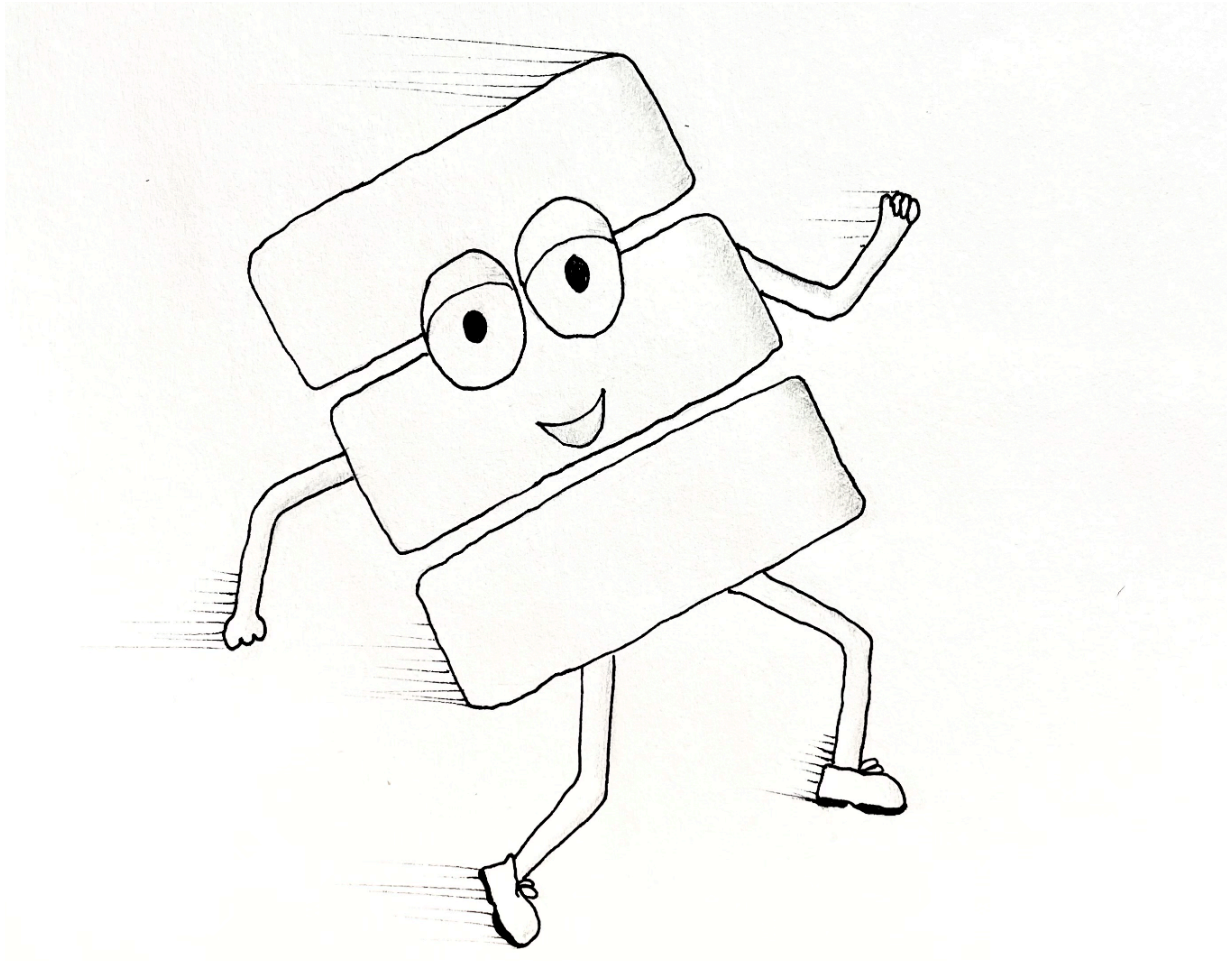
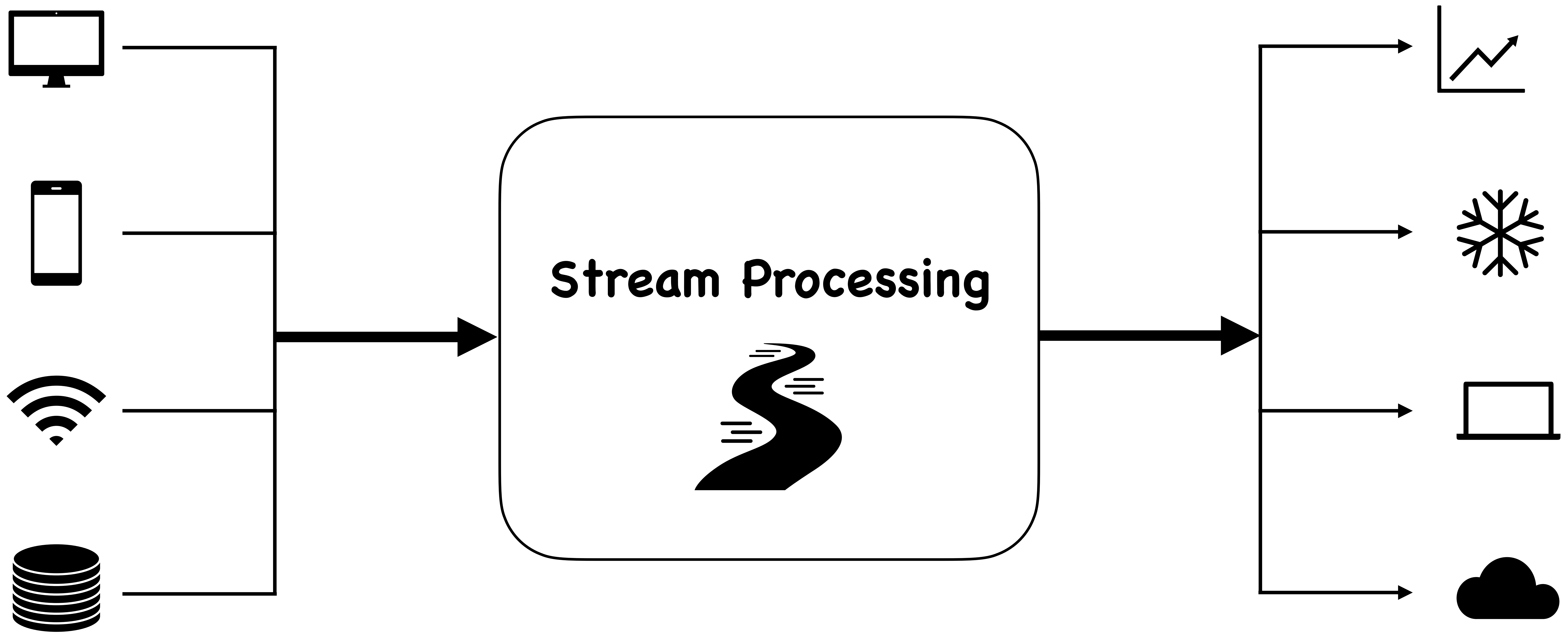
Harness the power of data in motion with Python and Kafka

Dave Klein

daveklein@usa.net twitter.com/daveklein linkedin.com/in/daveklein19

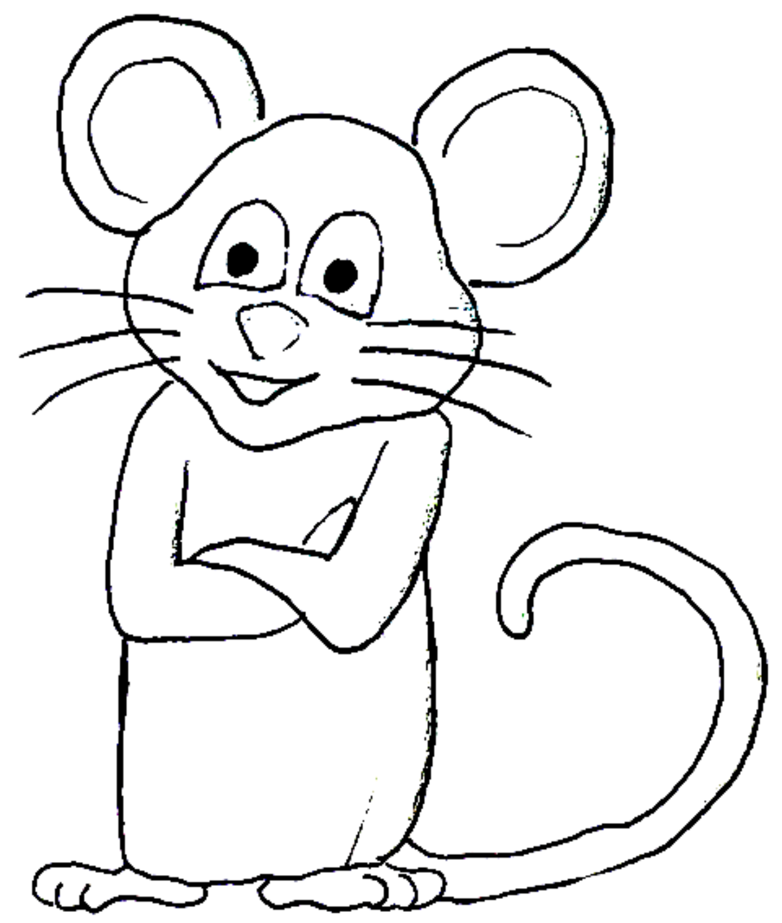
What is Data in Motion?




Stream Processing

What Is It Good For?

