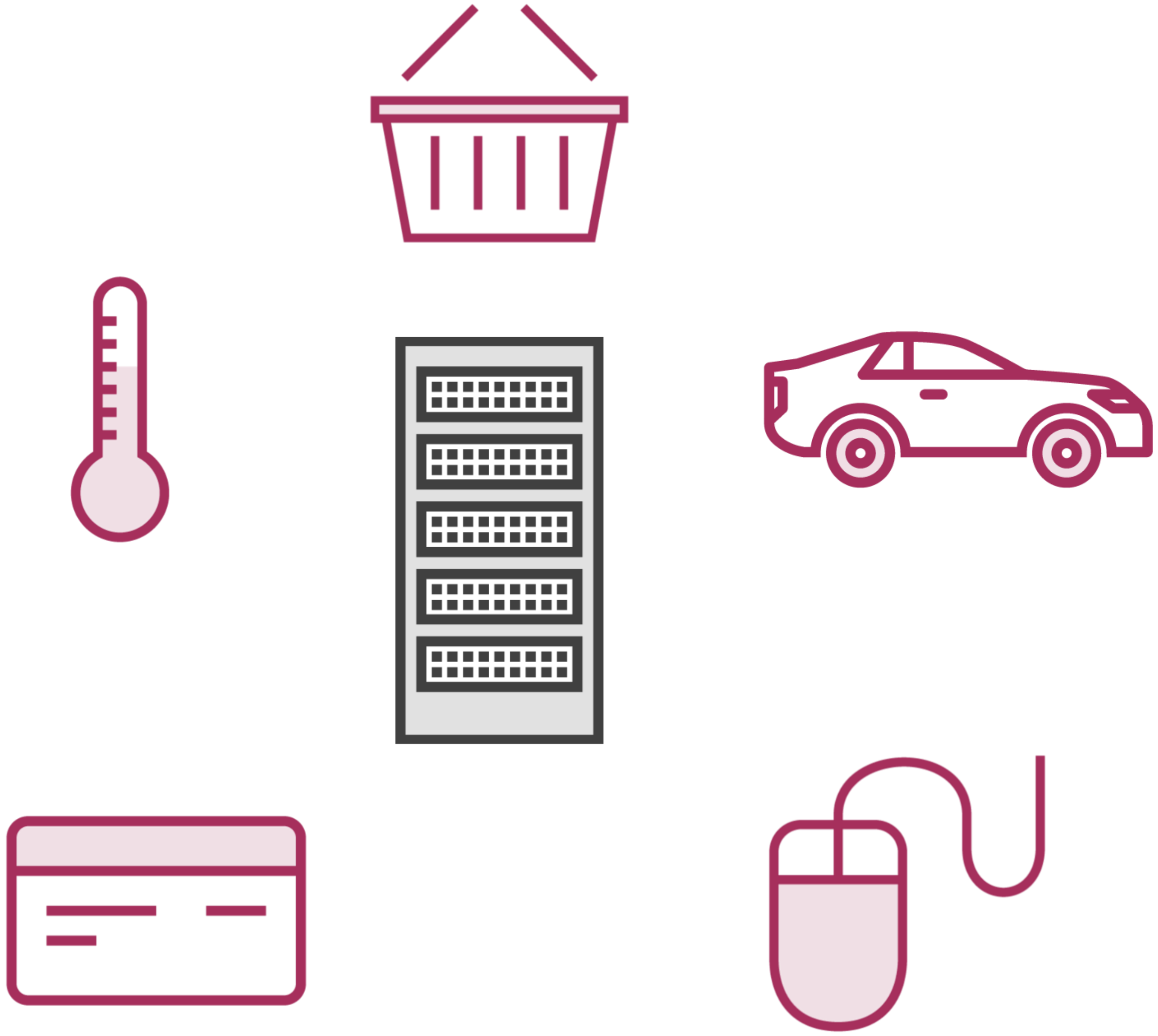
Fault Tolerance and High Availability

Paweł Kordek
DATA ENGINEER

@pawel_kordek   https://kordek.github.io



Using a single broker means accepting:

- Message loss
- Downtime

Desired Properties

Fault Tolerance

High Availability

Consistency

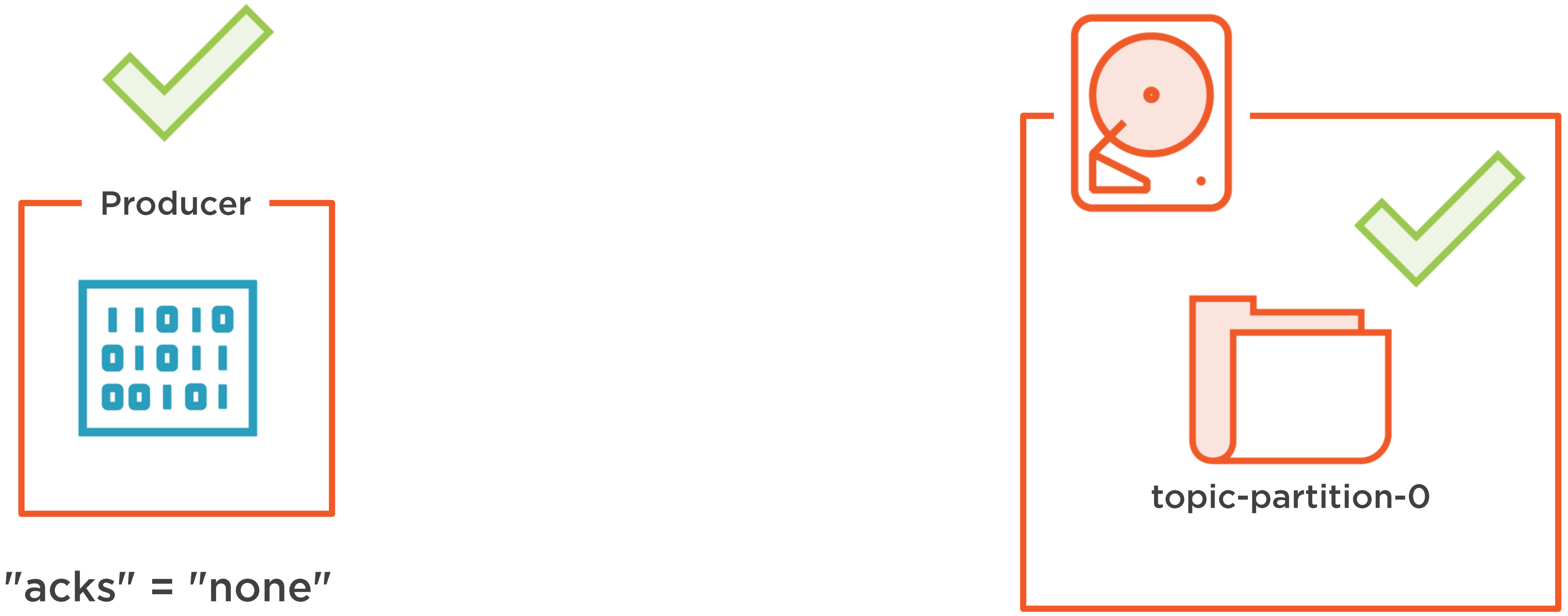
ZooKeeper

Source of truth about cluster members

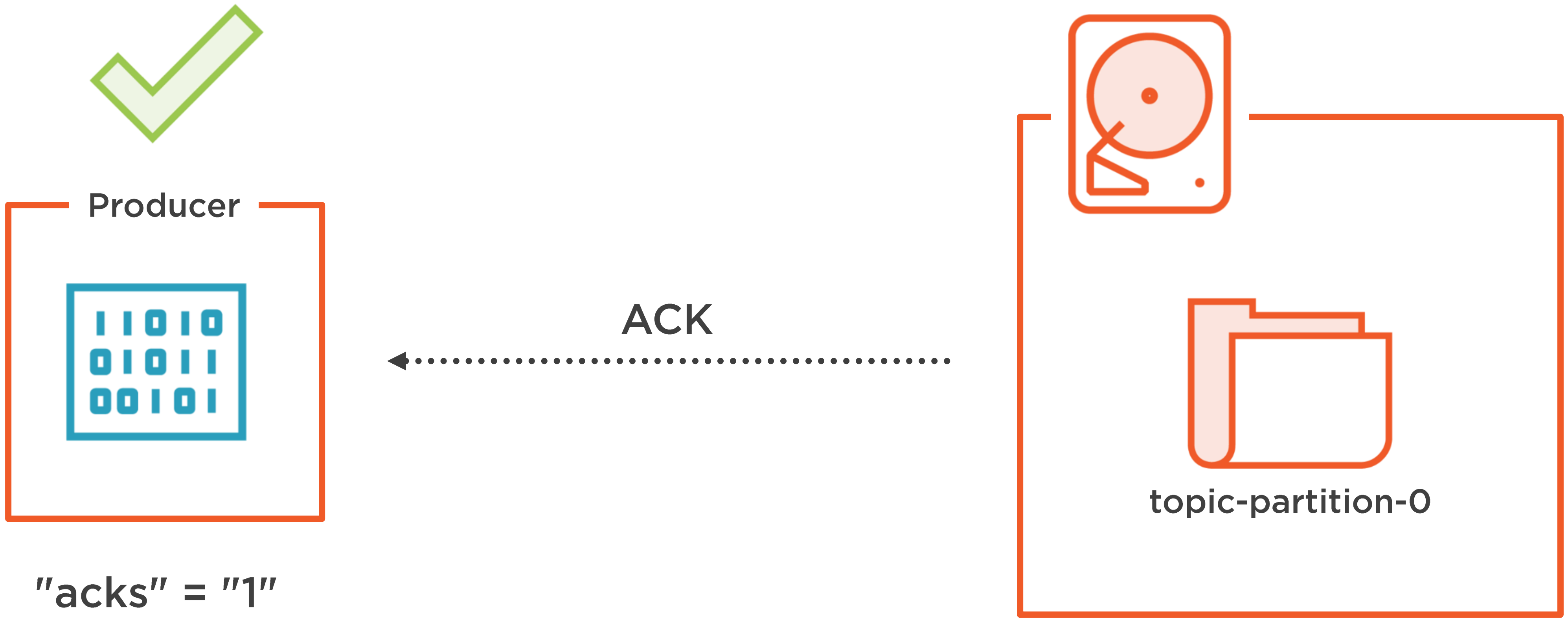
Point of contact for new brokers

Single ensemble can be shared with other applications

Message Persistence



"acks" = "none"

