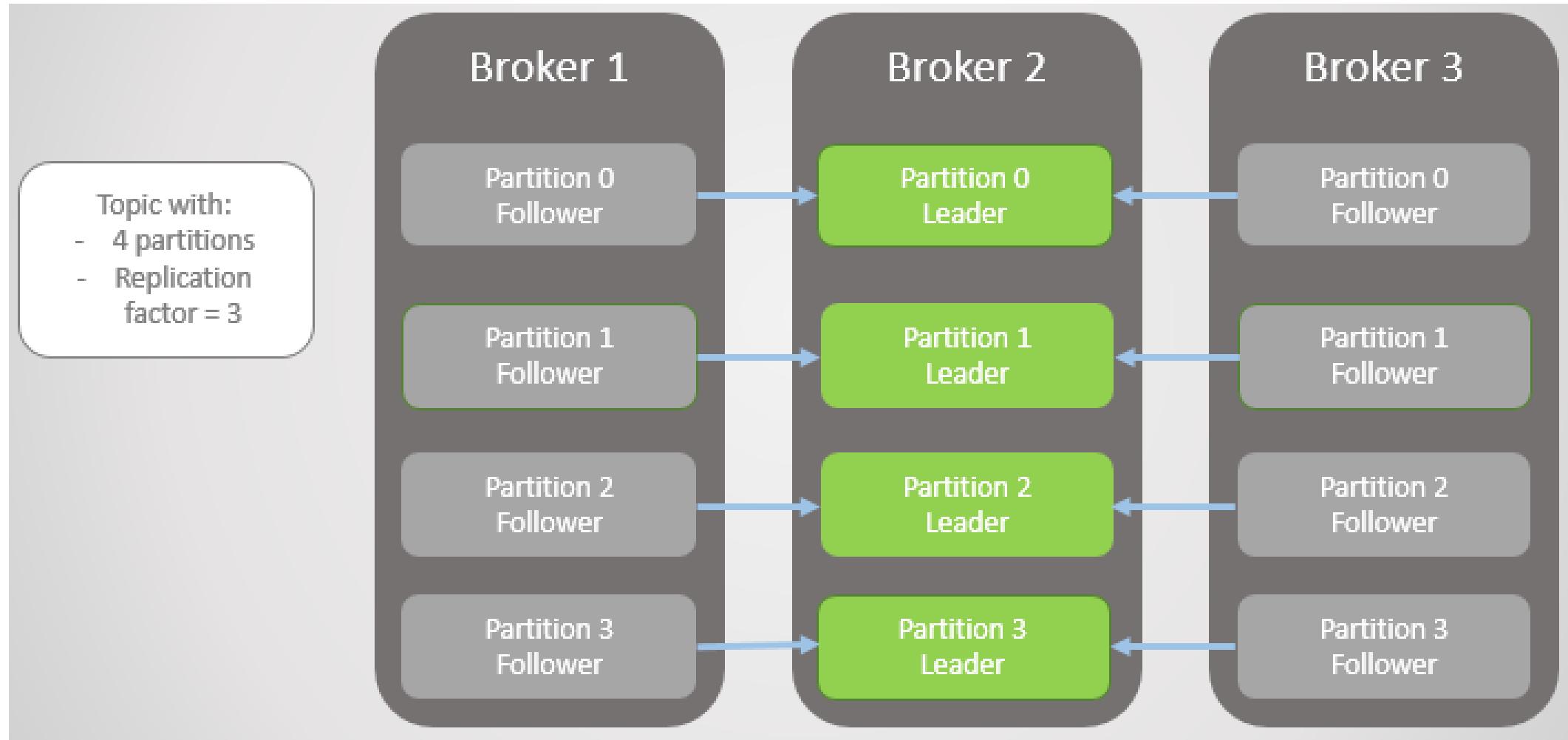
# Partition Fail-Over - Broker 1 dies too