- Customer 360
- Recommendation Engines
- Factory Automation
- Inventory Management
- Fraud Detection
- Customer Loyalty Programs
- Fleet Management
- Real-time Payments



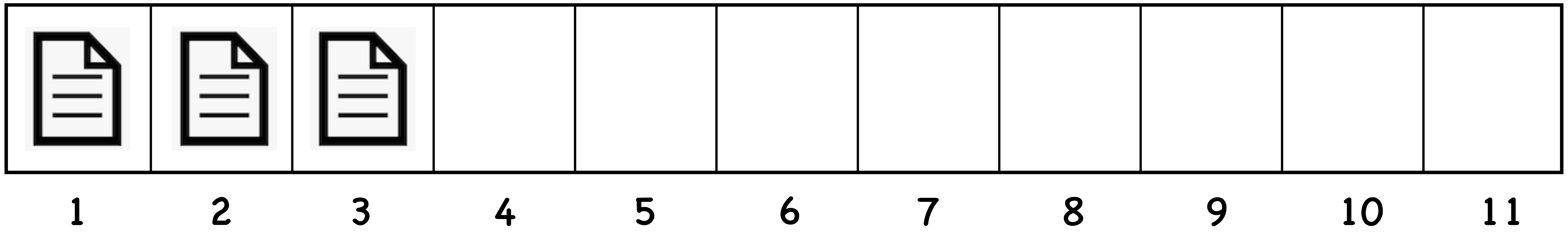
Kafka

The Diminutive Guide

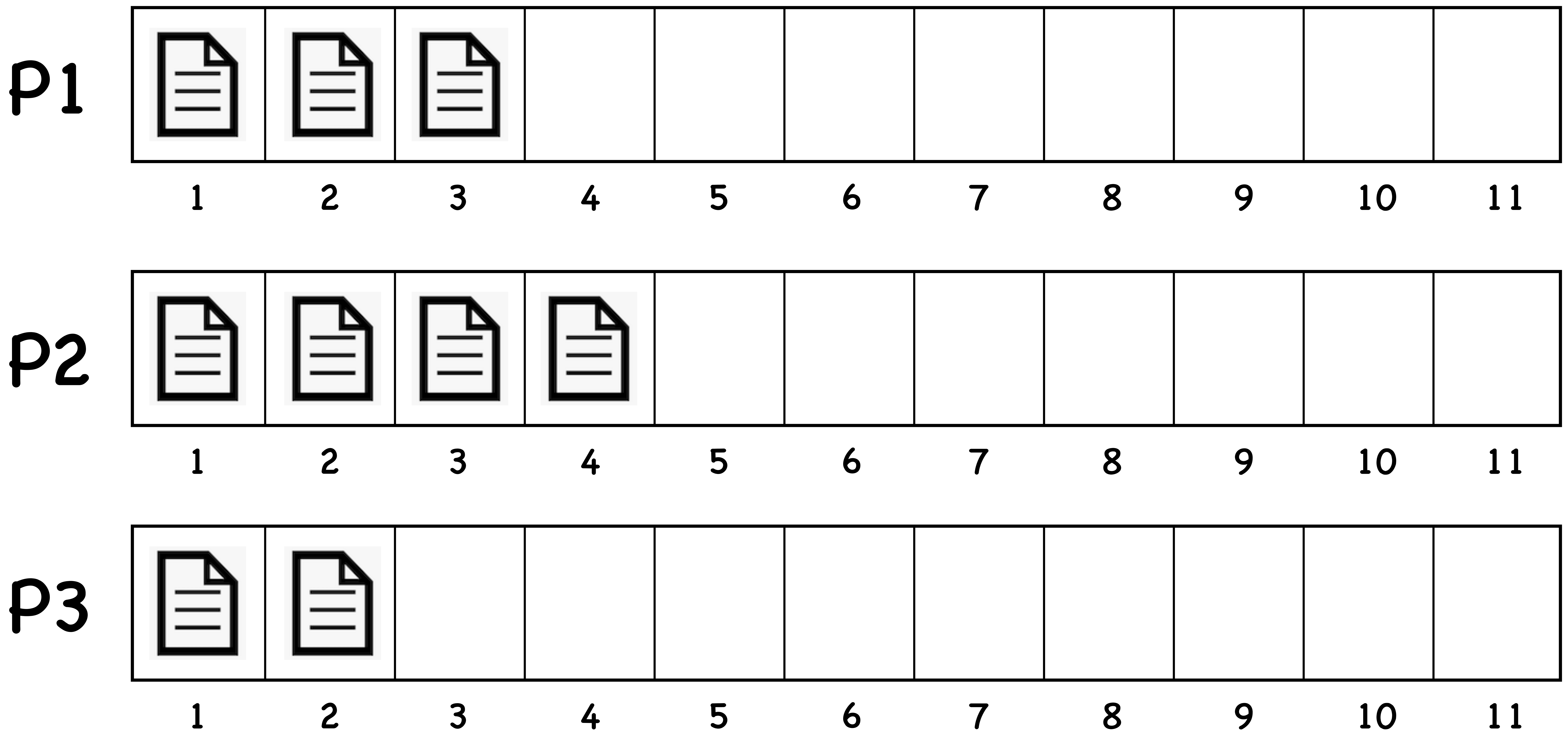
daveklein@usa.net

@daveklein

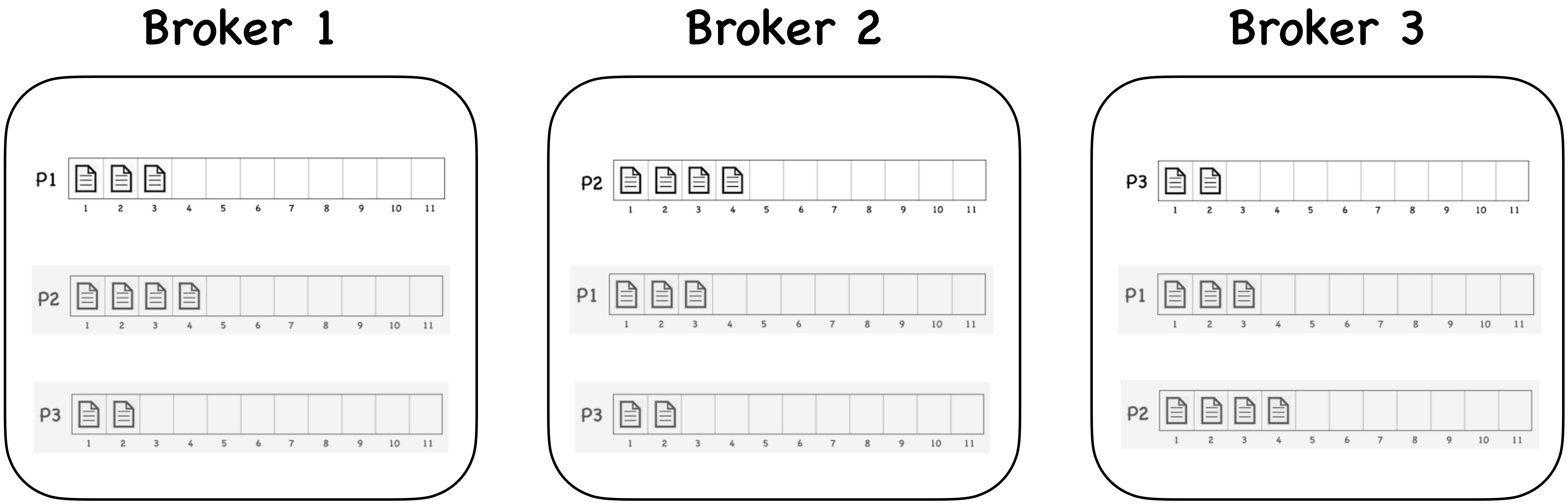
Topic (log)