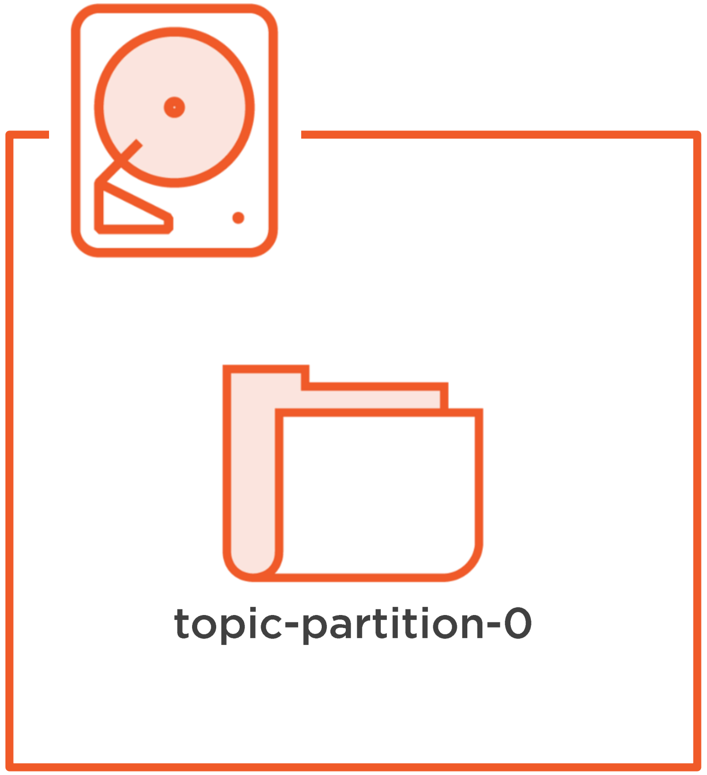


"acks" = "1"

Producer Delivery Guarantees

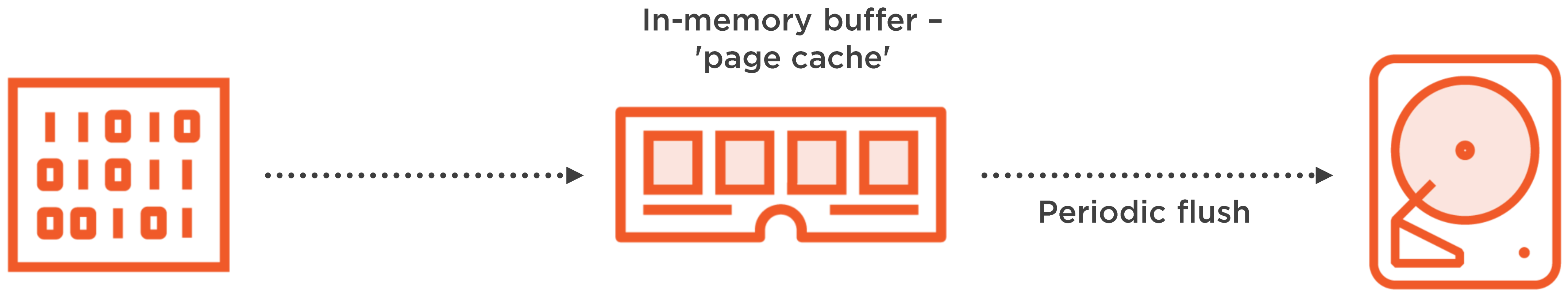
```
Properties props = new Properties();

. . .

KafkaProducer<String, String> = new KafkaProducer<>(props);

ProducerRecord<String, String> r = new ProducerRecord("topic", "key", "value");

RecordMetadata rm = producer.send(r).get();
```

When it returns depends on 'acks' value



Only after a successful write, data is committed and made available to consumers.

Writing to Disk



Replication

Creating a New Topic

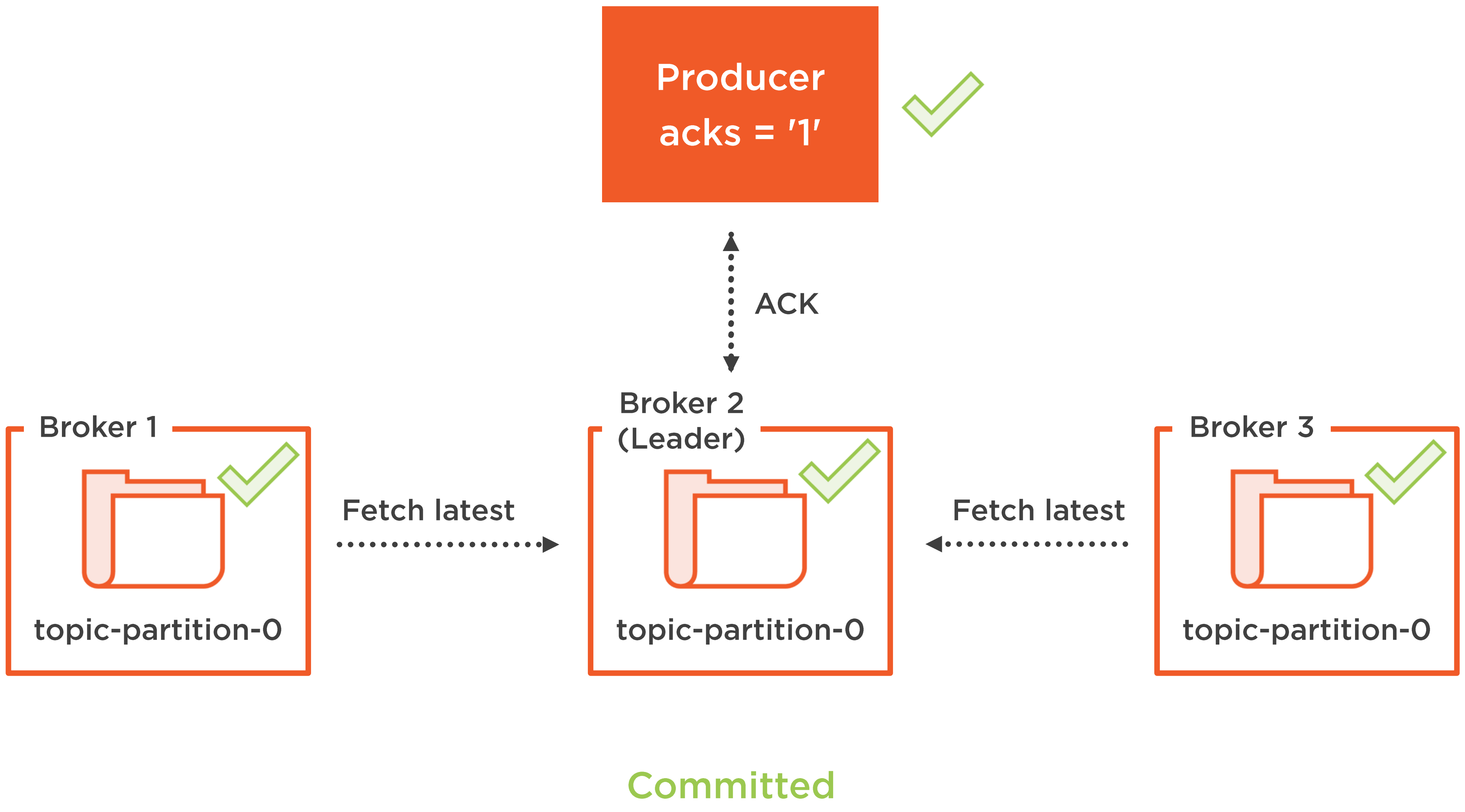
```
Admin admin = Admin.create(
        Map.of("bootstrap.servers", "localhost:9092")
);

NewTopic topic = new NewTopic("quote-feedback", 2, (short) 1)

admin.createTopics(List.of(topic)).get()
```

