

Apache Kafka

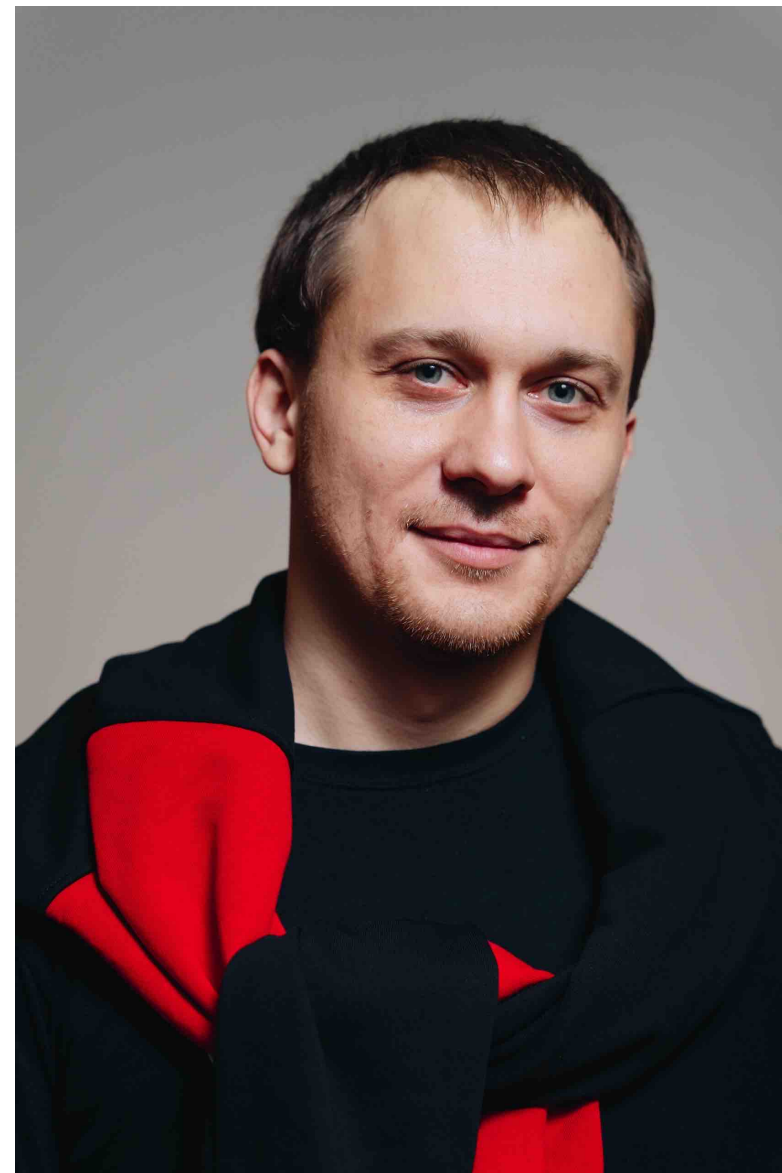
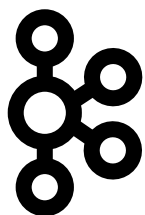
погружение на 45 минут

Григорий Кошелев
Контур

Григорий Кошелев

- 10+ лет опыта создания Messaging и Stream Processing систем
- 5+ лет опыта работы с Apache Kafka
- 10+ докладов и выступлений про Apache Kafka

Основной стек Java/JVM



Трилогия «Когда всё пошло по Кафке»

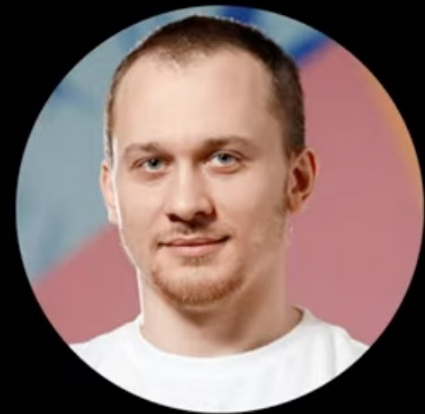


https://www.youtube.com/watch?v=A_yUaPARv8U

Трилогия «Когда всё пошло по Кафке»