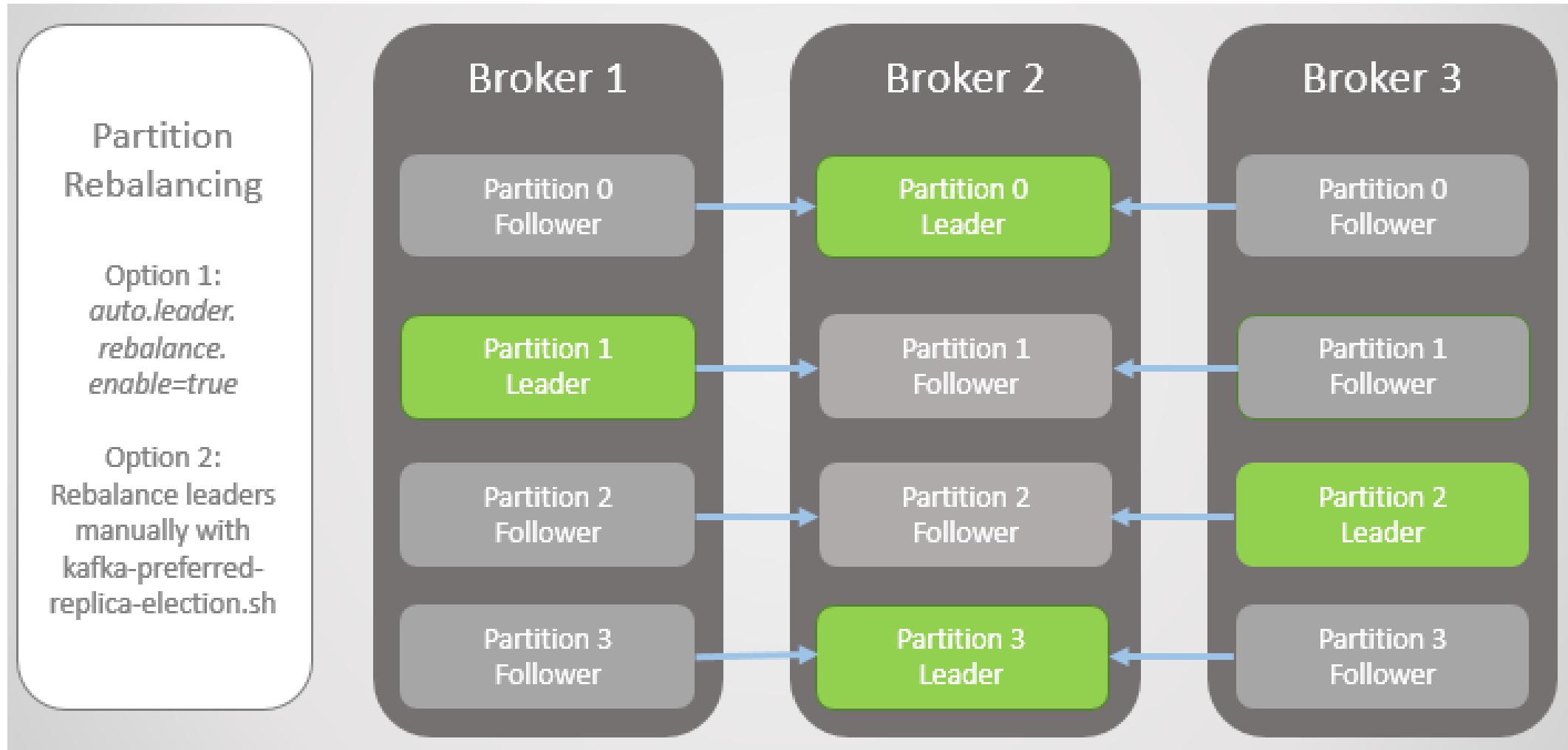
# Leaders remain on broker 2



# Unbalanced leaders after recovery of broker 1 and 3



# Replica leaders rebalanced





شرکت خدمات انفورماتیک  
informatics services corporation

## 10 - Broker – Fail Detection - Imbalance



# Controller Broker

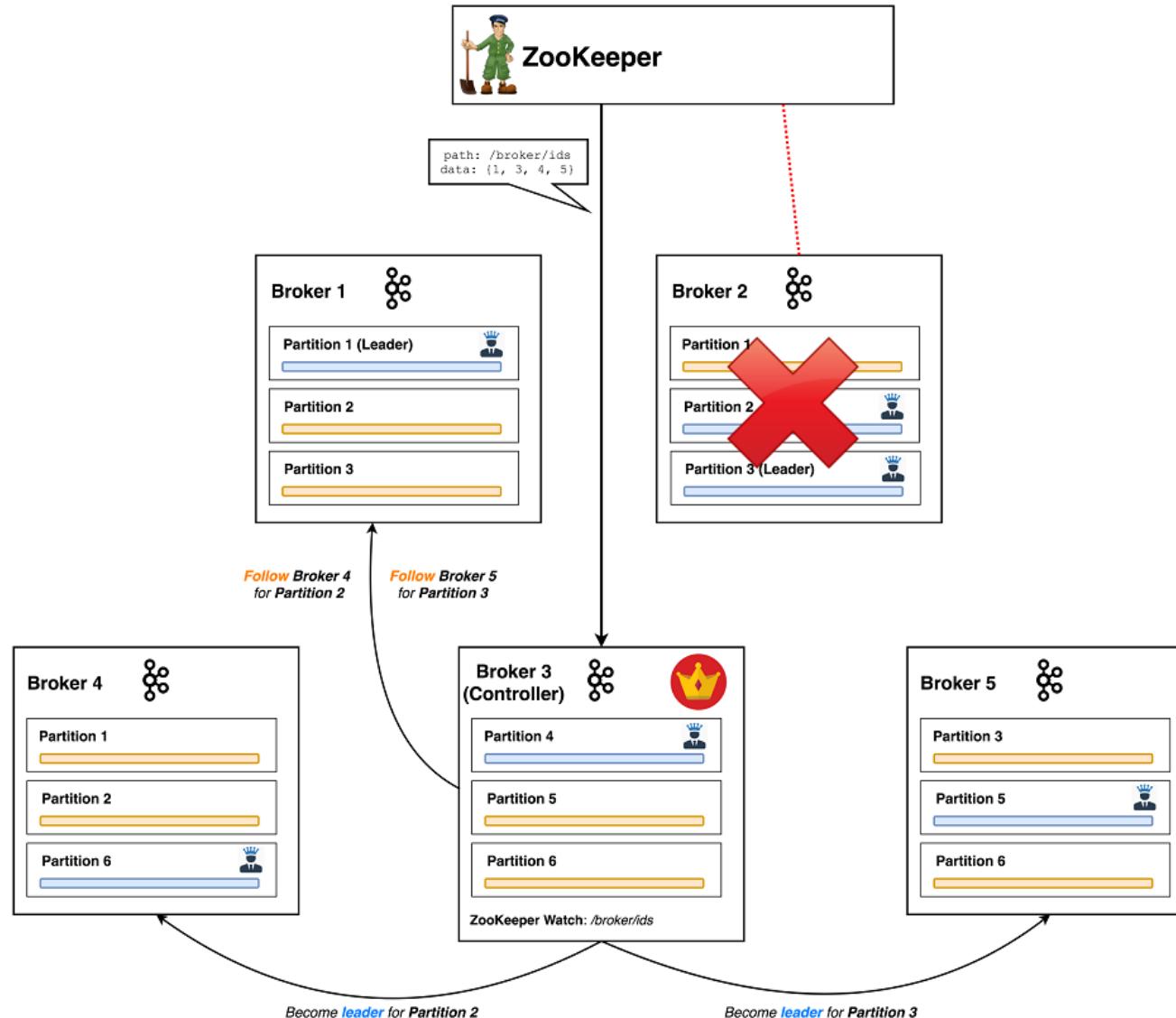
- \* A distributed system must be coordinated.
  - \* Event happens & React in an organized way
- \* It is a normal broker that simply has additional responsibility
- \* Additional responsibility
  - \* Keeping track of nodes in the cluster
  - \* Handling nodes that leave, join or fail
- \* There is always exactly one controller broker in a Kafka cluster.