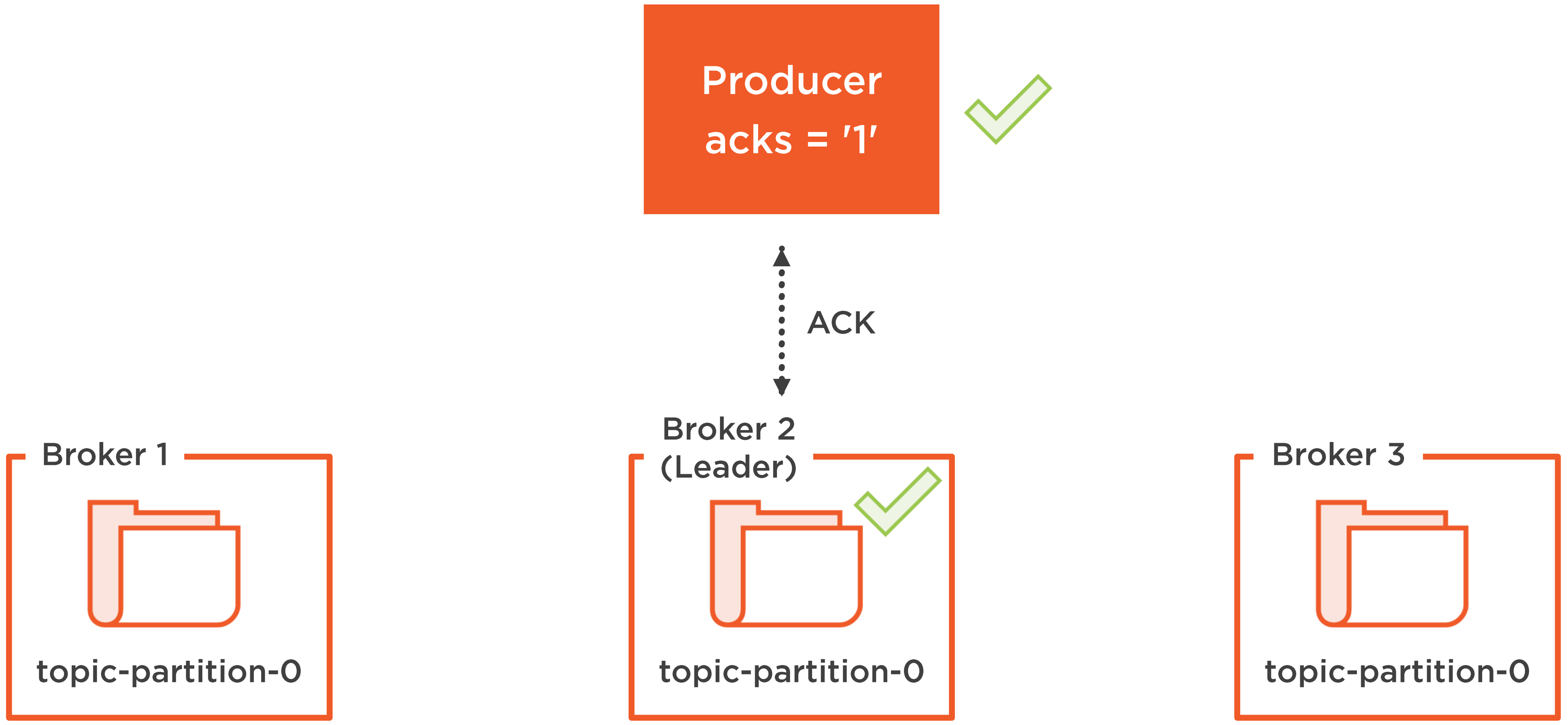
Replication factor

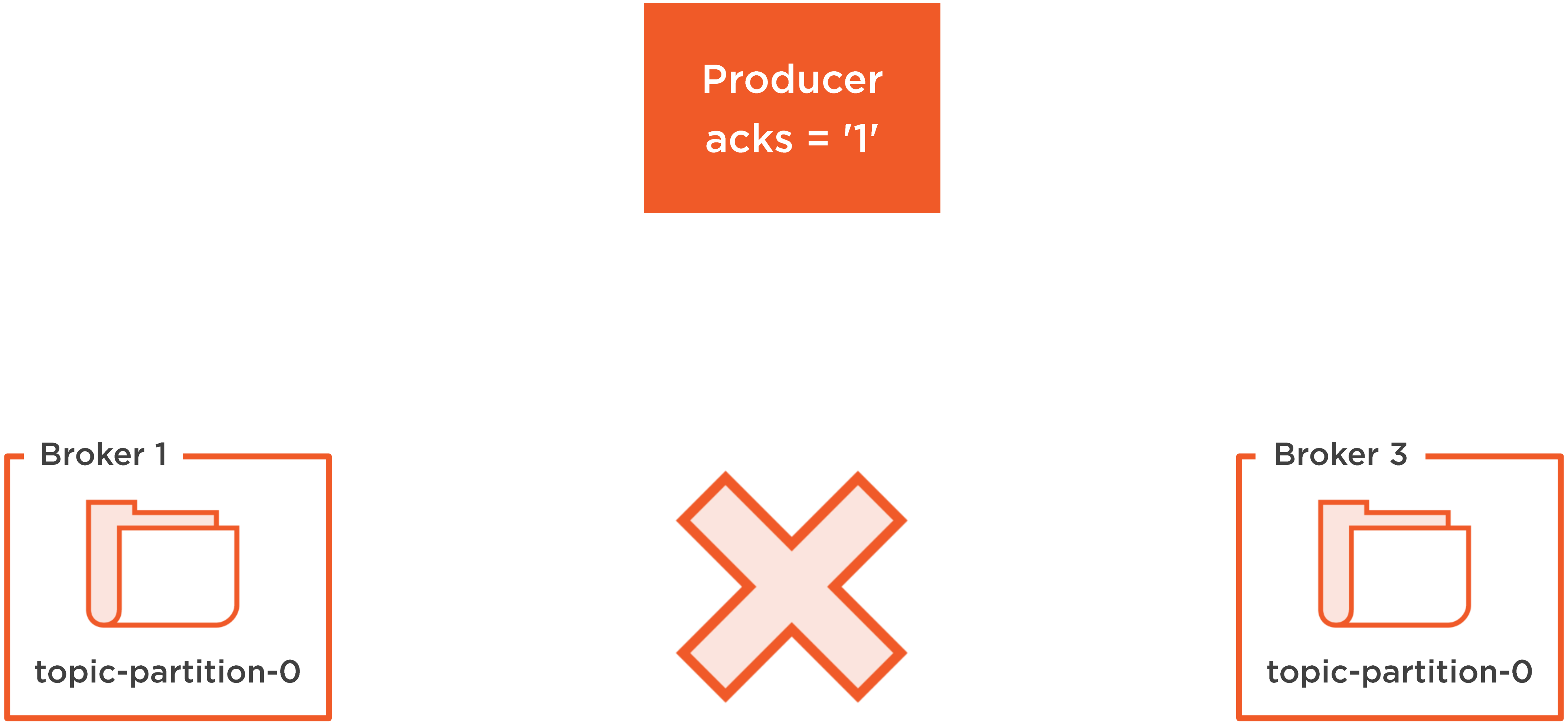
Creating a New Topic

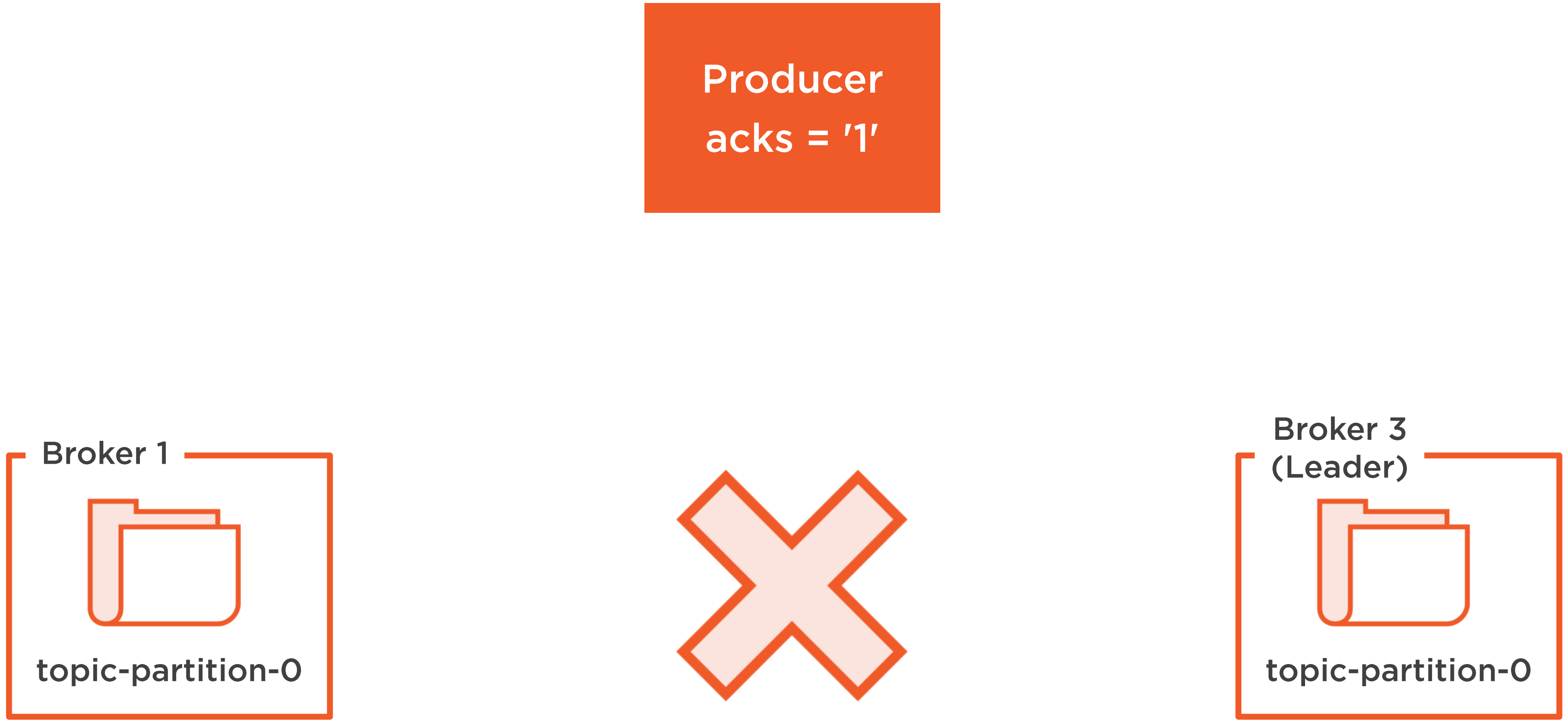
```
Admin admin = Admin.create(
        Map.of("bootstrap.servers", "localhost:9092")
);

NewTopic topic = new NewTopic("quote-feedback", 2, (short) 3)

admin.createTopics(List.of(topic)).get()
```