Когда всё пошло
по Кафке 2:
Разгоняем продьюсеров



Григорий Кошелев
Контур



Трилогия «Когда всё пошло по Kafka»

 JPoint 2023

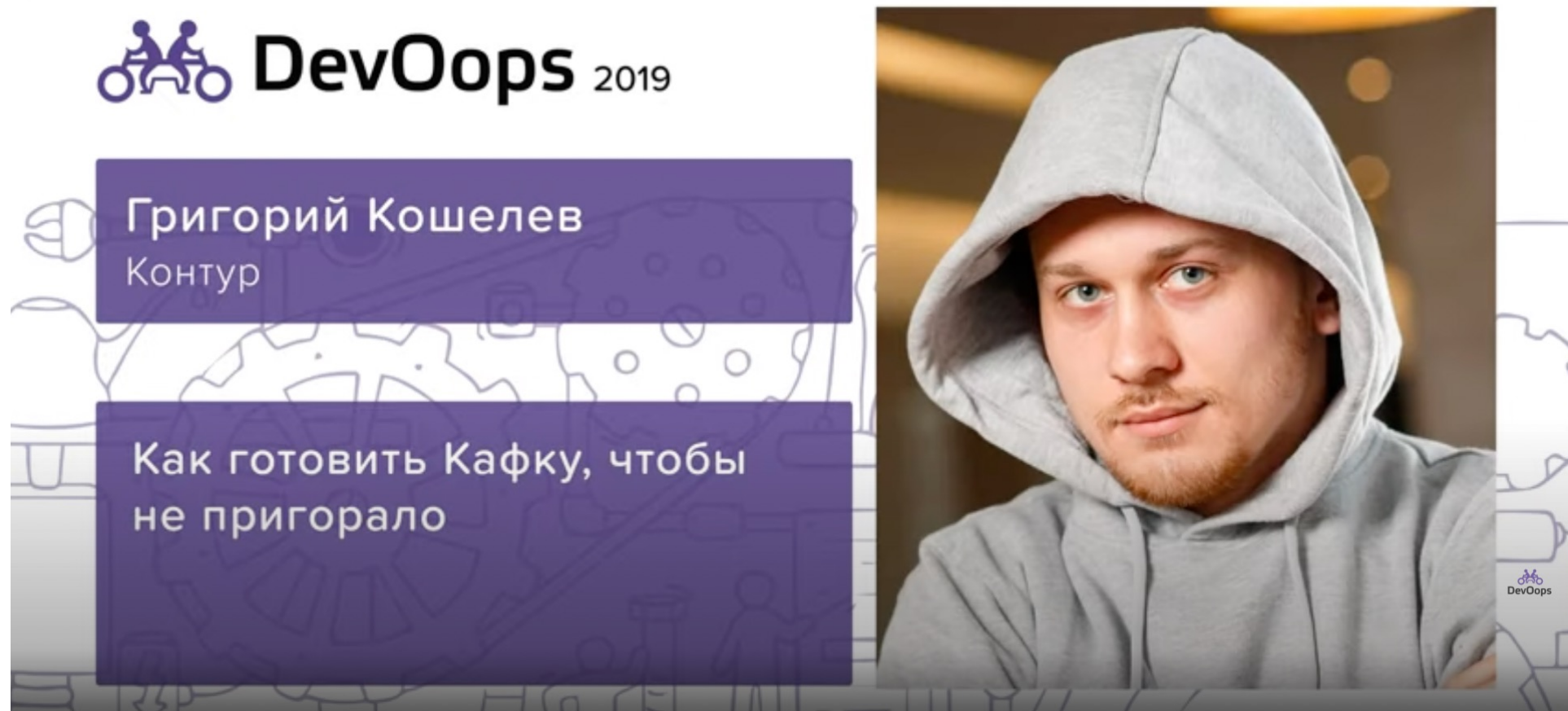
**Когда всё пошло
по Kafka 3:
где заканчивается
Apache Kafka и начинается
работу Consumer**