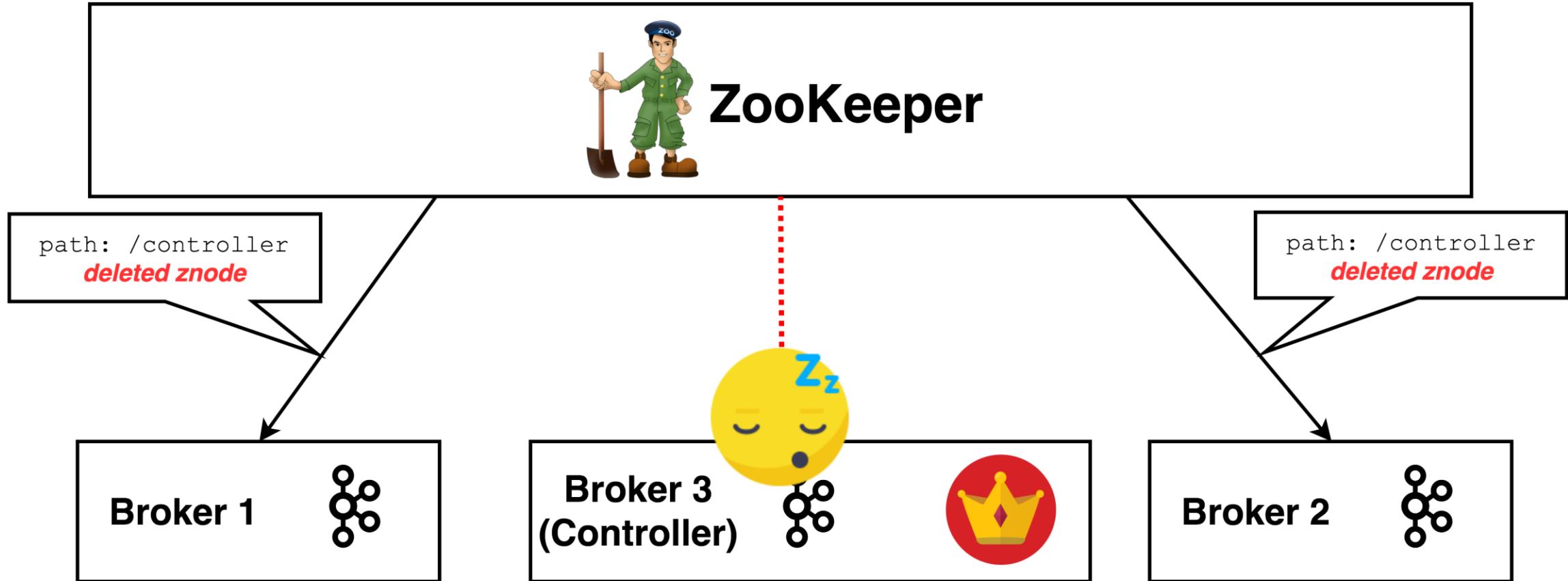
شرکت خدمات اینفورماتیک  
informatics services corporation