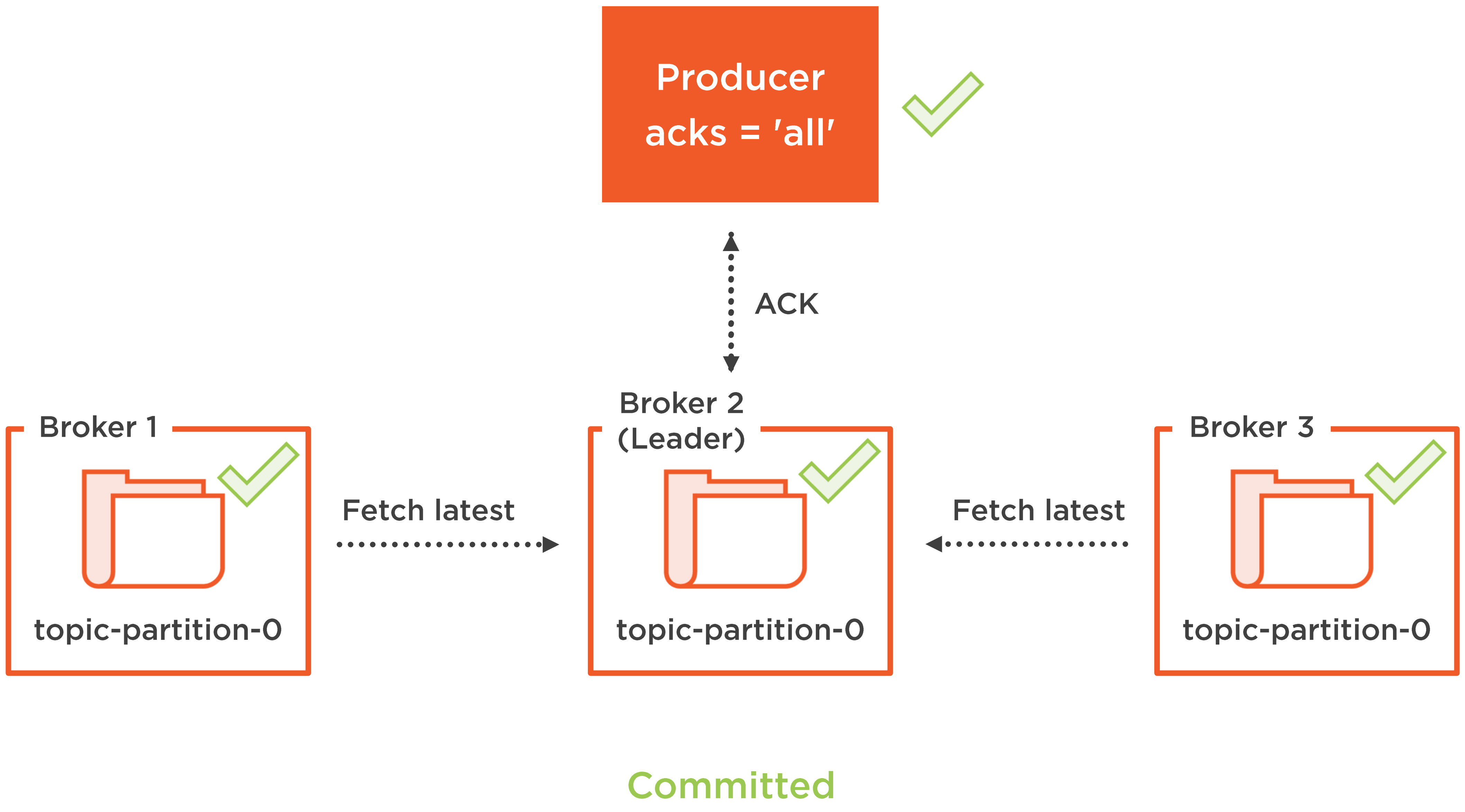
Replication factor

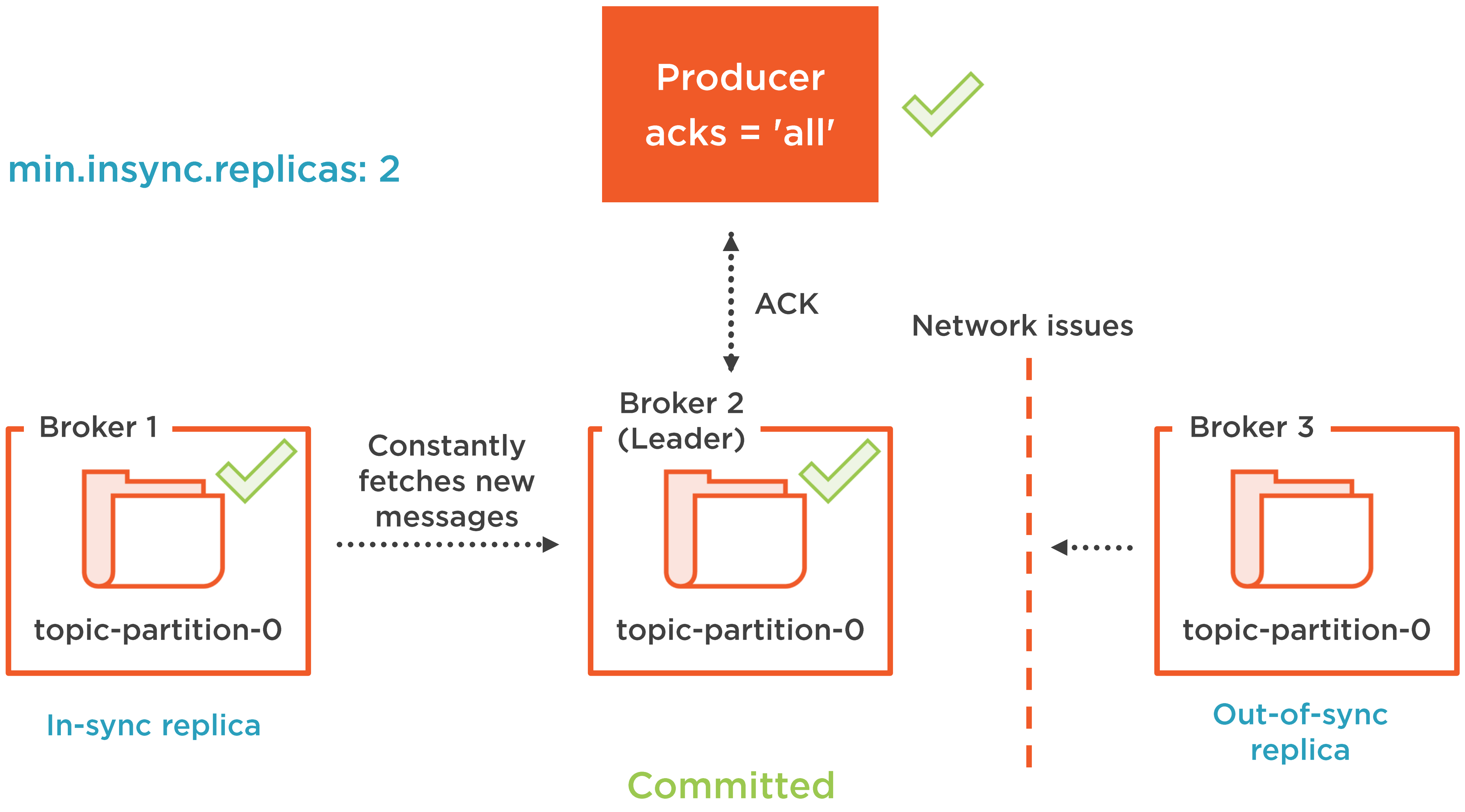

Producer
acks = '1'
ACK
Broker 1
topic-partition-0
Fetch latest
Broker 2
(Leader)
topic-partition-0
Fetch latest
Broker 3
topic-partition-0
Committed


Producer
acks = '1'
ACK
Broker 1
topic-partition-0
Broker 2
(Leader)
topic-partition-0
Broker 3
topic-partition-0