**Григорий
Кошелев**

Контур

Как готовить Кафку, чтобы не пригорало




<https://www.youtube.com/watch?v=M3HTM81P-Sg>

Apache Kafka для .NET-разработчиков

DOTNEXT 2022

**Kafka: от теории
к практике**



**Григорий
Кошелев**
Контур

Про что доклад

1. Введение в Apache Kafka
2. Kafka Producer
3. Kafka Consumer
4. Когда Apache Kafka 🔥
5. Подводные камни и ограничения

Чего не будет в докладе

- Kafka Streams
- KSQL
- Kafka Connect
- Mirror Maker
- Transactions

Кафка в Контуре

Около 20 кластеров Apache Kafka

Кафка в Контуре

Около 20 кластеров Apache Kafka

По типу окружения

Кафка в Контуре

Около 20 кластеров Apache Kafka

По типу окружения

— Тестовое

Кафка в Контуре

Около 20 кластеров Apache Kafka

По типу окружения

— Тестовое

- * функ. тестирование

- * нагрузочное тестирование

Кафка в Контуре

Около 20 кластеров Apache Kafka

По типу окружения

— Тестовое

- * функ. тестирование

- * нагрузочное тестирование

— Стейджинг

Кафка в Контуре

Около 20 кластеров Apache Kafka

По типу окружения

- Тестовое

 - * функ. тестирование

 - * нагрузочное тестирование

- Стейджинг

- Продакшен

Кафка в Контуре

Около 20 кластеров Apache Kafka

По типу окружения

— Тестовое

* функ. тестирование

* нагрузочное тестирование

— Стейджинг

— Продакшен

По назначению

Кафка в Контуре

Около 20 кластеров Apache Kafka

По типу окружения

— Тестовое

* функ. тестирование

* нагрузочное тестирование