Topic Partitions

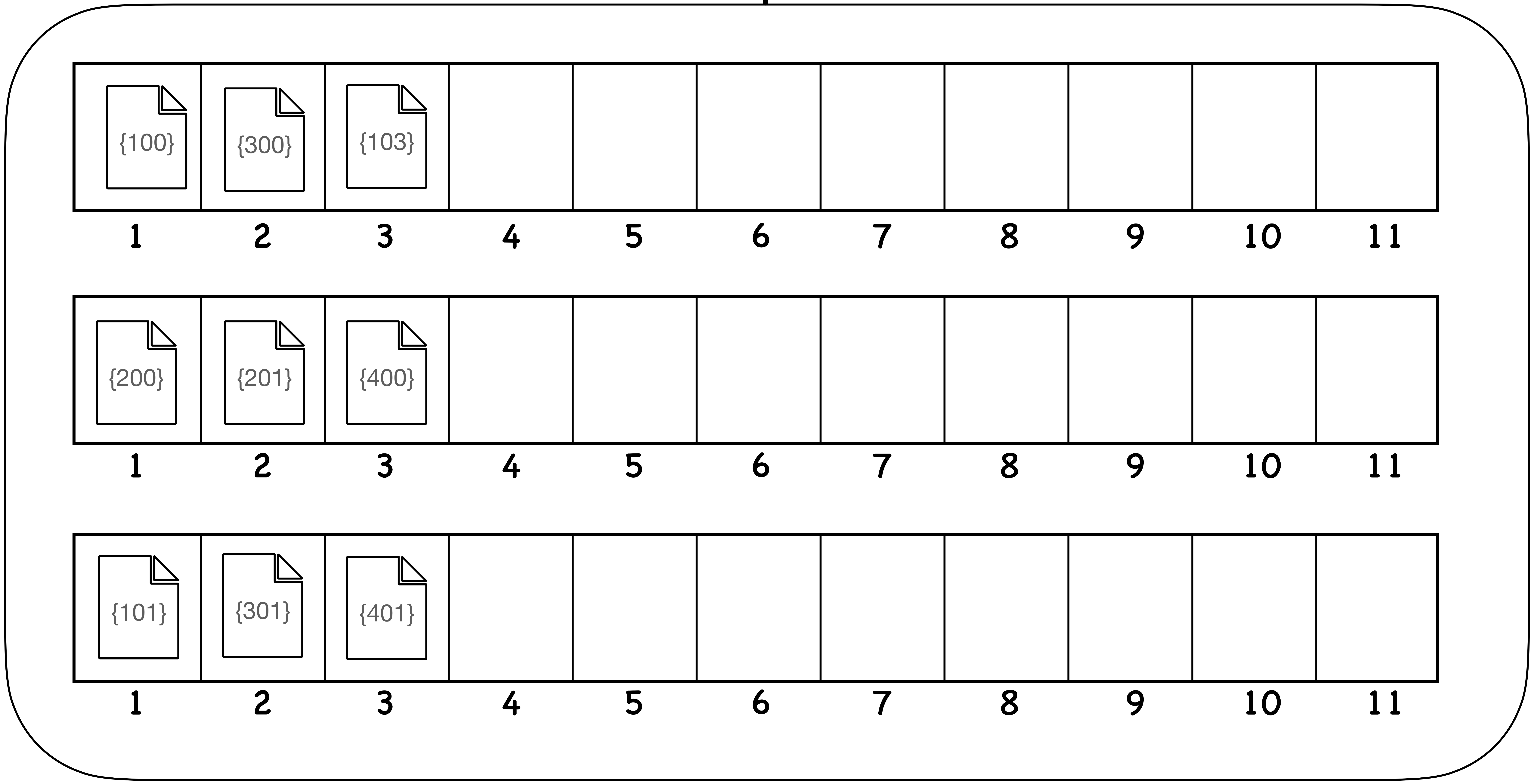


Brokers and Replication

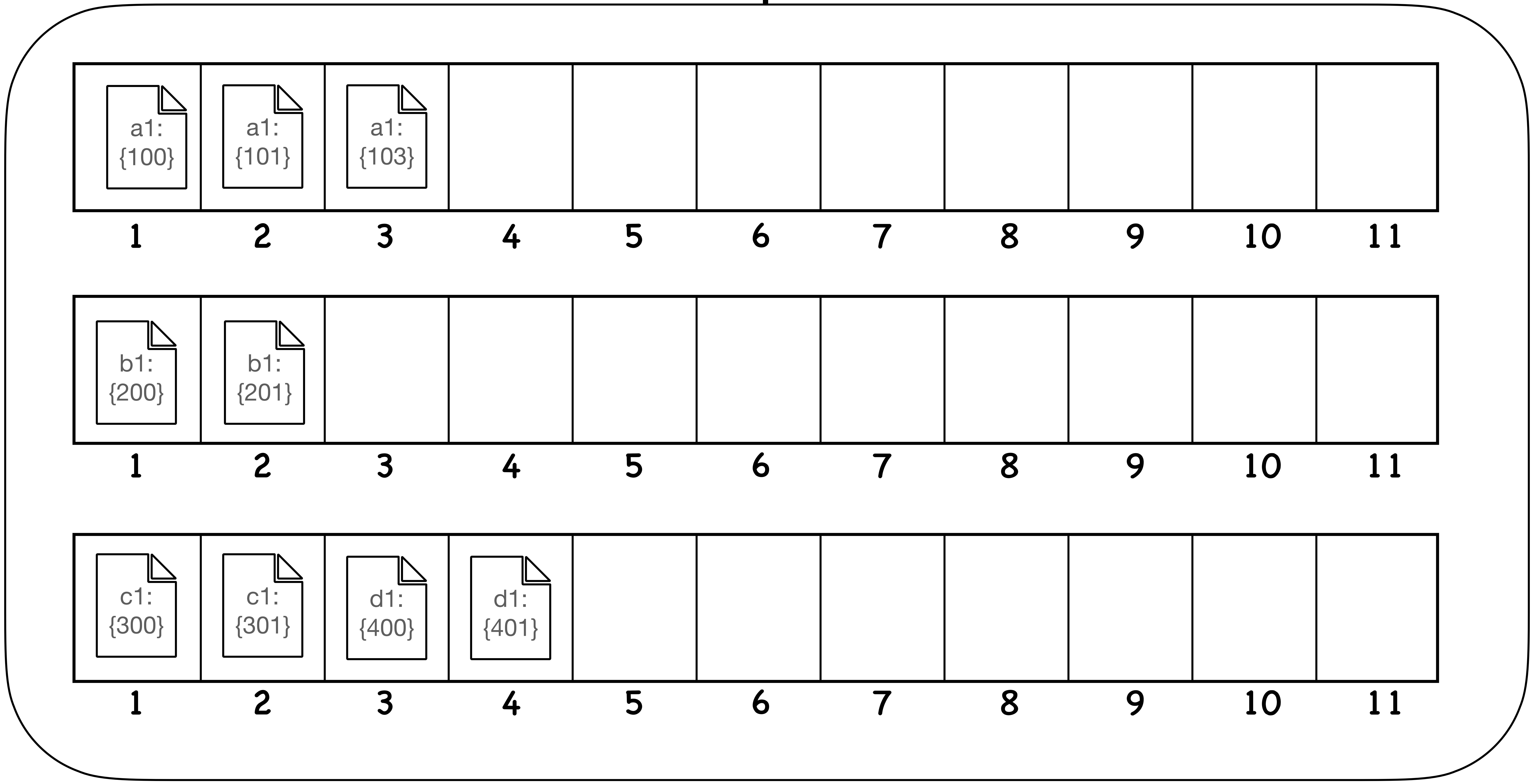


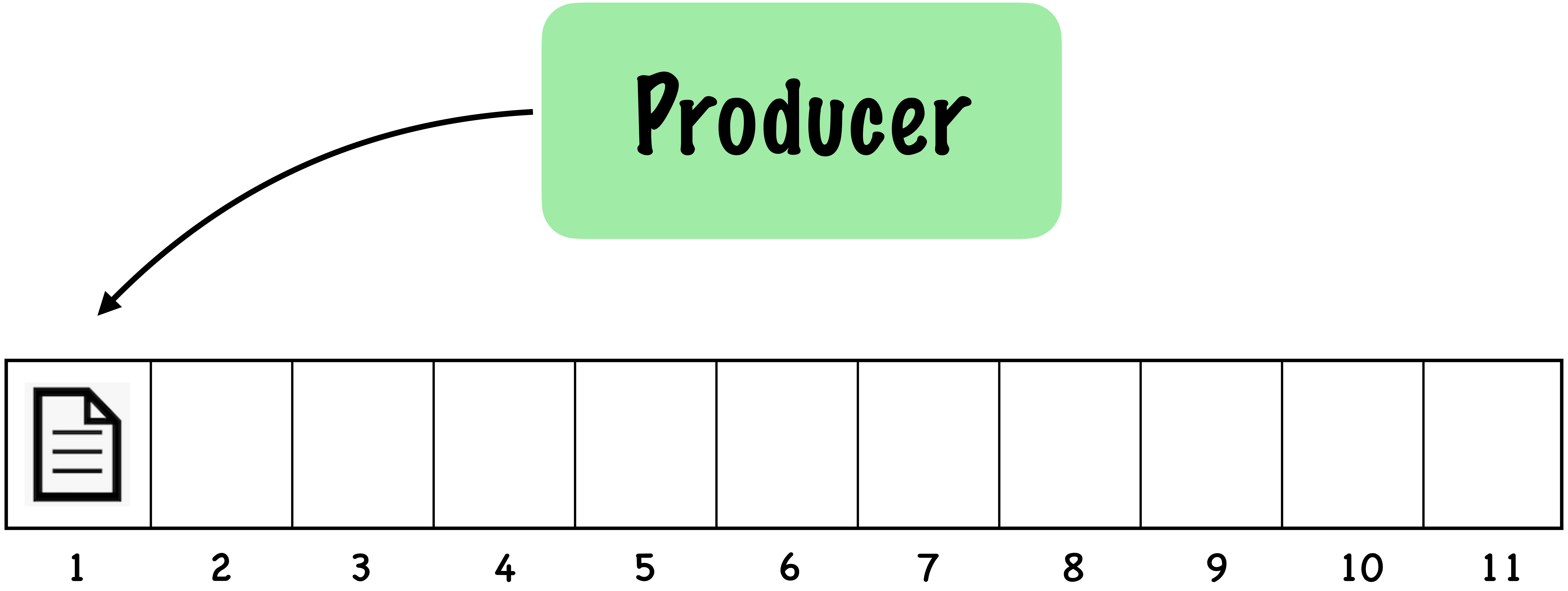
```
"key": "a1",
"value": {
    "eventType": “added-to-cart",
    "title": "Kafka Streams in Action”,
    "author": "Bill Bejeck",
    "price": 44.99
}
```