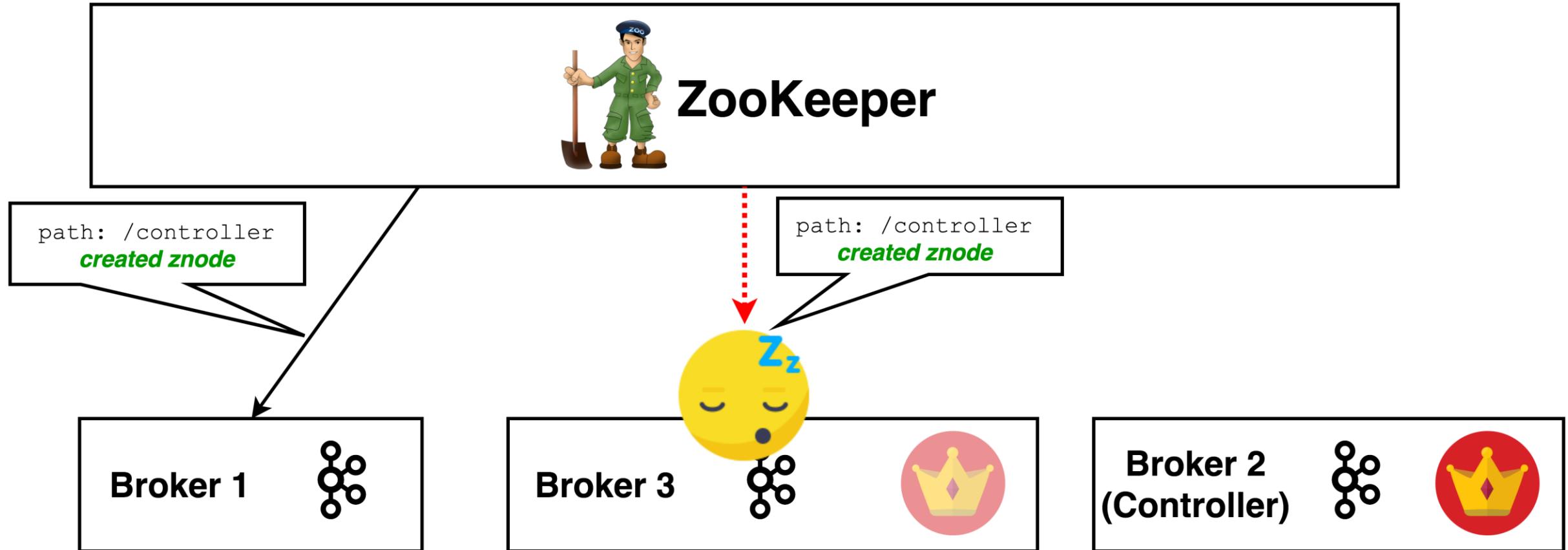
# Handle a node leaving the cluster



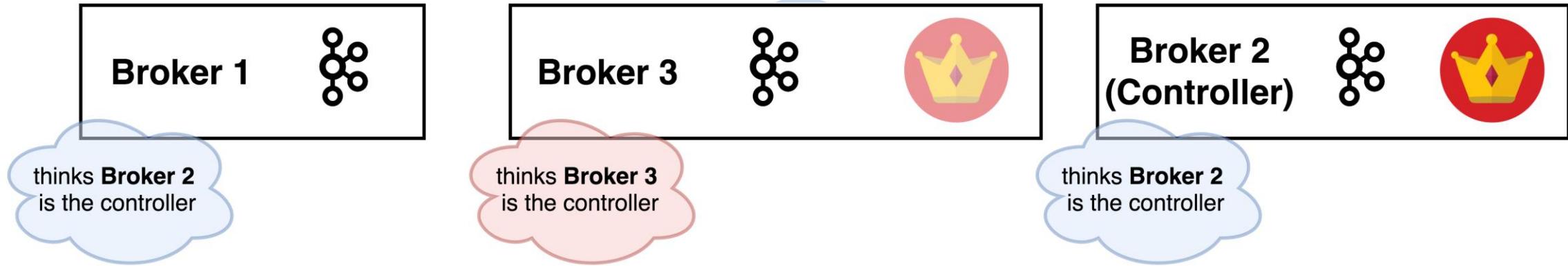
# Split Brain



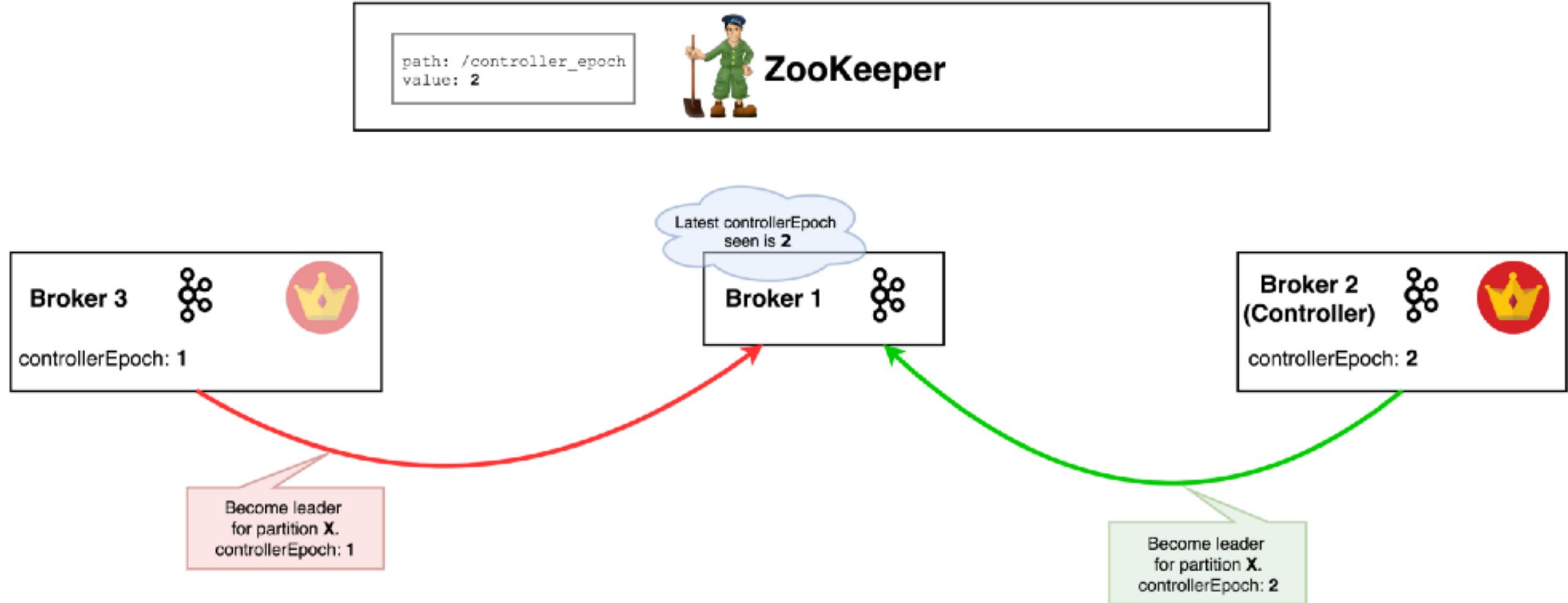
# Split Brain - Broker 2 won the race



# Broker 3 wake up



# Epoch Number





شرکت خدمات انفورماتیک  
informatics services corporation

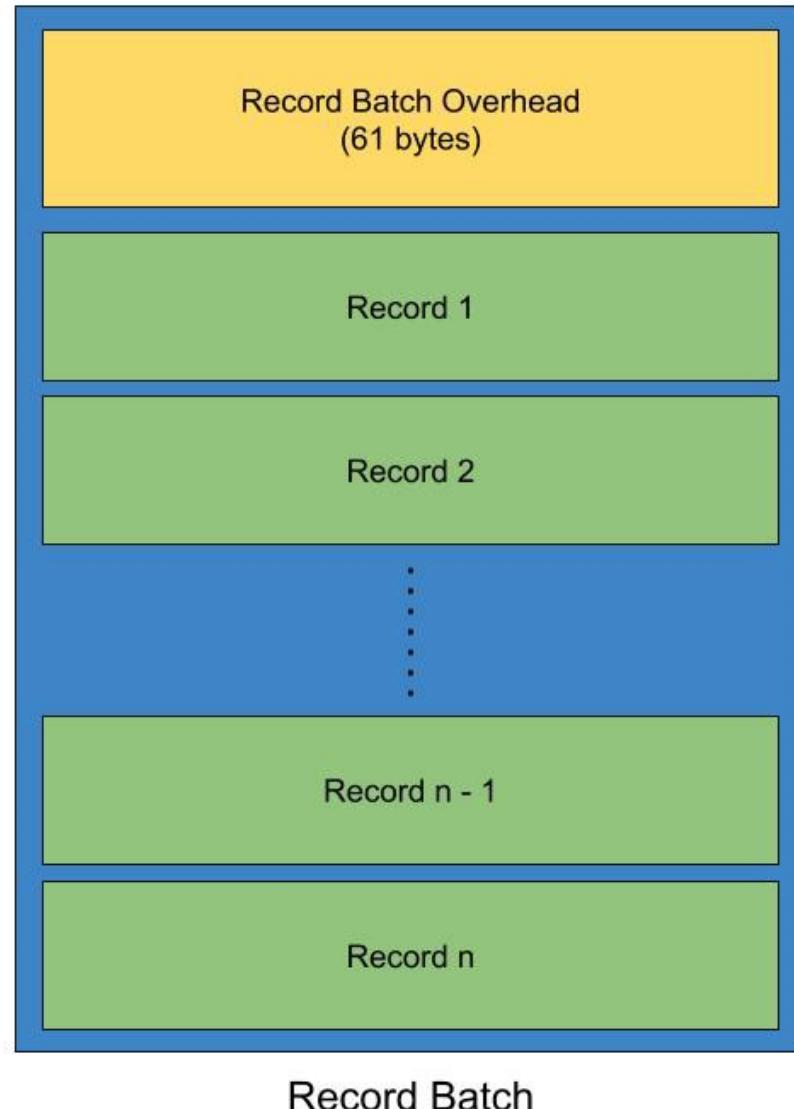