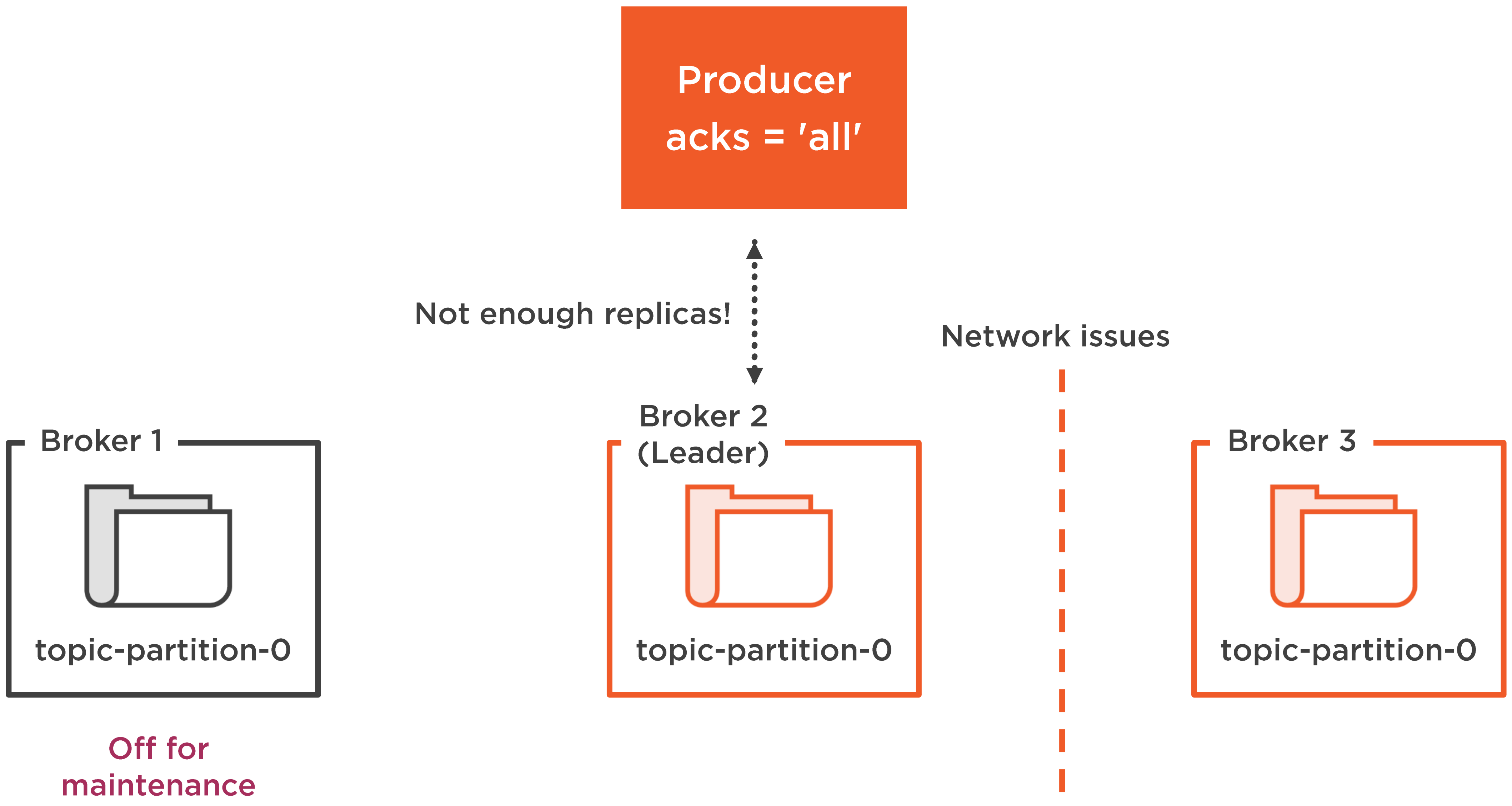

Producer
acks = '1'
Broker 1
topic-partition-0
Broker 3
topic-partition-0


Producer
acks = '1'
Broker 1
topic-partition-0
Broker 3
(Leader)
topic-partition-0

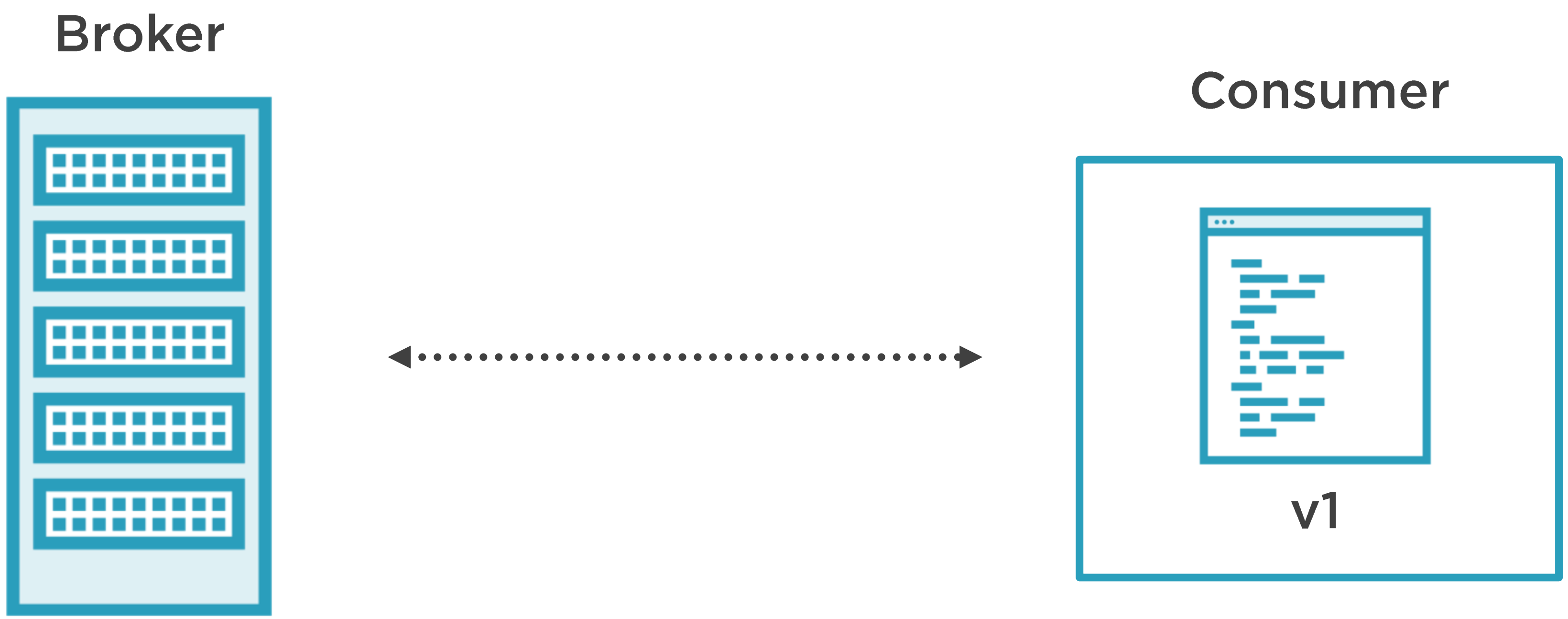

Producer
acks = 'all'
ACK
Broker 1
topic-partition-0
Fetch latest
Broker 2
(Leader)
topic-partition-0
Fetch latest
Broker 3
topic-partition-0
Committed