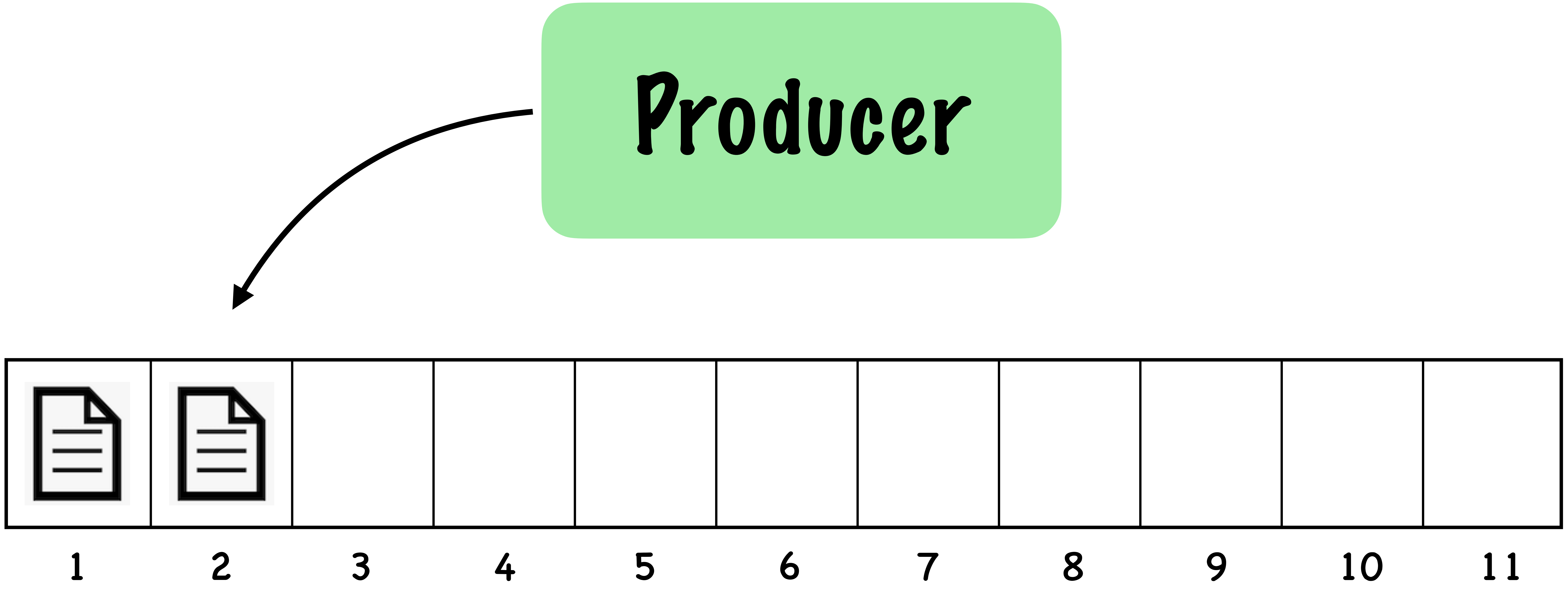
Topic

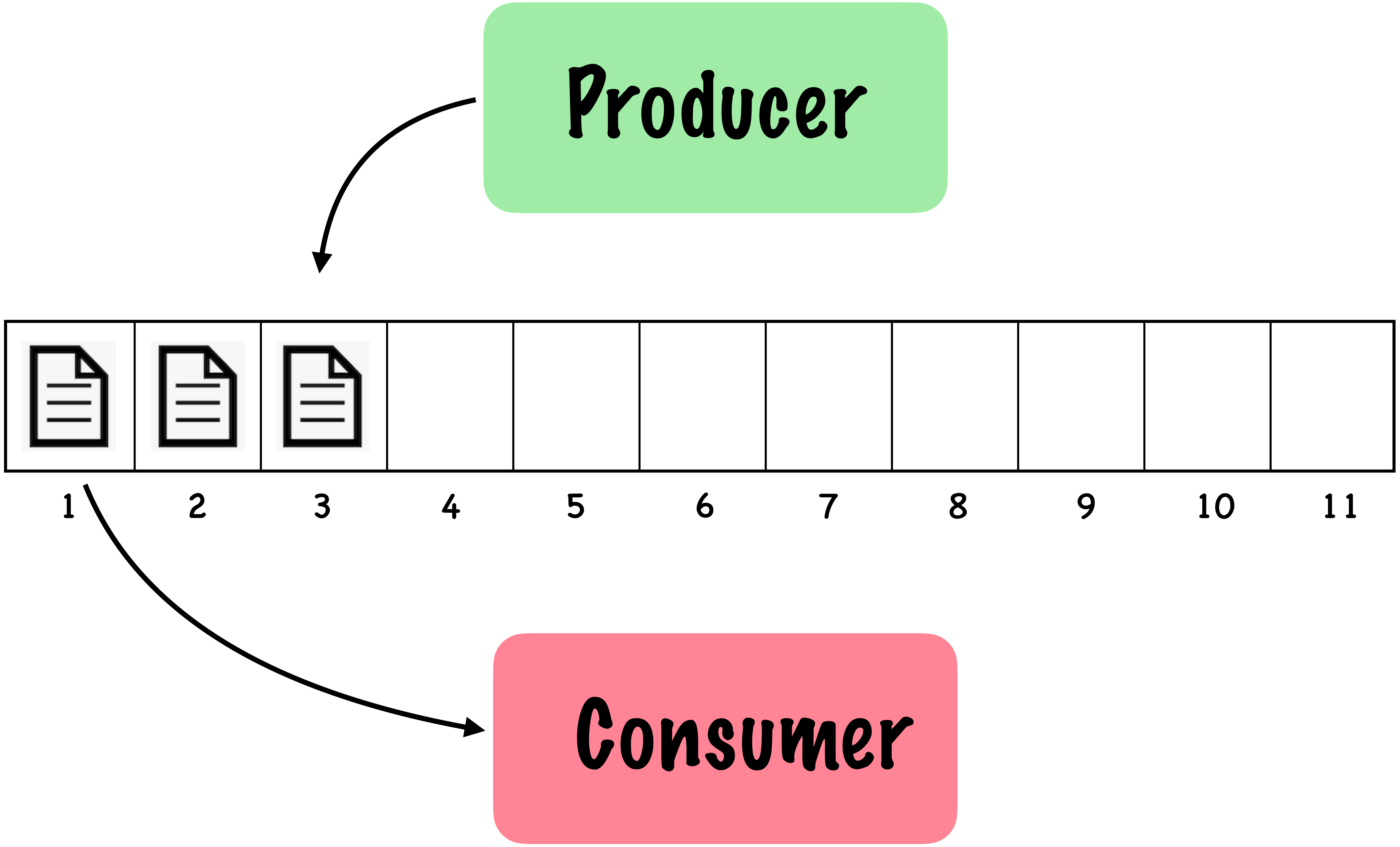

{100}
{300}
{103}
1
2
3
4
5
6
7
8
9
10
11
{200}
{201}
{400}
1
2
3
4
5
6
7
8
9
10
11
{101}
{301}
{401}
1
2
3
4
5
6
7
8
9
10
11

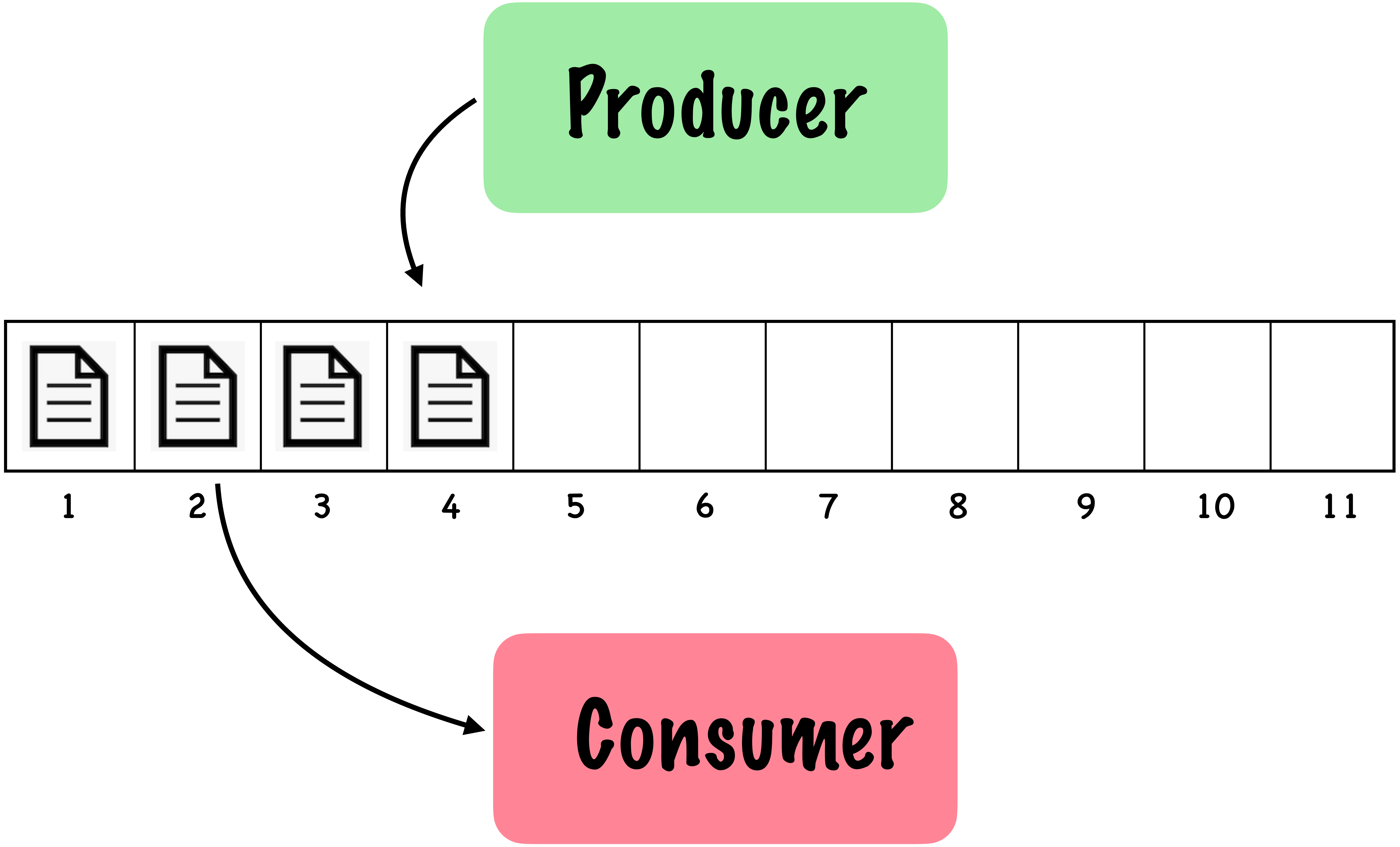
Topic

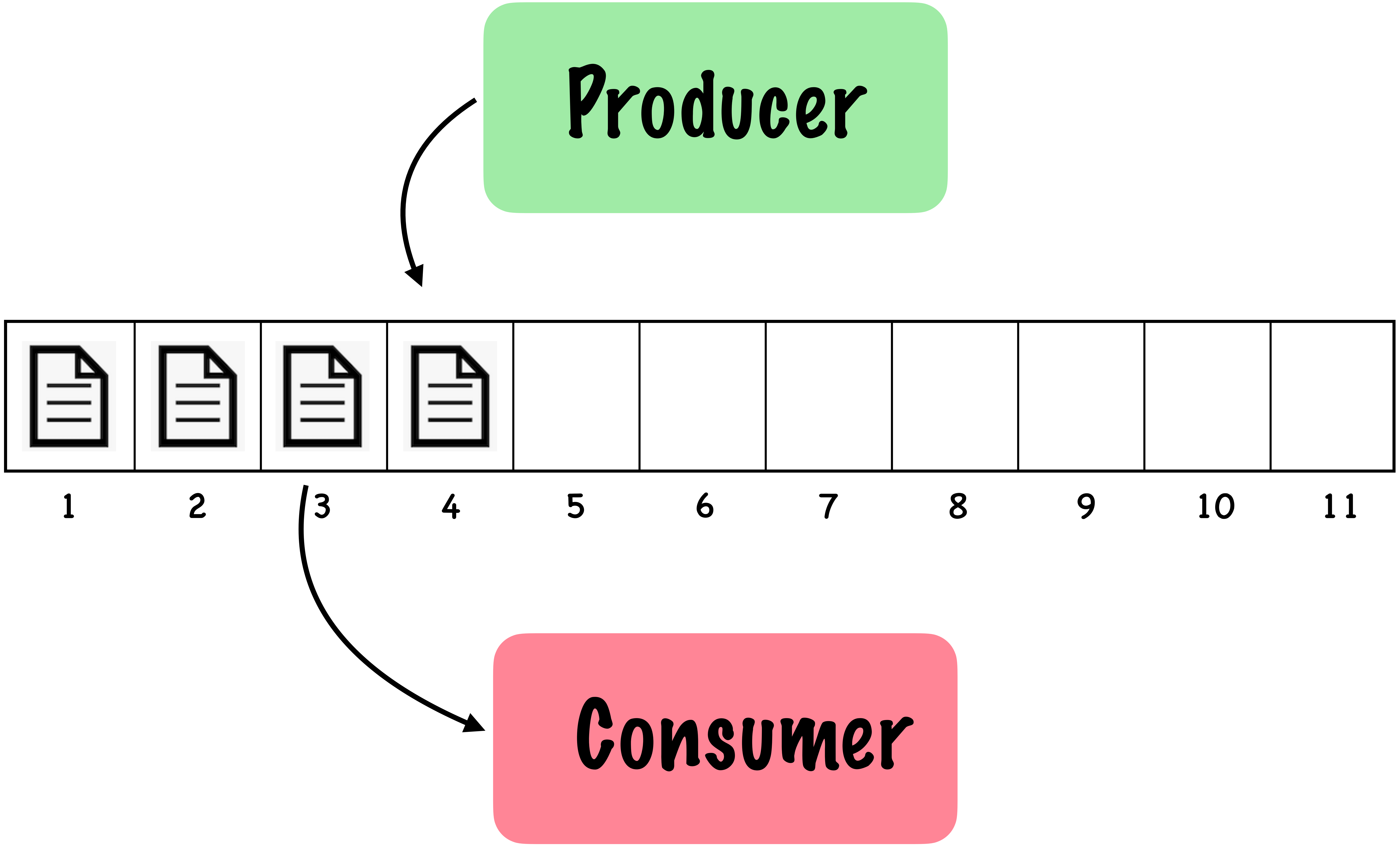

a1:
{100}
a1:
{101}
a1:
{103}
1 2 3 4 5 6 7 8 9 10 11
b1:
{200}
b1:
{201}
1 2 3 4 5 6 7 8 9 10 11
c1:
{300}
c1:
{301}
d1:
{400}
d1:
{401}
1 2 3 4 5 6 7 8 9 10 11