## 11 - Logs – Compression - Retention



# Why do I need to reduce message size?



# Message Format

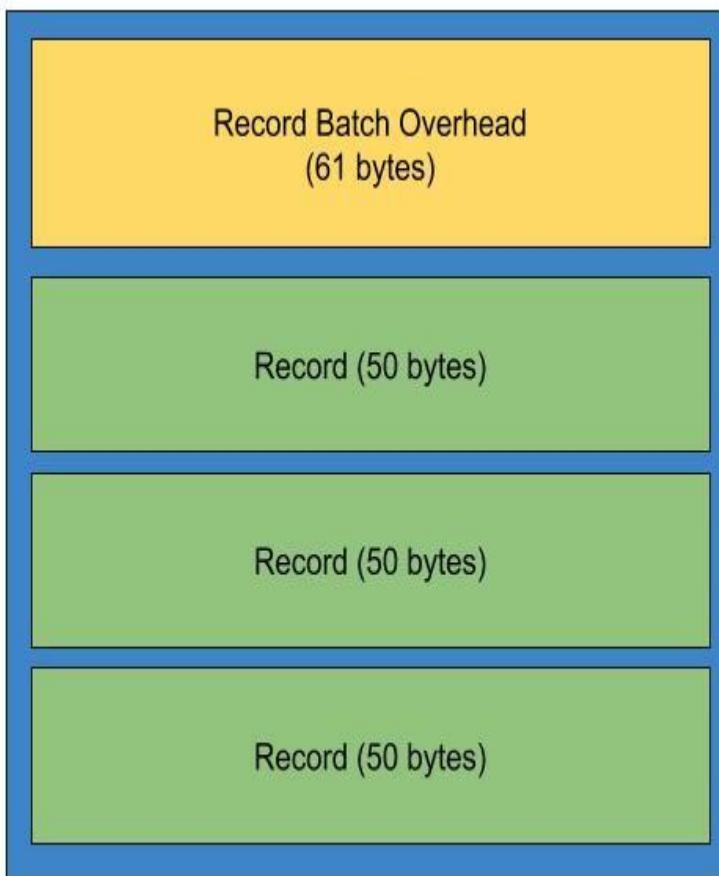




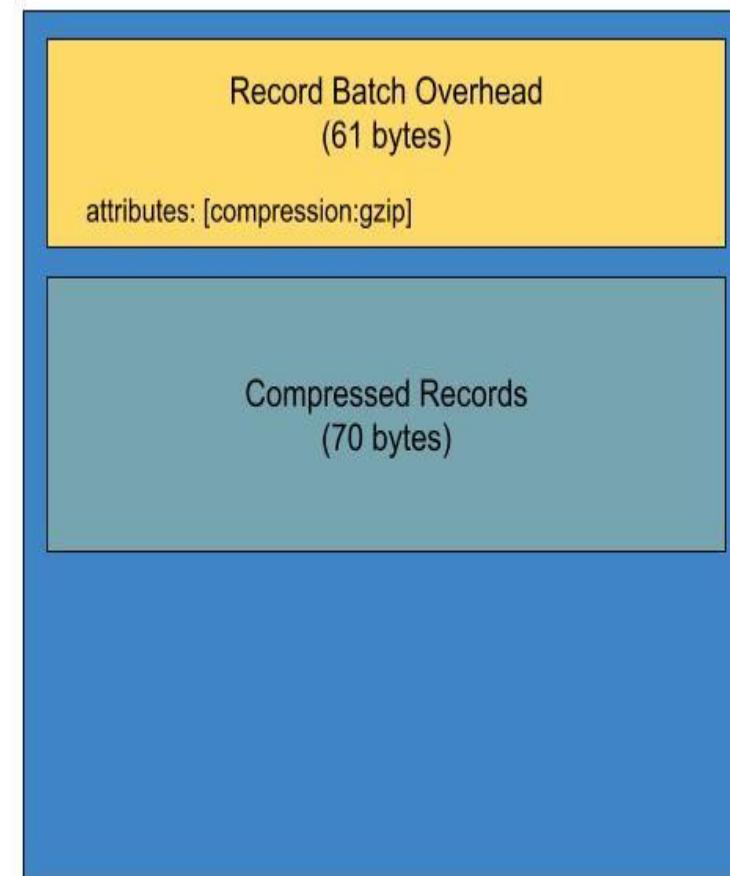
# Without Lingering VS Using Lingering



# Compression



Using lingering we need 211 bytes to store 3 records



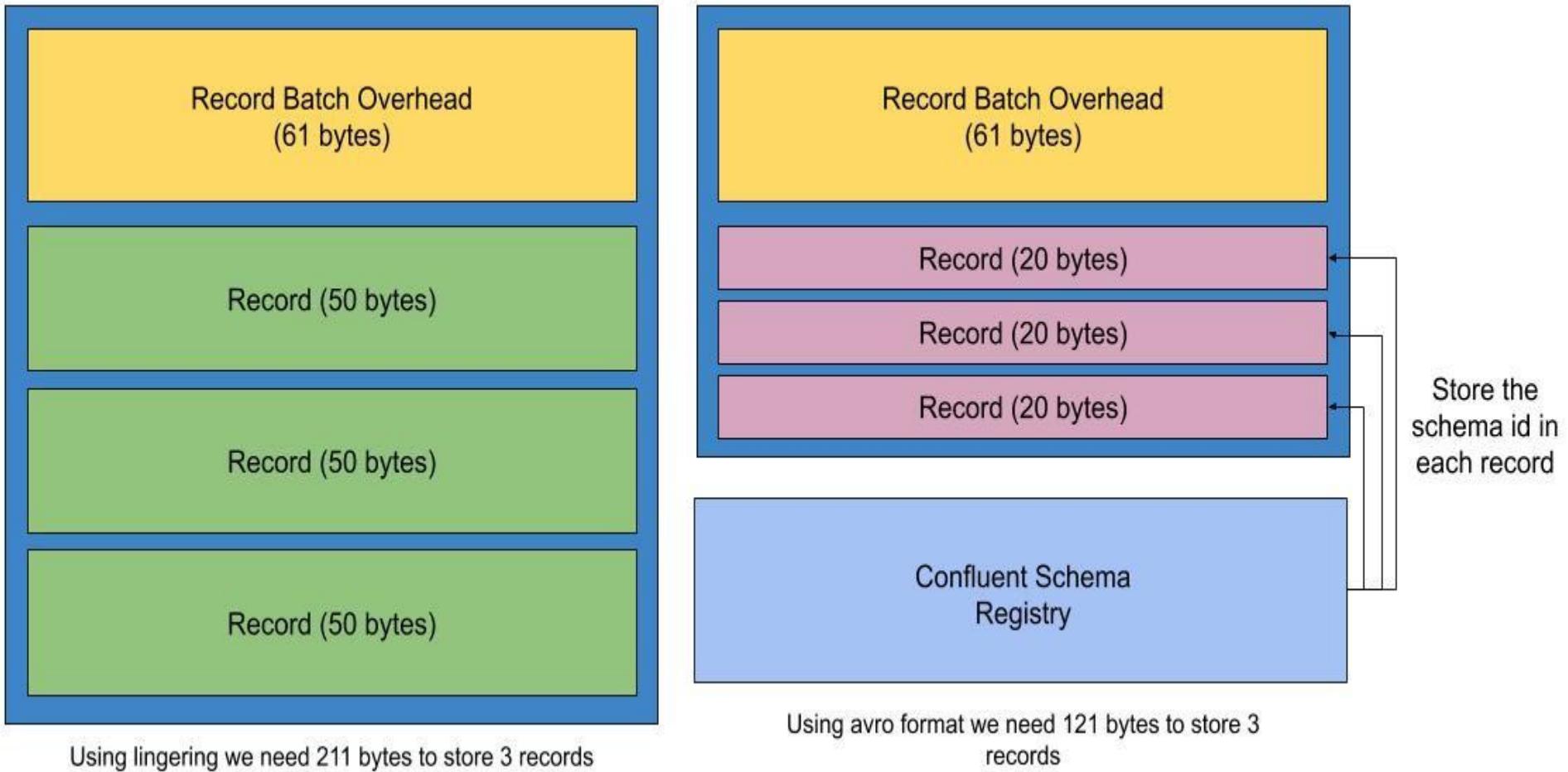
Using lingering and compression we need 131 bytes to store 3 records



# Support for Zstandard Compression (KIP-110)

	Ratio	Comp Speed MB/s	Decomp Speed MB/s
<b>zstd cli</b>	3.14	100.1	459.40
<b>gzip cli</b>	3.11	19.66	125.55





# Conclusion

- ❖ Lingering needs you wait a bit more to gather more records.
- ❖ Compression uses more CPU usage but will reduce the amount of IO.
- ❖ Using Avro impose
  - ❖ Dependency on clients (consumer and producer)
  - ❖ have Confluent Schema Registry to keep the schema of the records.