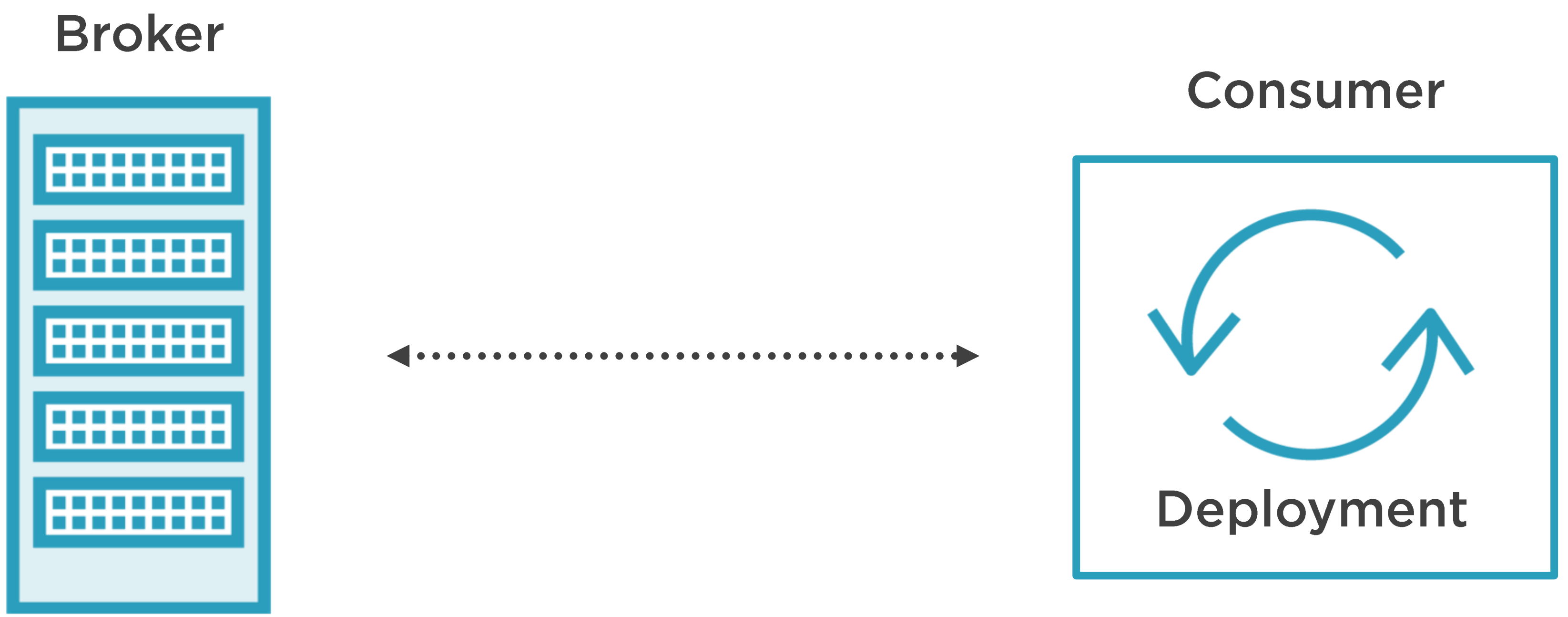


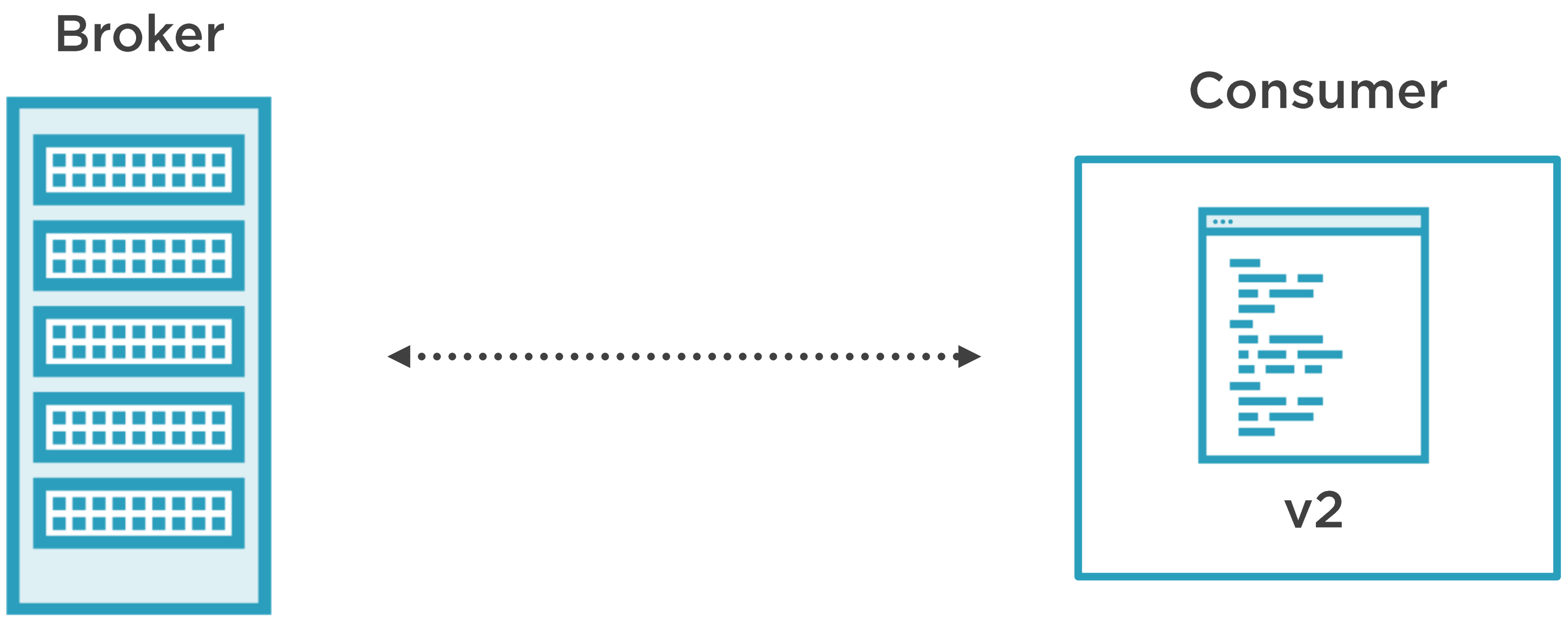
min.insync.replicas: 2
Producer
acks = 'all'
ACK
Network issues
Broker 1
topic-partition-0
In-sync replica
Constantly fetches new messages
Broker 2 (Leader)
topic-partition-0
Committed
Broker 3
topic-partition-0
Out-of-sync replica

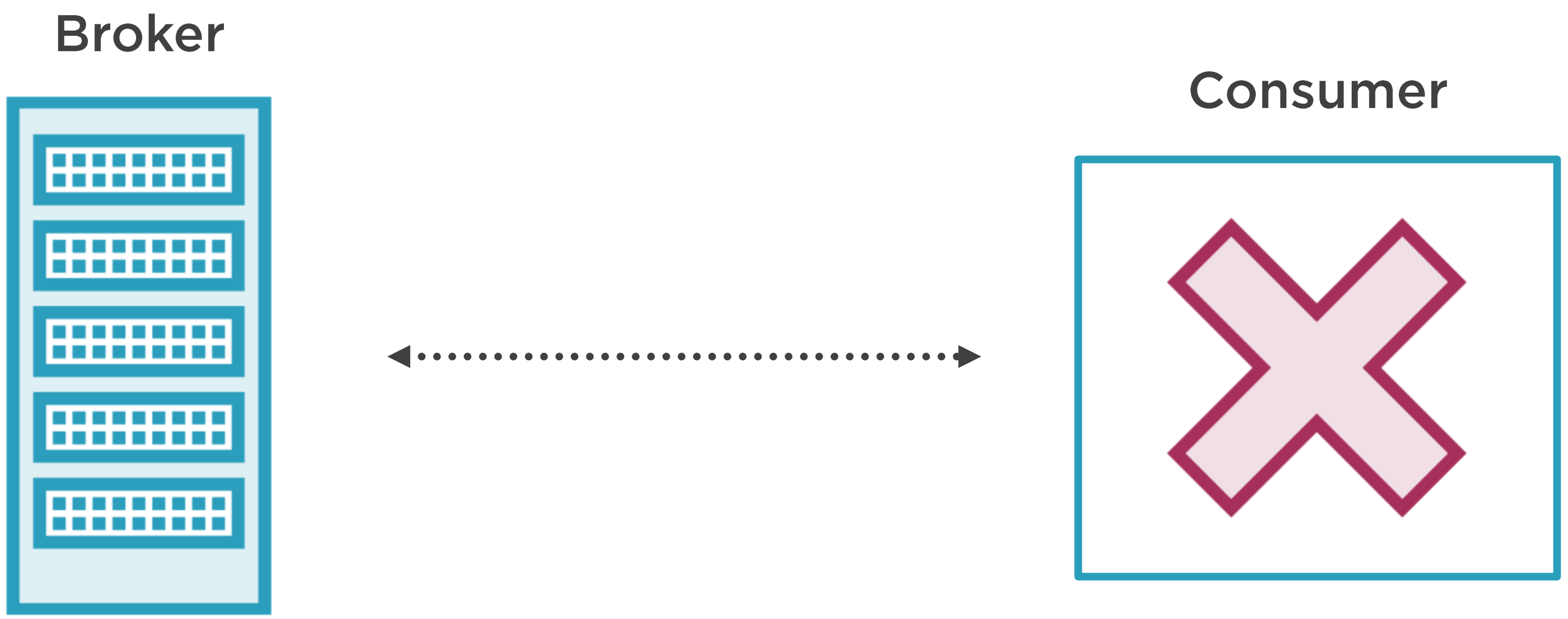

Producer
acks = 'all'
Not enough replicas!
Network issues
Broker 1
topic-partition-0
Off for maintenance
Broker 2 (Leader)
topic-partition-0
Broker 3
topic-partition-0