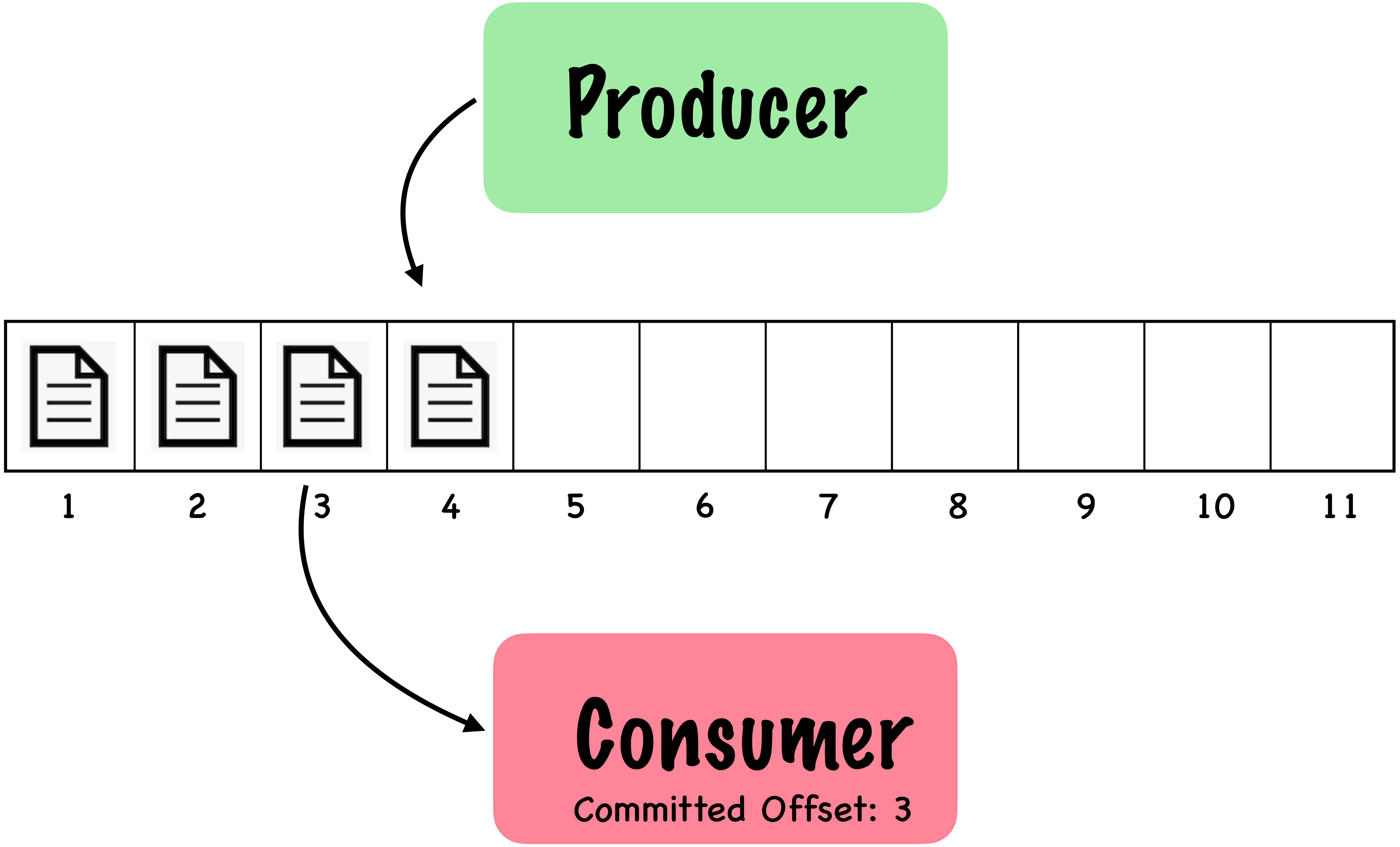

Producer
1
2
3
4
5
6
7
8
9
10
11

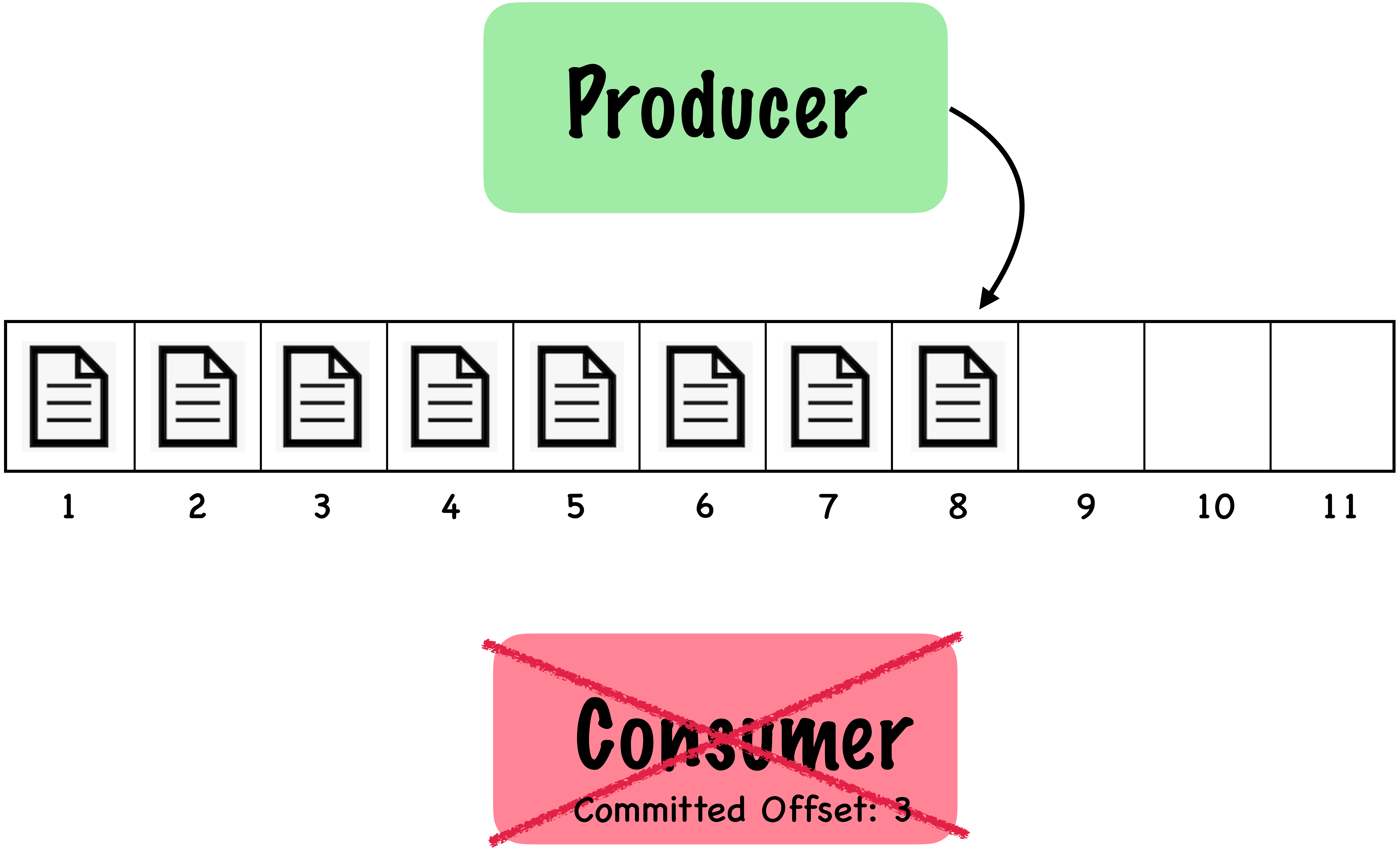

Producer
1
2
3
4
5
6
7
8
9
10
11

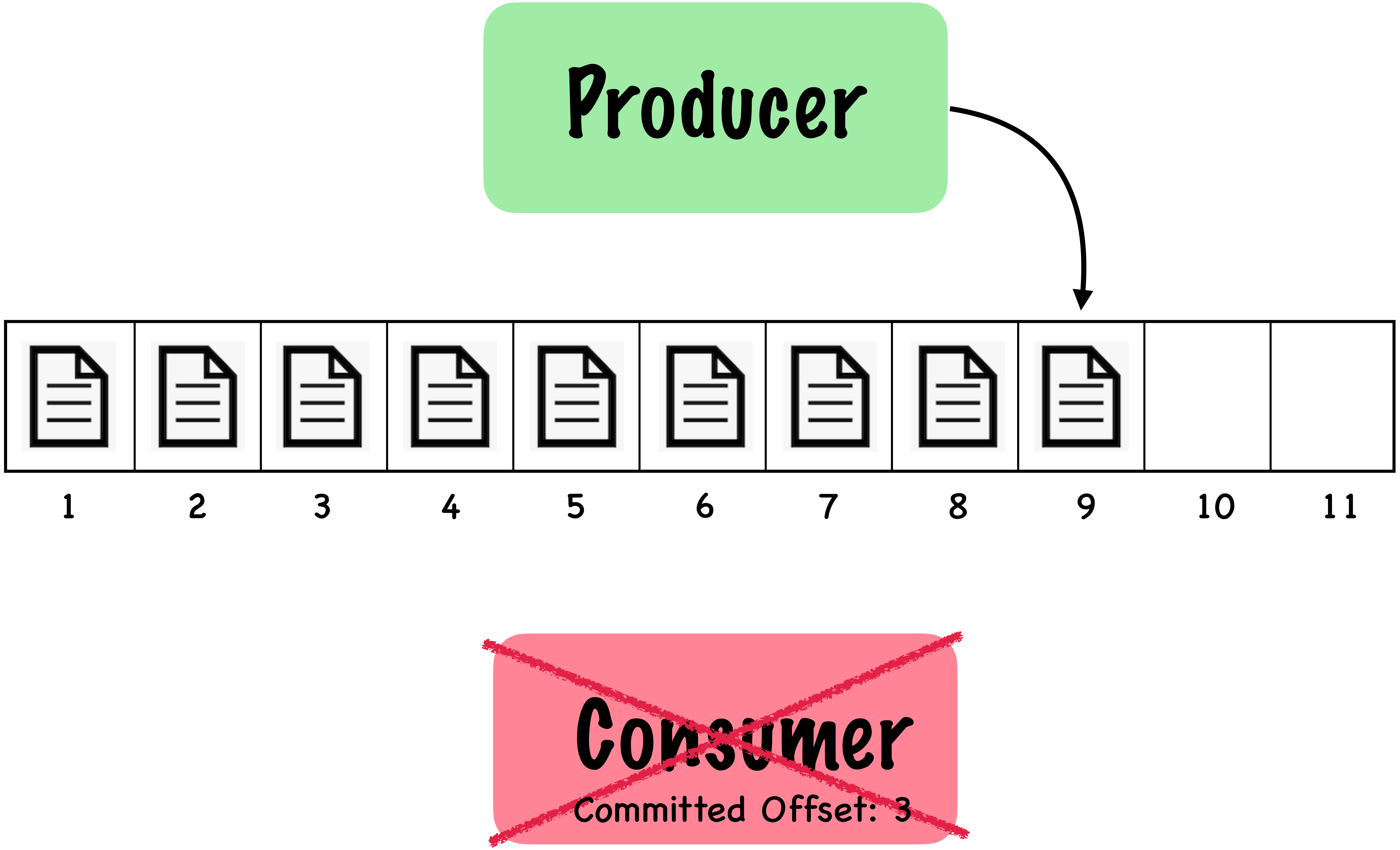

Producer
1
2
3
4
5
6
7
8
9
10
11
Consumer

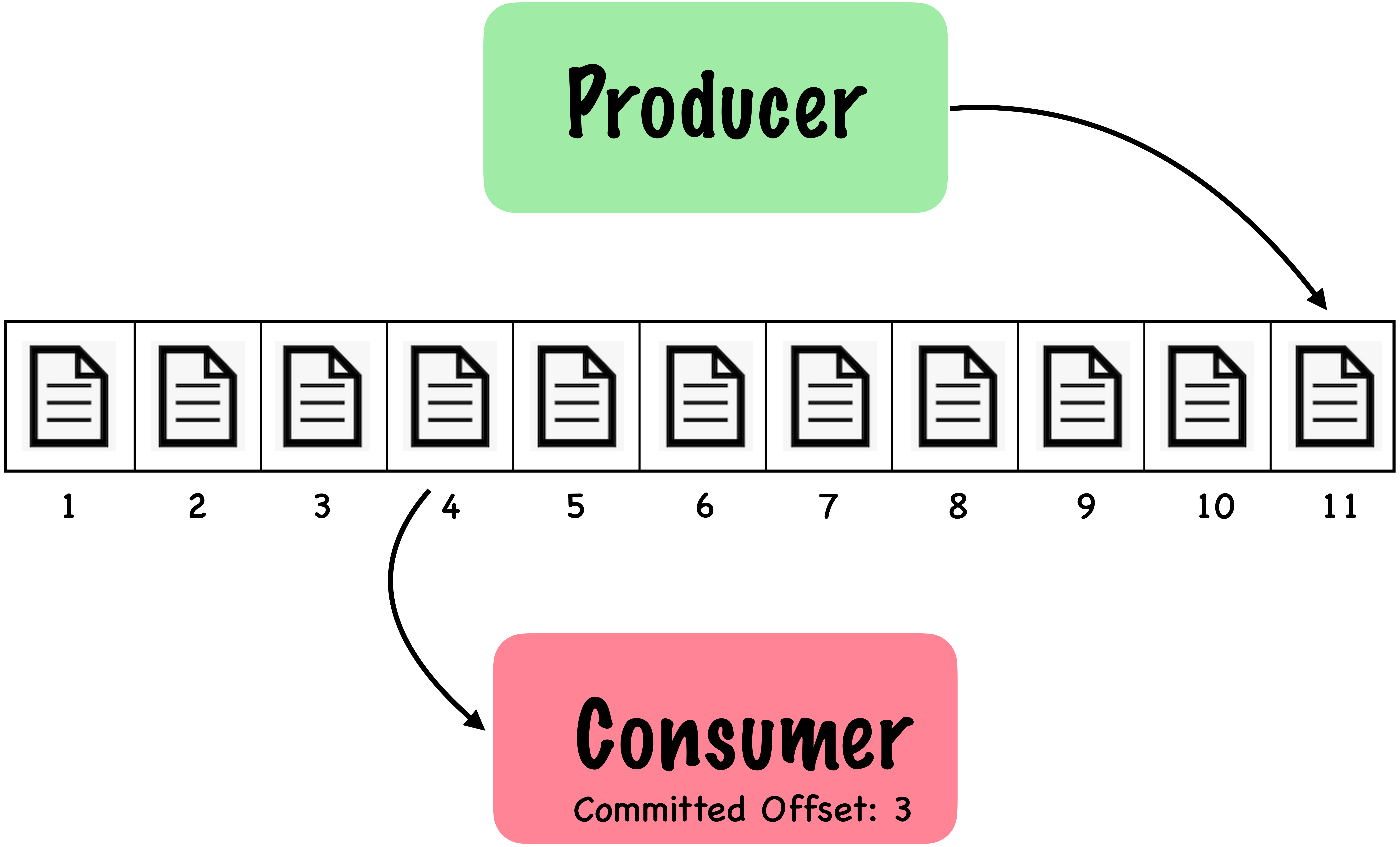

Producer
1
2
3
4
5
6
7
8
9
10
11
Consumer

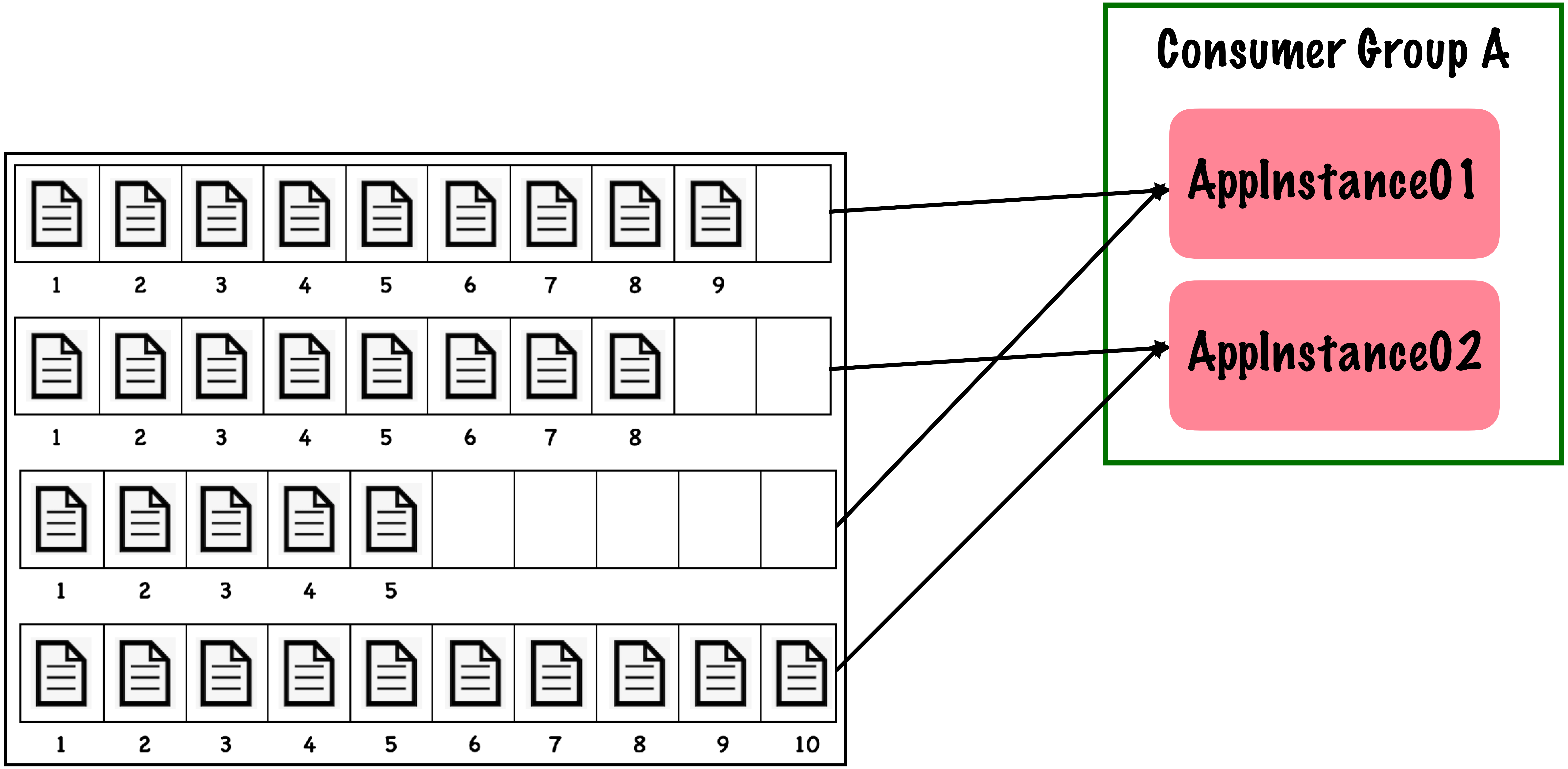

Producer
1
2
3
4
5
6
7
8
9
10
11
Consumer

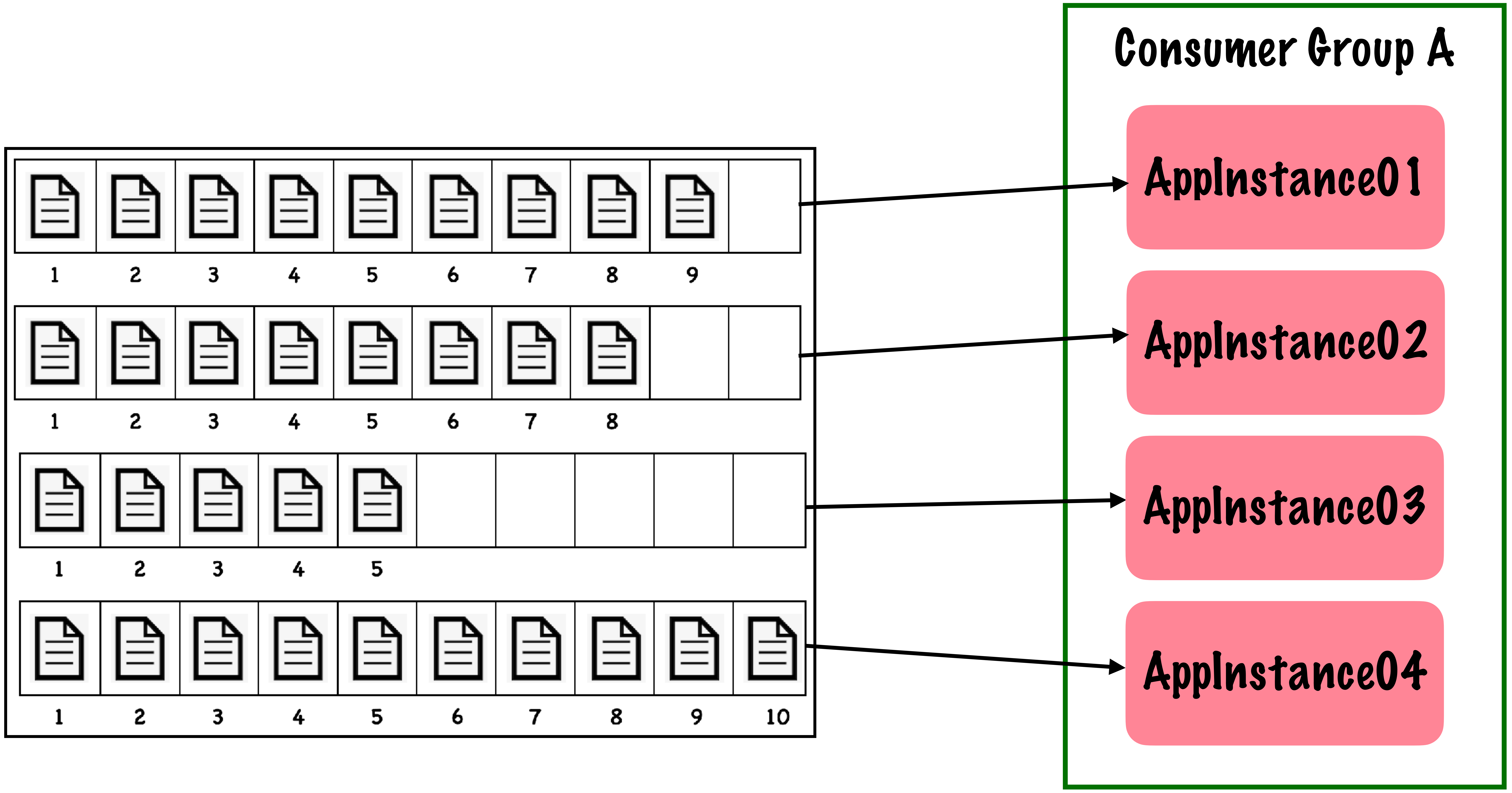

Producer
1
2
3
4
5
6
7
8
9
10
11
Consumer
Committed Offset: 3

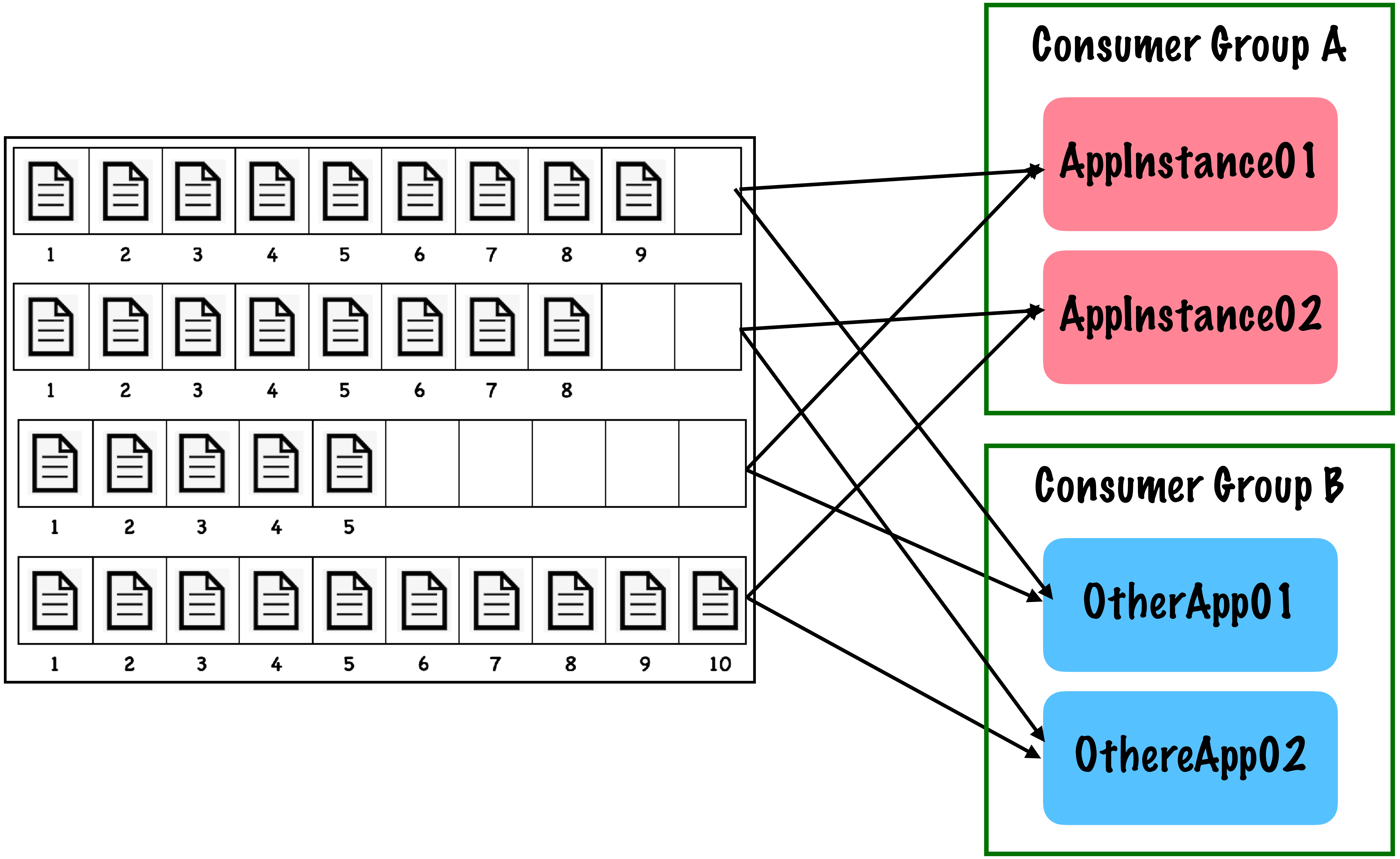