# Time Based Retention

Config Name	Description	Default Value	Valid Values
log.retention.ms	The number of milliseconds to keep a log file before deleting it (in milliseconds), If not set, the value in log.retention.minutes is used	null	any long value
log.retention.minutes	The number of minutes to keep a log file before deleting it (in minutes), secondary to log.retention.ms property. If not set, the value in log.retention.hours is used	null	any long value
log.retention.hours	The number of hours to keep a log file before deleting it (in hours), tertiary to log.retention.ms property	168	any long value

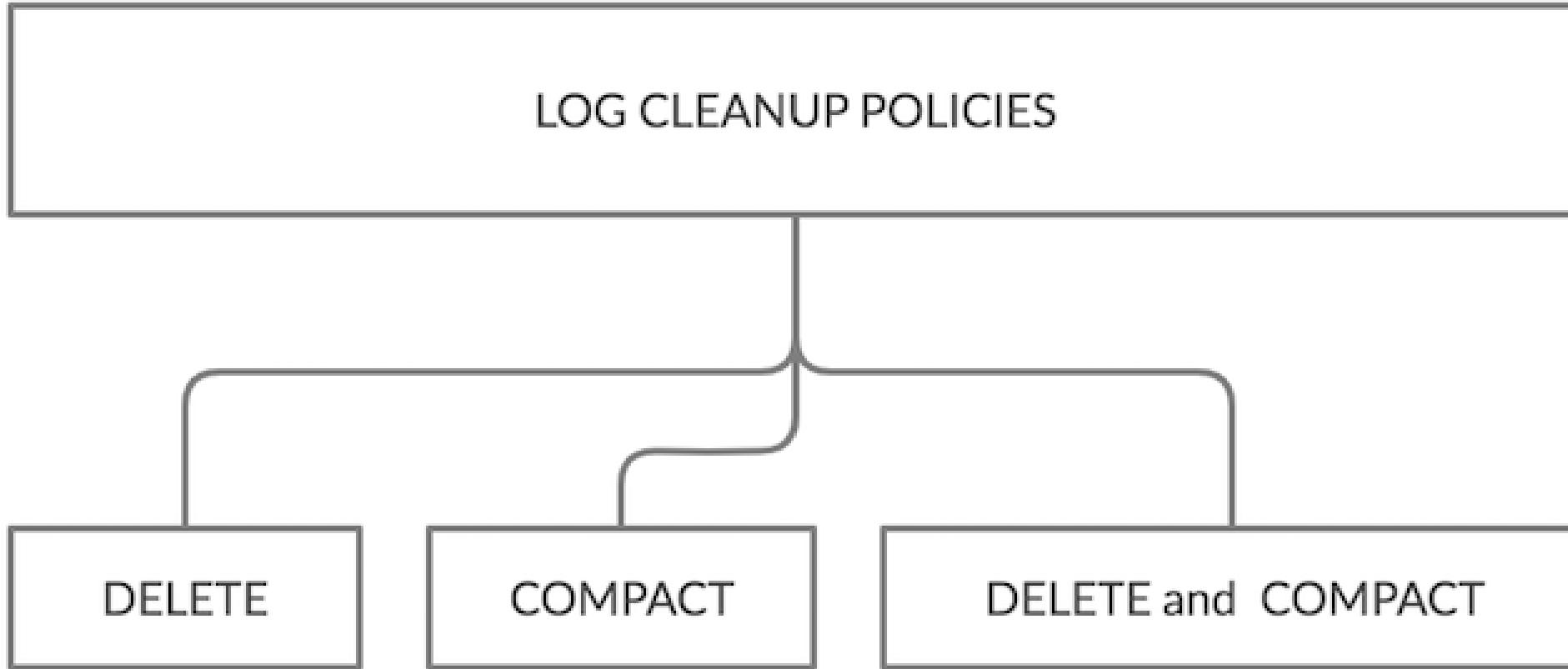


# Size Based Retention

Config Name	Description	Default Value	Valid Values
log.retention.bytes	The maximum size of the log before deleting it	-1	any long value



# Log Cleanup Policies



# Compact Policy

Offset	0	1	2	3	4	5	6	7	8	9	10
Key	K1	K2	K1	K1	K3	K2	K4	K5	K5	K2	K6
Value	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11