Data Retention and Cluster Sizing

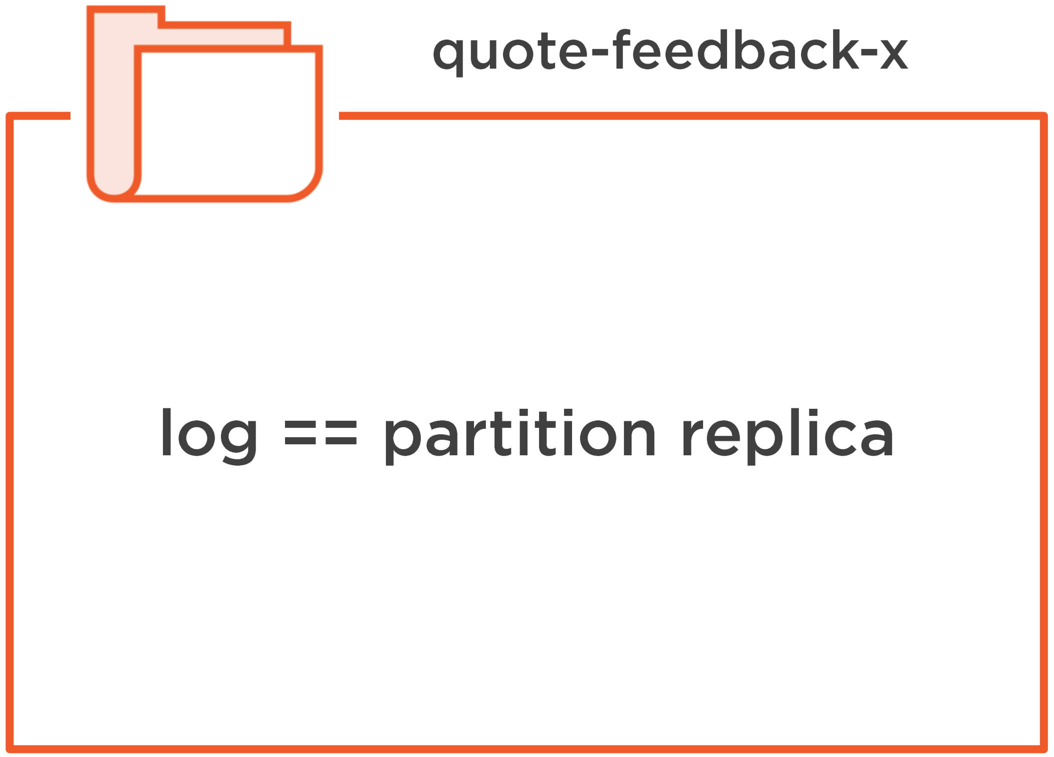

Broker
Consumer
v1


Broker
Consumer
Deployment


Broker
Consumer
v2

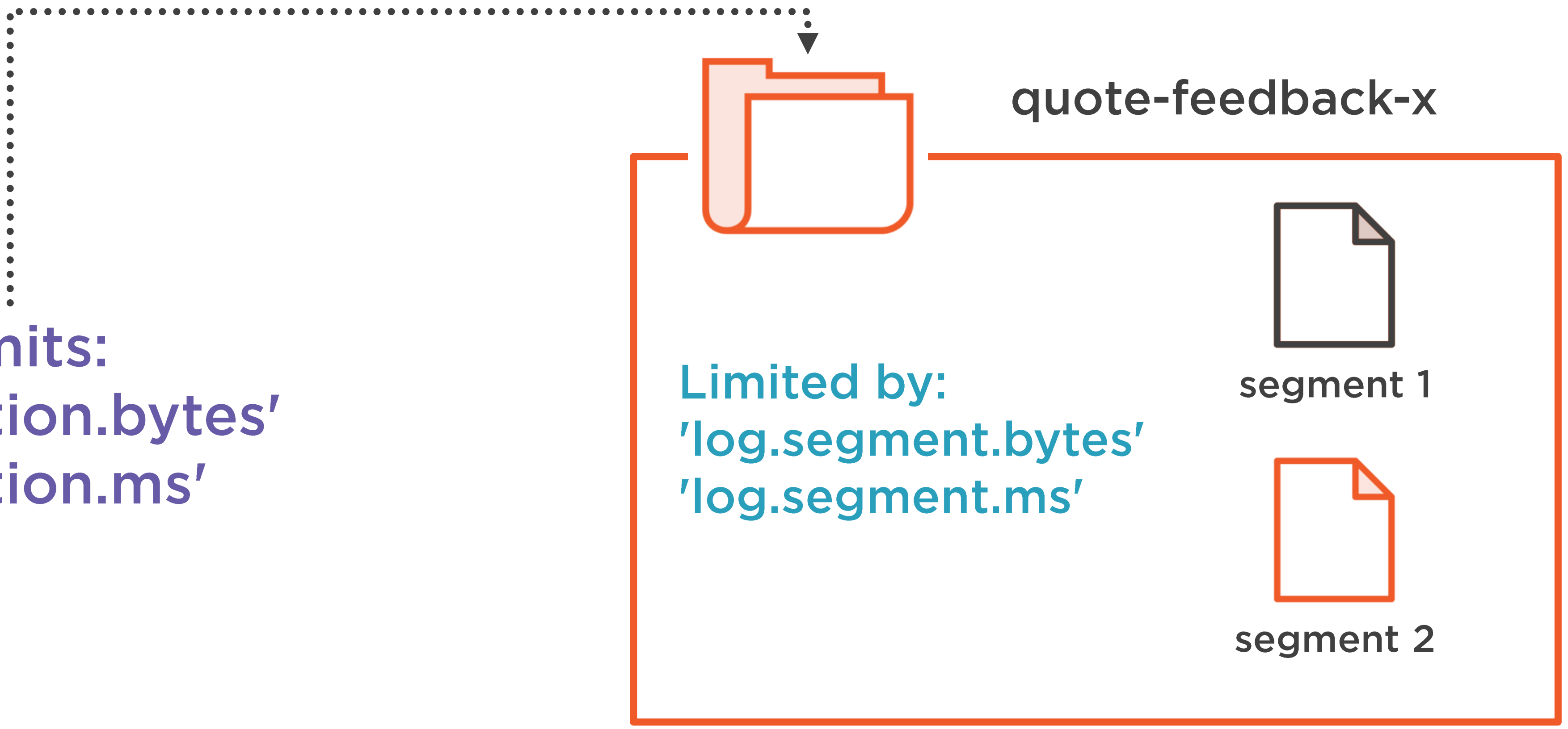

Broker
Consumer

Data Retention



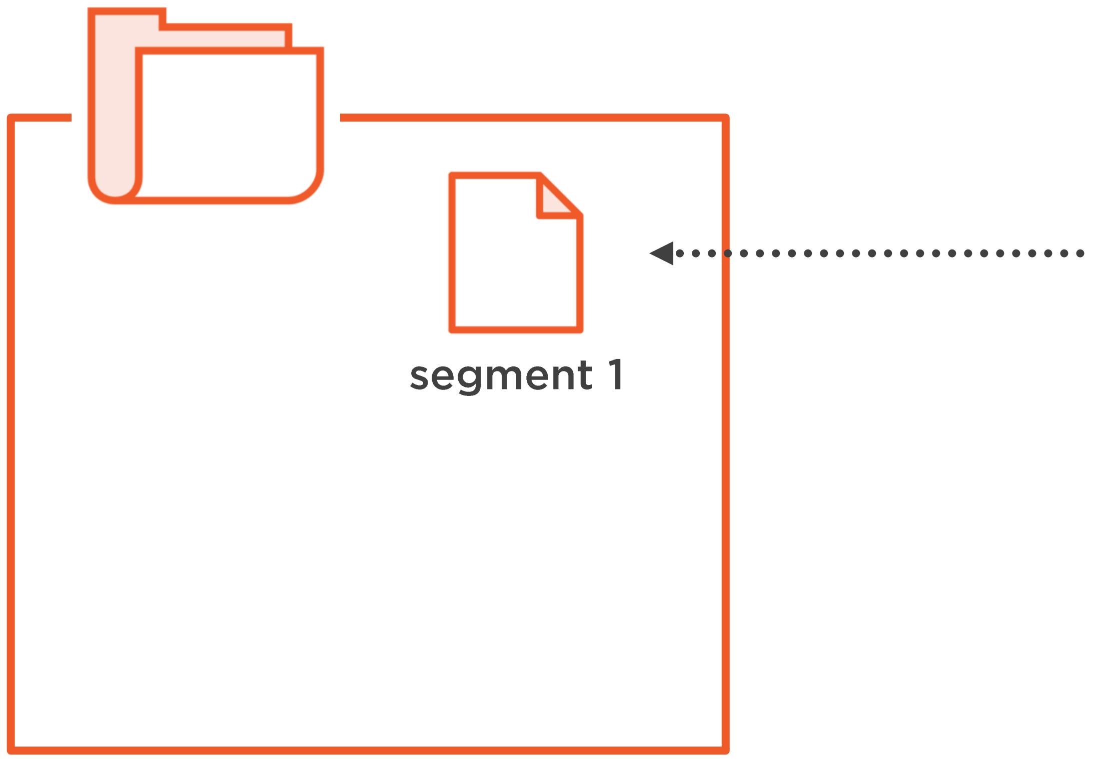
Think about possible log size when choosing the number of partitions for a topic

Data Retention



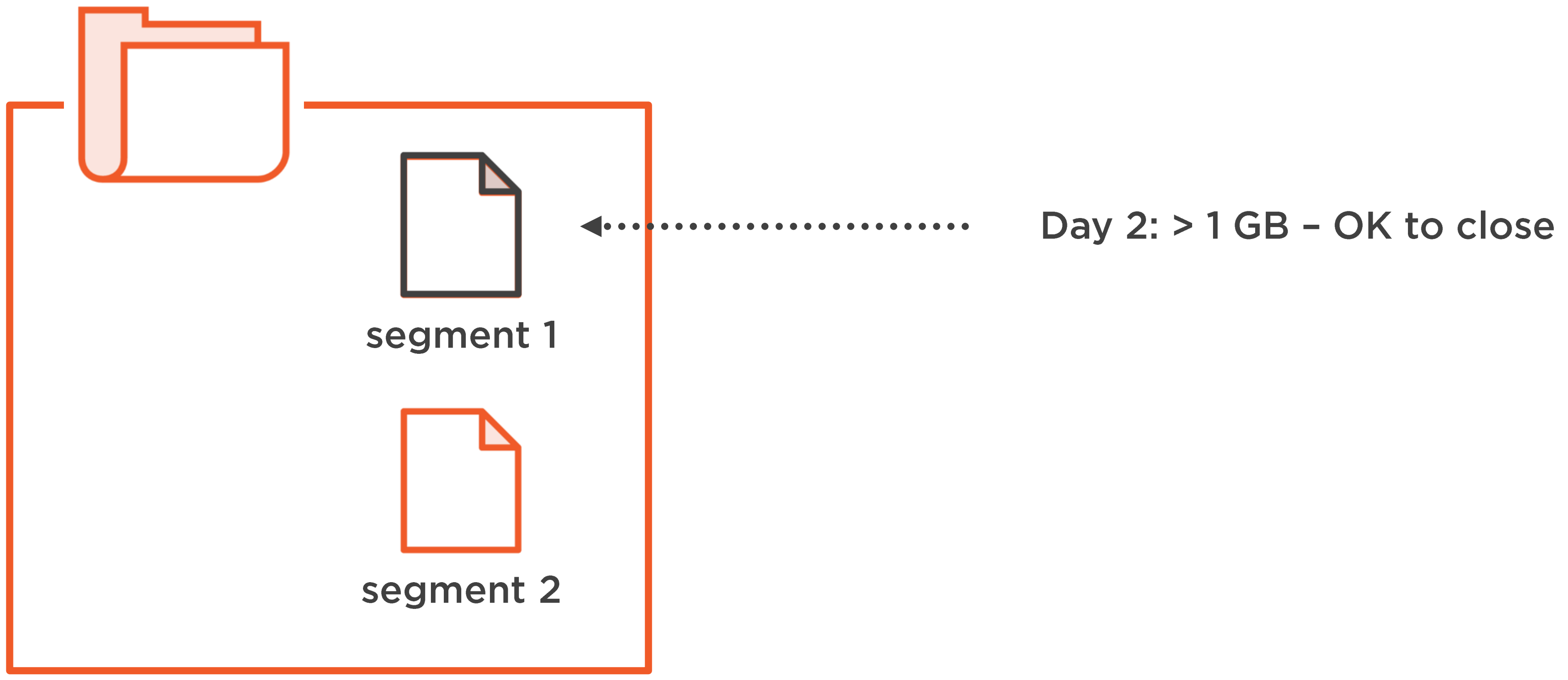
Per-log limits:
'log.retention.bytes'
'log.retention.ms'