Producer
1
2
3
4
5
6
7
8
9
10
11
~~Consumer~~
~~Committed Offset: 3~~


Producer
1
2
3
4
5
6
7
8
9
10
11
Consumer
Committed Offset: 3


Producer
1
2
3
4
5
6
7
8
9
10
11
Consumer
Committed Offset: 3


Consumer Group A
AppInstance01
AppInstance02
1
2
3
4
5
6
7
8
9
1
2
3
4
5
6
7
8
1
2
3
4
5
1
2
3
4
5
6
7
8
9
10


Consumer Group A
AppInstance01
AppInstance02
AppInstance03
AppInstance04
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8
1 2 3 4 5
1 2 3 4 5 6 7 8 9 10


1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8
1 2 3 4 5
1 2 3 4 5 6 7 8 9 10
Consumer Group A
AppInstance01
AppInstance02
Consumer Group B
OtherApp01
OthereApp02

Python Kafka Clients

confluent-kafka

https://github.com/confluentinc/confluent-kafka-python

kafka-python

https://github.com/dpkp/kafka-python

aiokafka

https://github.com/aio-libs/aiokafka

Producing data to a Kafka topic

```
from confluent_kafka import Producer

my_key = "a001"
my_val = "Hello World"

producer = Producer({"bootstrap.servers": "localhost:29092"})

producer.produce("my_topic", key=my_key, value=my_val)
```