Compaction

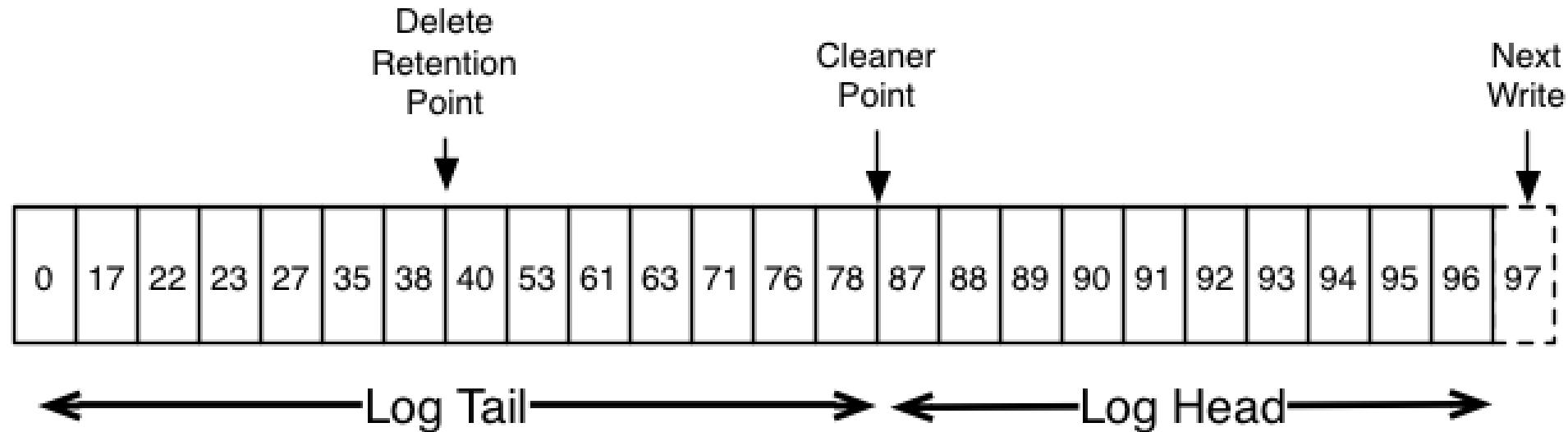
Keys	3	4	6	8	9	10
Values	K1	K3	K4	K5	K2	K6
Values	V4	V5	V7	V9	V10	V11

Log After Compaction

شرکت خدمات اینفورماتیک



# Log Cleaner



- 1. Does Log Cleaner impacts Read performance?**
  
- 2. Does offset of message gets changed after compaction?**
  
- 3. Does Log Cleaner also deletes Tombstone messages?**
  
- 4. Does order of messages gets changed after compaction?**



شرکت خدمات اینفورماتیک  
informatics services corporation





## 12 - Security – TLS – Authentication - Authorization



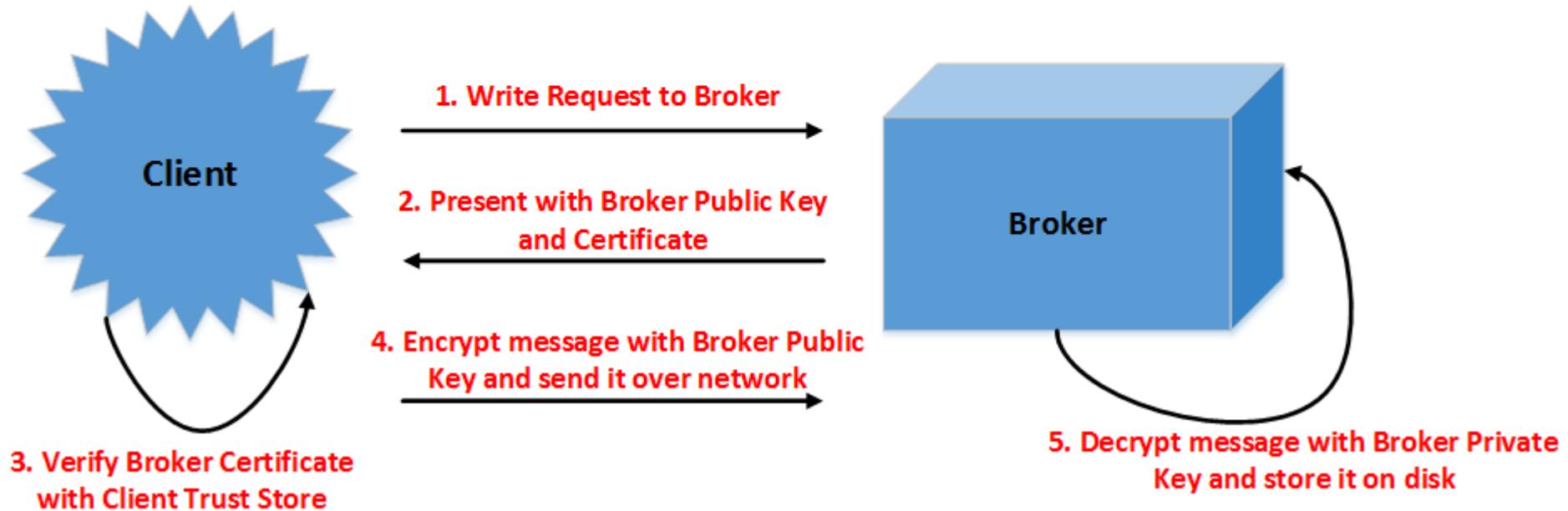
# The need for security



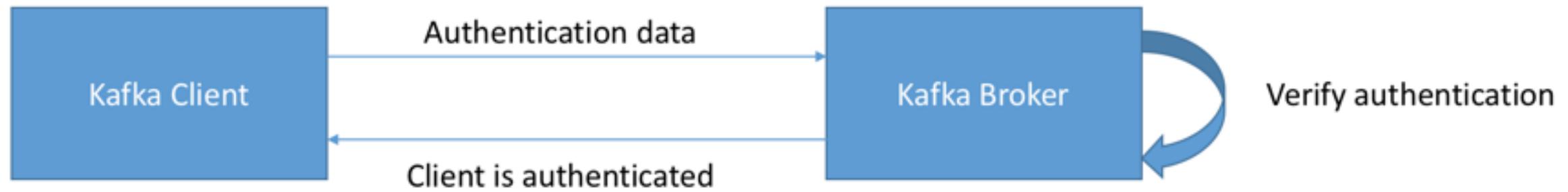
# Problems Security is solving



# Encryption



# Authentication



- “User alice can view topic finance”
- “User bob cannot view topic trucks”





با سپاس



شرکت خدمات انفورماتیک  
informatics services corporation

زمستان - ۱۳۹۹