Limited by:
'log.segment.bytes'
'log.segment.ms'

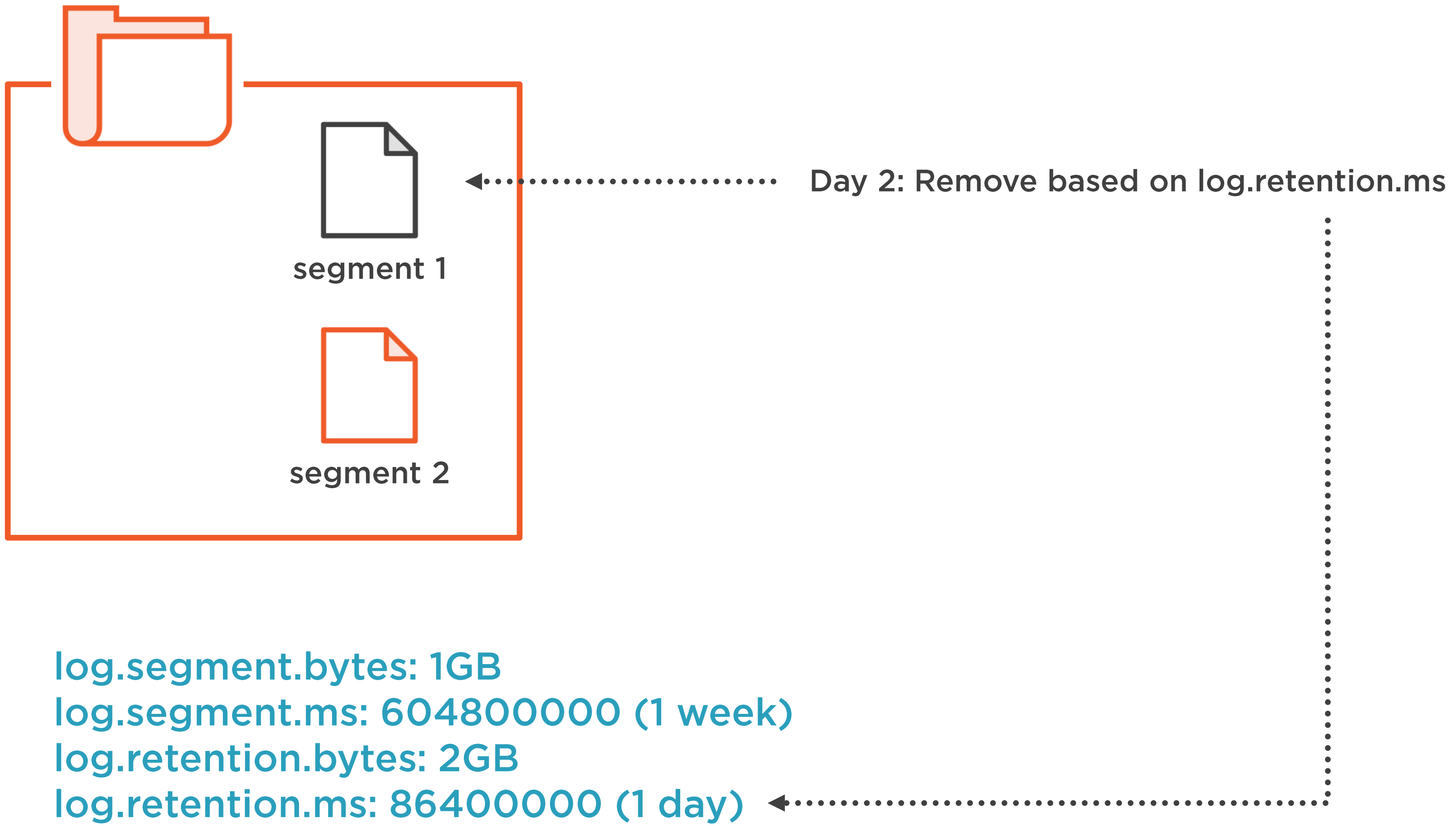


Day 1: ~500MB – cannot close

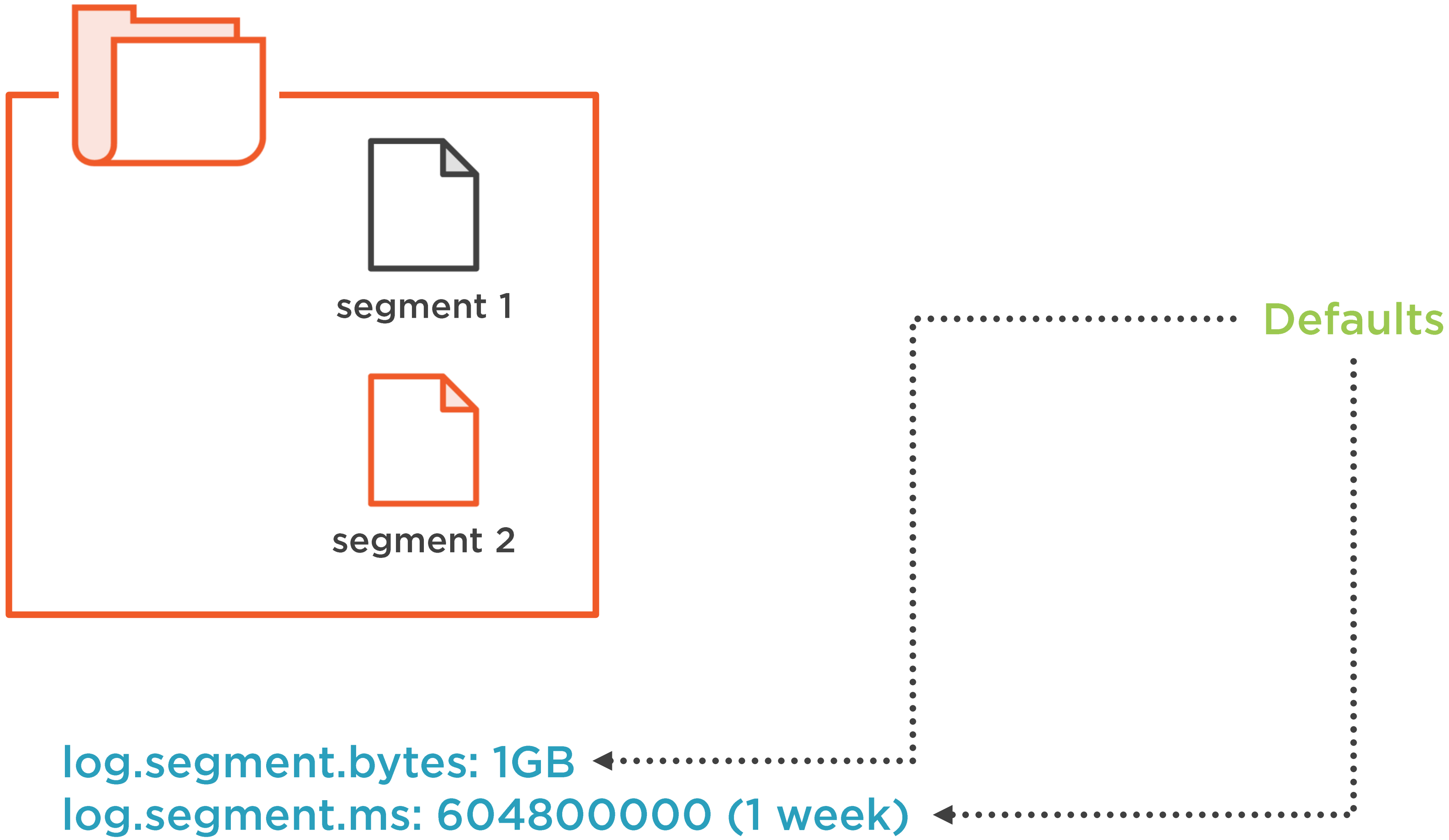
```
log.segment.bytes: 1GB
log.segment.ms: 604800000 (1 week)
log.retention.bytes: 2GB
log.retention.ms: 86400000 (1 day)
```



log.segment.bytes: 1GB
log.segment.ms: 604800000 (1 week)
log.retention.bytes: 2GB
log.retention.ms: 86400000 (1 day)



log.segment.bytes: 1GB
log.segment.ms: 604800000 (1 week)
log.retention.bytes: 2GB
log.retention.ms: 86400000 (1 day)



log.segment.bytes: 1GB
log.segment.ms: 604800000 (1 week)
log.retention.bytes: 2GB
log.retention.ms: 86400000 (1 day)

Cluster Sizing

Usual bottlenecks:

- Disk size/throughput
- Network throughput

Always estimate the load before deploying a cluster

Summary

Why multiple brokers are needed

Reliability guarantees

Practical example

Basics of data retention and cluster sizing