Consuming data from a Kafka topic

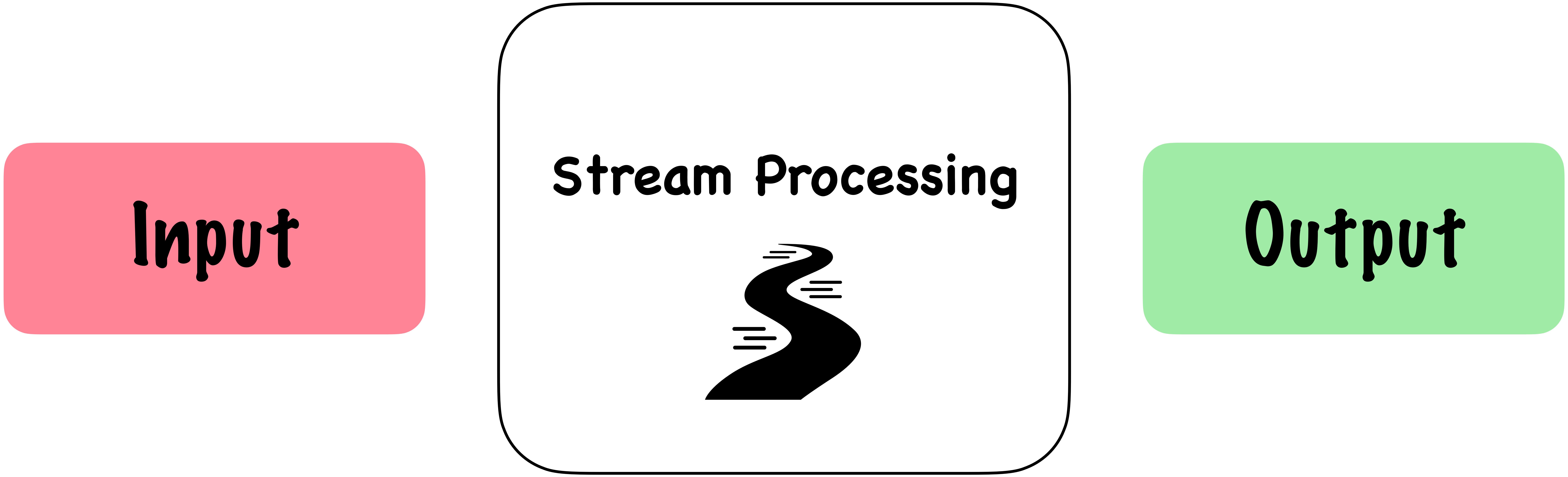
```
from confluent_kafka import Consumer

consumer = Consumer({"bootstrap.servers": "localhost:29092",
                     "group.id": "my_consumer_group"})

consumer.subscribe(["my_topic"])

while True:
    event = consumer.poll(1.0)
    if event is None:
        pass
    else:
        val = event.value().decode("utf-8")
        print(val)
```

Stream Processing



Stream Processing

Stateless Operations

Filter

```
producer = Producer(...)
consumer = Consumer(...)
consumer.subscribe(["input_topic"])

while True:
    event = consumer.poll(1.0)
    if check_predicate(event):
        producer.produce("output_topic", value=event)
```

Stream Processing

Stateless Operations

Branch

```
producer = Producer(...)
consumer = Consumer(...)
consumer.subscribe(["input_topic"])

while True:
    event = consumer.poll(1.0)
    if check_predicate_a(event):
        producer.produce("output_a", value=event)
    elif check_predicate_b(event):
        producer.produce("output_b", value=event)
    else:
        producer.produce("output_other", value=event)
```

Stream Processing

Stateless Operations

Map

```
producer = Producer(...)
consumer = Consumer(...)
consumer.subscribe(["input_topic"])

def my_func(event):
    # some process we want done on all events

while True:
    my_func(consumer.poll(1.0))
```

Stream Processing

Stateful Operations

Count

```
...
counts = {}
while True:
    event = consumer.poll(1.0)
    if event.key() in counts:
        counts[event.key()] += 1
    else
        counts[event.key() = 1
    producer.produce("output", key=event.key(),
        value=counts[event.key()])
```

Stream Processing

Stateful Operations

Sum

```
...
sums = {}
while True:
    event = consumer.poll(1.0)
    if event.key() in sums:
        sums[event.key()] += event.value()
    else
        sums[event.key()] = event.value()
    producer.produce("output", key=event.key(),
        value=sums[event.key()])
```

Stream Processing

Stateful Operations

Aggregate

```
...
aggs = {}
while True:
    event = consumer.poll(1.0)
    aggs[event.key()] = agg_func(aggs[event.key()],event.value())
    producer.produce("output", key=event.key(),
                     value=aggs[event.key()])
```

Stateful Operations

Join

```
# a_dict, b_dict, c_dict, and consumers declared above
while True:
    c_event = None
    a_event = consumer_a.poll(1.0)
    a_dict[a_event.key()] = a_event
    if a_event.key() in b_dict and a_event.key() not in c_dict:
        c_event = join_func(a_event, b_dict[a_event.key()])
        c_dict[a_event.key()] = c_event

    b_event = consumer_b.poll(1.0)
    b_dict[b_event.key()] = b_event
    if b_event.key() in a_dict and b_event.key() not in c_dict:
        c_event = join_func(a_dict[b_event.key()], b_event)
        c_dict[b_event.key()] = c_event

    if c_event is not None:
        producer.produce(c_topic, key=c_event.key(), value=c_event)
```

Stateful Operations

Join

```
# a_dict, b_dict, c_dict, and consumers declared above
while True:
    c_event = None
    a_event = consumer_a.poll(1.0)
    a_dict[a_event.key()] = a_event
    if a_event.key() in b_dict and a_event.key() not in c_dict:
        c_event = join_func(a_event, b_dict[a_event.key()])
        c_dict[a_event.key()] = c_event

    b_event = consumer_b.poll(1.0)
    b_dict[b_event.key()] = b_event
    if b_event.key() in a_dict and b_event.key() not in c_dict:
        c_event = join_func(a_dict[b_event.key()], b_event)
        c_dict[b_event.key()] = c_event

    if c_event is not None:
        producer.produce(c_topic, key=c_event.key(), value=c_event)
```

Stateful Operations

Join

```
# a_dict, b_dict, c_dict, and consumers declared above
while True:
    c_event = None
    a_event = consumer_a.poll(1.0)
    a_dict[a_event.key()] = a_event
    if a_event.key() in b_dict and a_event.key() not in c_dict:
        c_event = join_func(a_event, b_dict[a_event.key()])
        c_dict[a_event.key()] = c_event

    b_event = consumer_b.poll(1.0)
    b_dict[b_event.key()] = b_event
    if b_event.key() in a_dict and b_event.key() not in c_dict:
        c_event = join_func(a_dict[b_event.key()], b_event)
        c_dict[b_event.key()] = c_event

    if c_event is not None:
        producer.produce(c_topic, key=c_event.key(), value=c_event)
```

Stateful Operations

Join

```
# a_dict, b_dict, c_dict, and consumers declared above
while True:
    c_event = None
    a_event = consumer_a.poll(1.0)
    a_dict[a_event.key()] = a_event
    if a_event.key() in b_dict and a_event.key() not in c_dict:
        c_event = join_func(a_event, b_dict[a_event.key()])
        c_dict[a_event.key()] = c_event

    b_event = consumer_b.poll(1.0)
    b_dict[b_event.key()] = b_event
    if b_event.key() in a_dict and b_event.key() not in c_dict:
        c_event = join_func(a_dict[b_event.key()], b_event)
        c_dict[b_event.key()] = c_event

    if c_event is not None:
        producer.produce(c_topic, key=c_event.key(), value=c_event)
```

Stateful Operations

Join

```
# a_dict, b_dict, c_dict, and consumers declared above
while True:
    c_event = None
    a_event = consumer_a.poll(1.0)
    a_dict[a_event.key()] = a_event
    if a_event.key() in b_dict and a_event.key() not in c_dict:
        c_event = join_func(a_event, b_dict[a_event.key()])
        c_dict[a_event.key()] = c_event

    b_event = consumer_b.poll(1.0)
    b_dict[b_event.key()] = b_event
    if b_event.key() in a_dict and b_event.key() not in c_dict:
        c_event = join_func(a_dict[b_event.key()], b_event)
        c_dict[b_event.key()] = c_event

    if c_event is not None:
        producer.produce(c_topic, key=c_event.key(), value=c_event)
```

Stateful Operations

Join

```
# a_dict, b_dict, c_dict, and consumers declared above
while True:
    c_event = None
    a_event = consumer_a.poll(1.0)
    a_dict[a_event.key()] = a_event
    if a_event.key() in b_dict and a_event.key() not in c_dict:
        c_event = join_func(a_event, b_dict[a_event.key()])
        c_dict[a_event.key()] = c_event

    b_event = consumer_b.poll(1.0)
    b_dict[b_event.key()] = b_event
    if b_event.key() in a_dict and b_event.key() not in c_dict:
        c_event = join_func(a_dict[b_event.key()], b_event)
        c_dict[b_event.key()] = c_event

    if c_event is not None:
        producer.produce(c_topic, key=c_event.key(), value=c_event)
```

Stateful Operations

Join

```
# a_dict, b_dict, c_dict, and consumers declared above
while True:
    c_event = None
    a_event = consumer_a.poll(1.0)
    a_dict[a_event.key()] = a_event
    if a_event.key() in b_dict and a_event.key() not in c_dict:
        c_event = join_func(a_event, b_dict[a_event.key()])
        c_dict[a_event.key()] = c_event

    b_event = consumer_b.poll(1.0)
    b_dict[b_event.key()] = b_event
    if b_event.key() in a_dict and b_event.key() not in c_dict:
        c_event = join_func(a_dict[b_event.key()], b_event)
        c_dict[b_event.key()] = c_event

    if c_event is not None:
        producer.produce(c_topic, key=c_event.key(), value=c_event)
```

Stateful Operations

Join

```
# a_dict, b_dict, c_dict, and consumers declared above
while True:
    c_event = None
    a_event = consumer_a.poll(1.0)
    a_dict[a_event.key()] = a_event
    if a_event.key() in b_dict and a_event.key() not in c_dict:
        c_event = join_func(a_event, b_dict[a_event.key()])
        c_dict[a_event.key()] = c_event

    b_event = consumer_b.poll(1.0)
    b_dict[b_event.key()] = b_event
    if b_event.key() in a_dict and b_event.key() not in c_dict:
        c_event = join_func(a_dict[b_event.key()], b_event)
        c_dict[b_event.key()] = c_event

    if c_event is not None:
        producer.produce(c_topic, key=c_event.key(), value=c_event)
```

Stateful Operations

Join

```
# a_dict, b_dict, c_dict, and consumers declared above
while True:
    c_event = None
    a_event = consumer_a.poll(1.0)
    a_dict[a_event.key()] = a_event
    if a_event.key() in b_dict and a_event.key() not in c_dict:
        c_event = join_func(a_event, b_dict[a_event.key()])
        c_dict[a_event.key()] = c_event

    b_event = consumer_b.poll(1.0)
    b_dict[b_event.key()] = b_event
    if b_event.key() in a_dict and b_event.key() not in c_dict:
        c_event = join_func(a_dict[b_event.key()], b_event)
        c_dict[b_event.key()] = c_event

    if c_event is not None:
        producer.produce(c_topic, key=c_event.key(), value=c_event)
```

Stream Processing Challenges

State

Scale

Python Event Streaming Libraries

daveklein@usa.net

@daveklein

Python Event Streaming Libraries

FAUST https://github.com/faust-streaming/faust

bytewax https://github.com/bytewax/bytewax

Quix Streams https://github.com/quixio/quix-streams