— Стейджинг

— Продакшен

По назначению

— Телеметрия

Кафка в Контуре

Около 20 кластеров Apache Kafka

По типу окружения

— Тестовое

* функ. тестирование

* нагрузочное тестирование

— Стейджинг

— Продакшен

По назначению

— Телеметрия

— Интеграционная шина

Кафка в Контуре

Около 20 кластеров Apache Kafka

По типу окружения

— Тестовое

* функ. тестирование

* нагрузочное тестирование

— Стейджинг

— Продакшен

По назначению

— Телеметрия

— Интеграционная шина

— Шина данных

Кафка в Контуре

Около 20 кластеров Apache Kafka

По типу окружения

— Тестовое

* функ. тестирование

* нагрузочное тестирование

— Стейджинг

— Продакшен

По назначению

— Телеметрия

— Интеграционная шина

— Шина данных

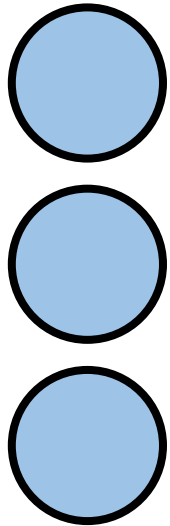
— **K**-архитектура

Введение в Apache Kafka

Введение в Apache Kafka

Kafka Producer

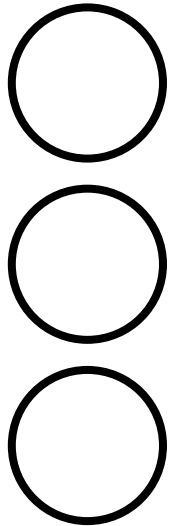
Producer



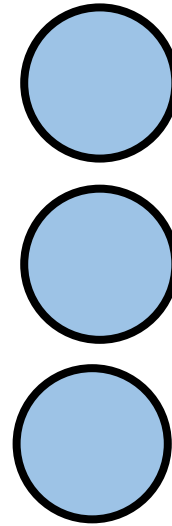
Введение в Apache Kafka

Kafka Consumer

Producer

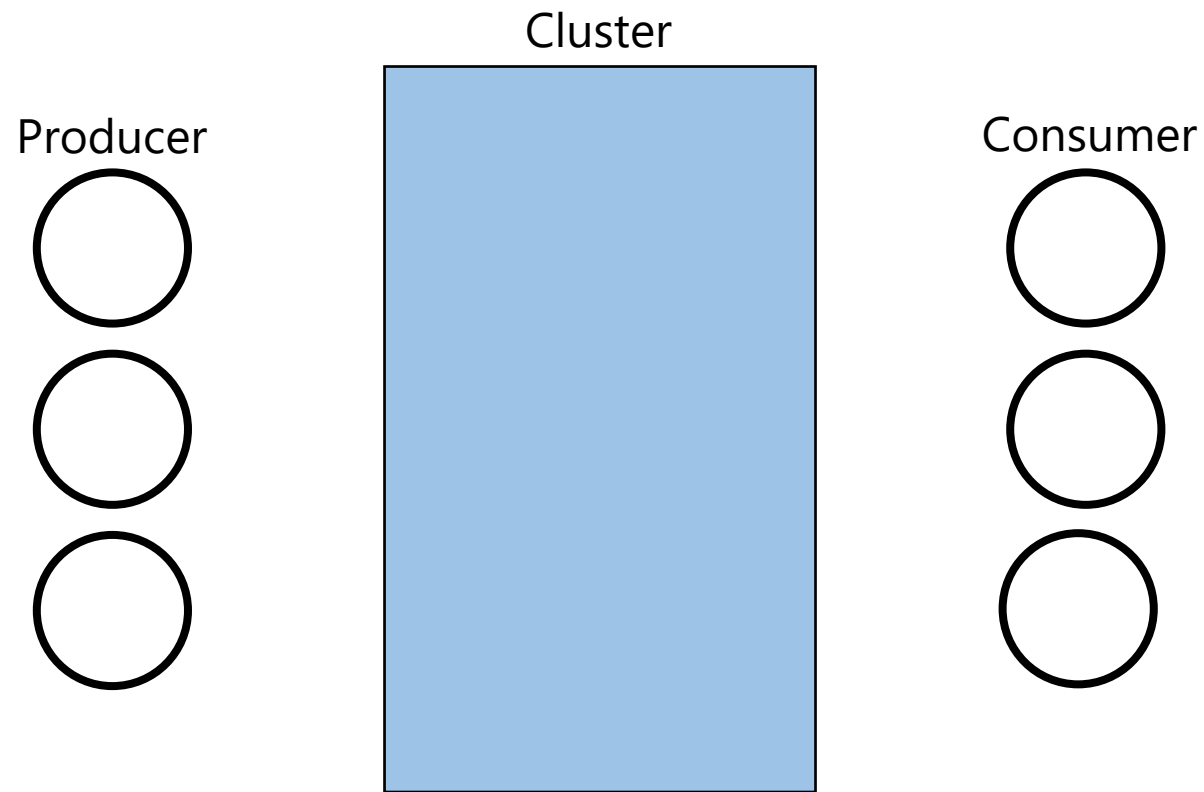


Consumer



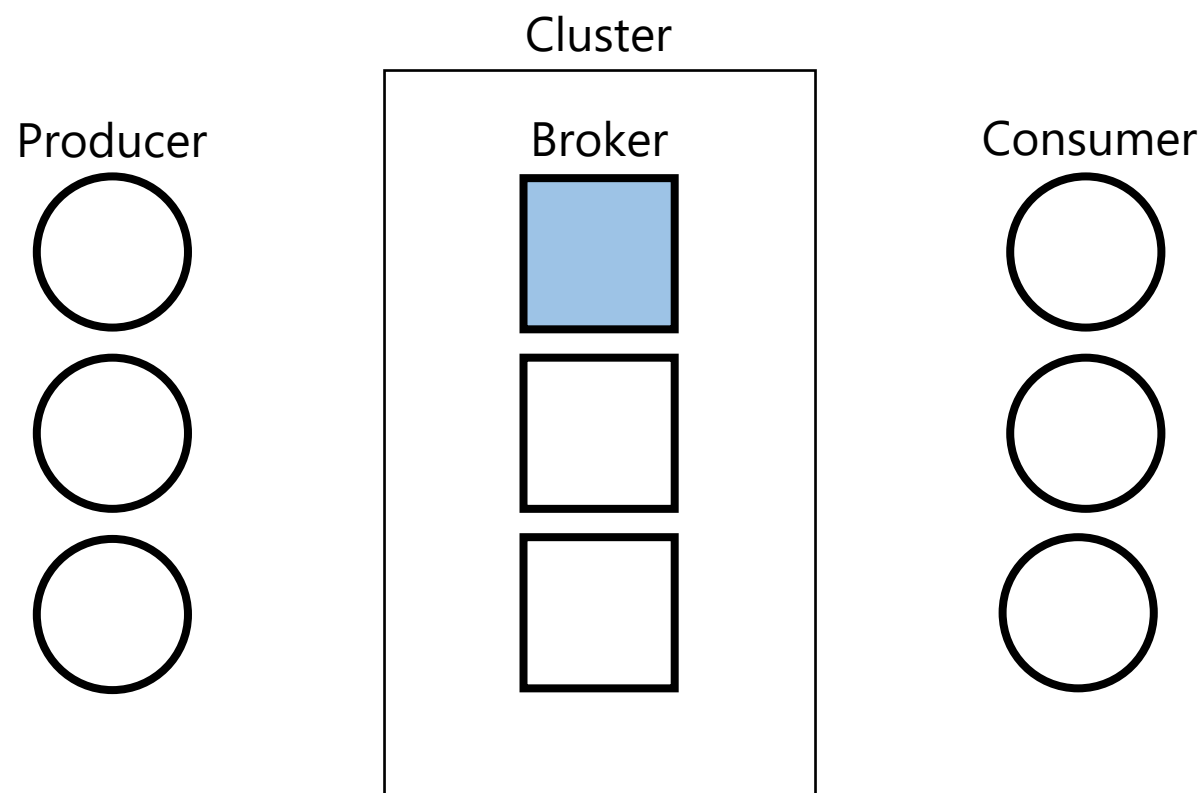
Введение в Apache Kafka

Kafka Cluster

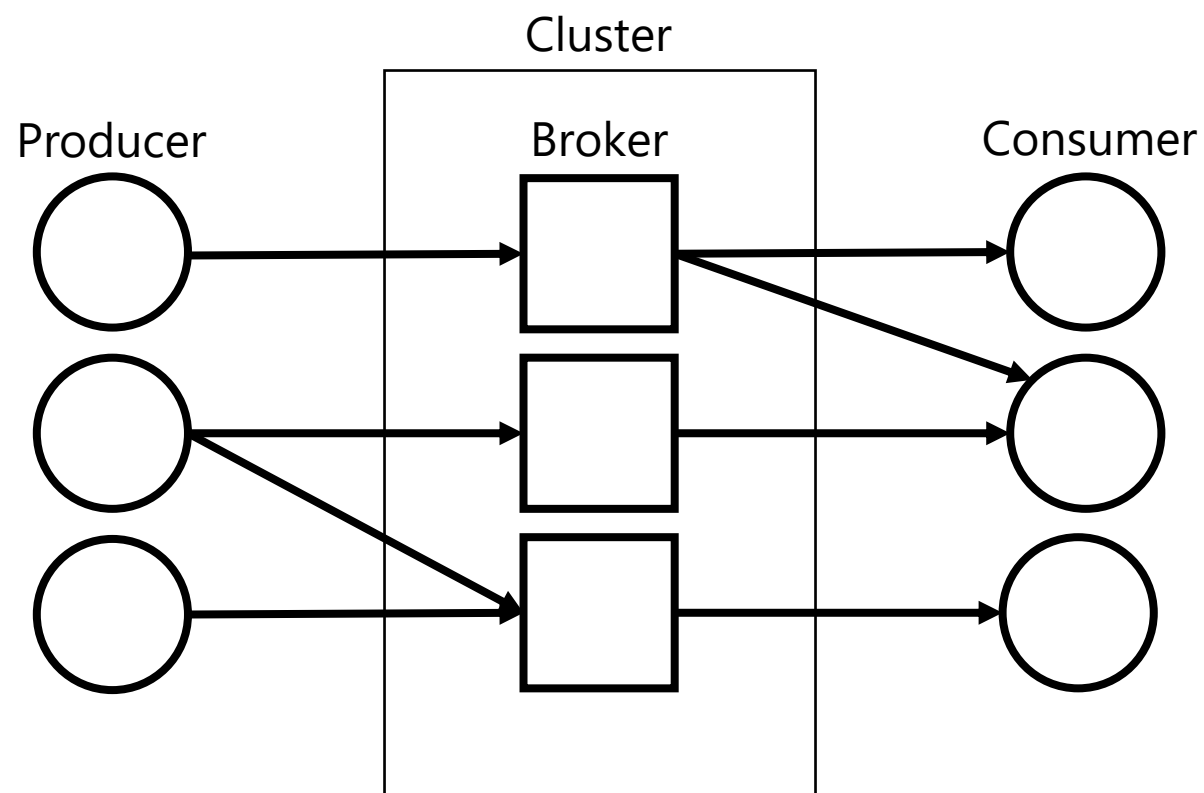


Введение в Apache Kafka

Kafka Broker

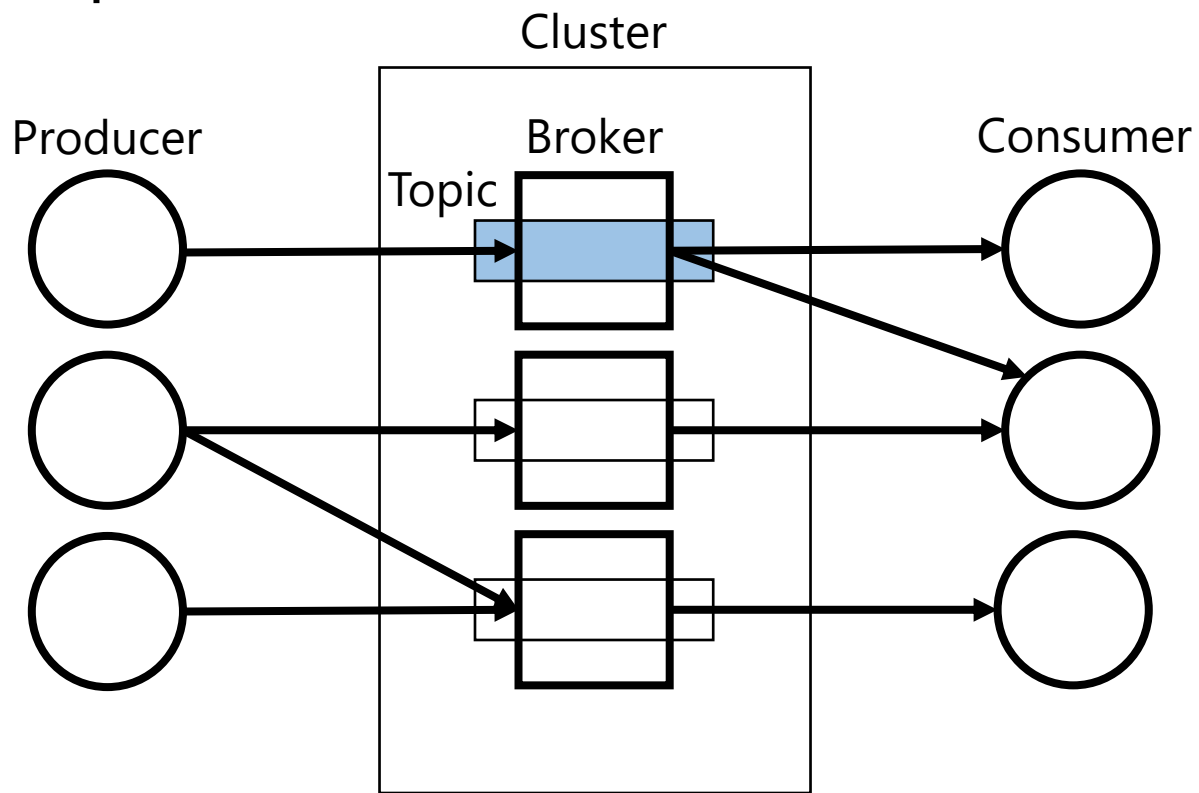


Введение в Apache Kafka

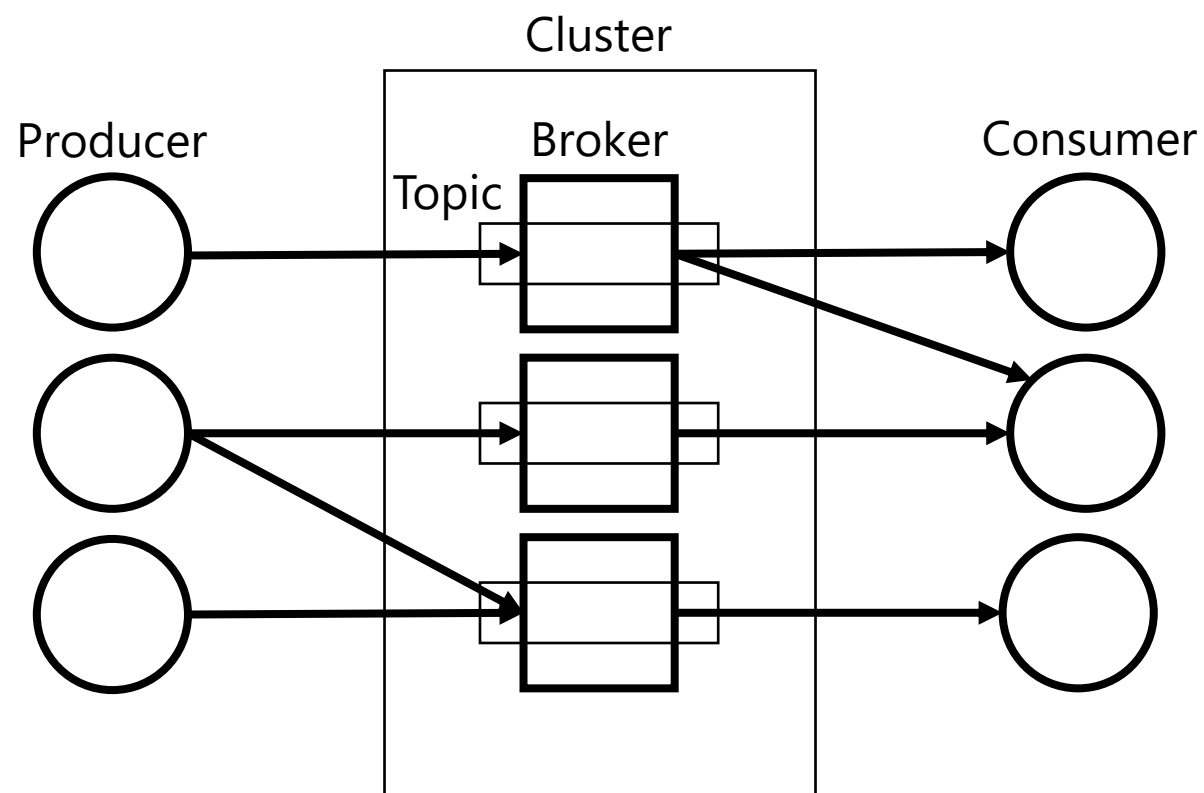


Введение в Apache Kafka

Kafka Topic

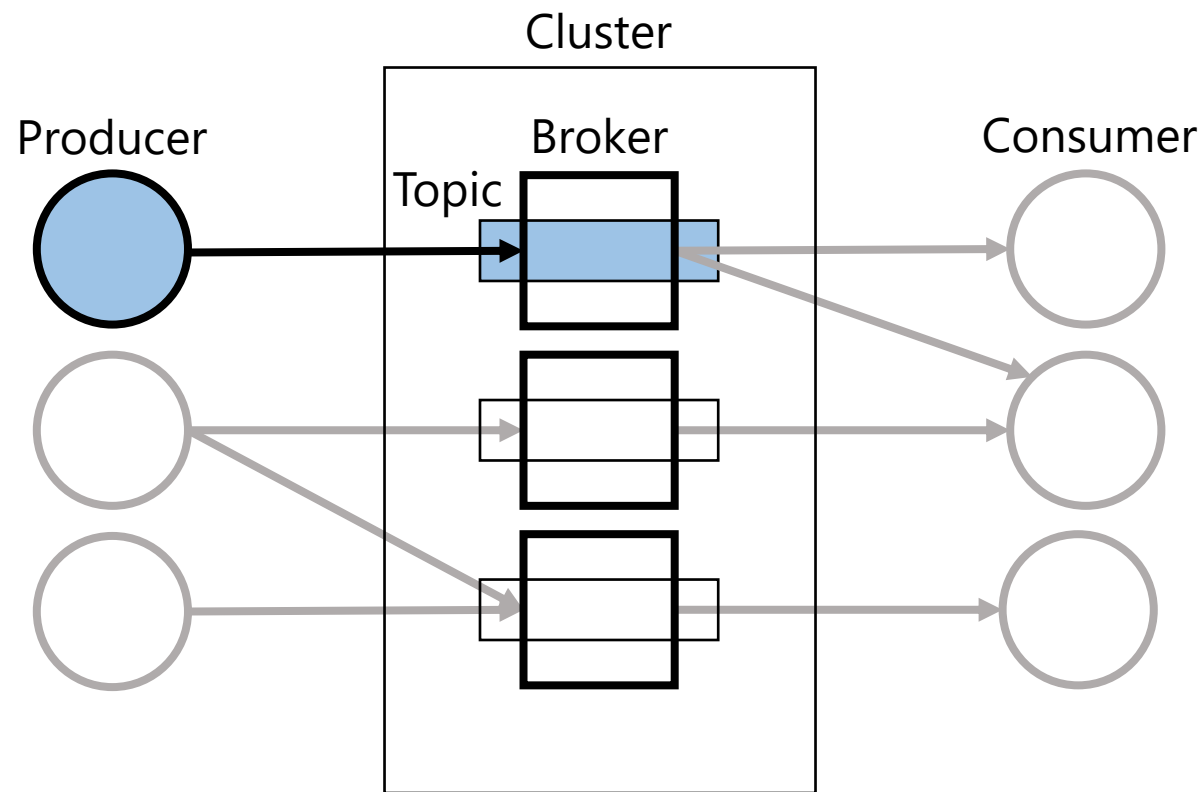


Введение в Apache Kafka



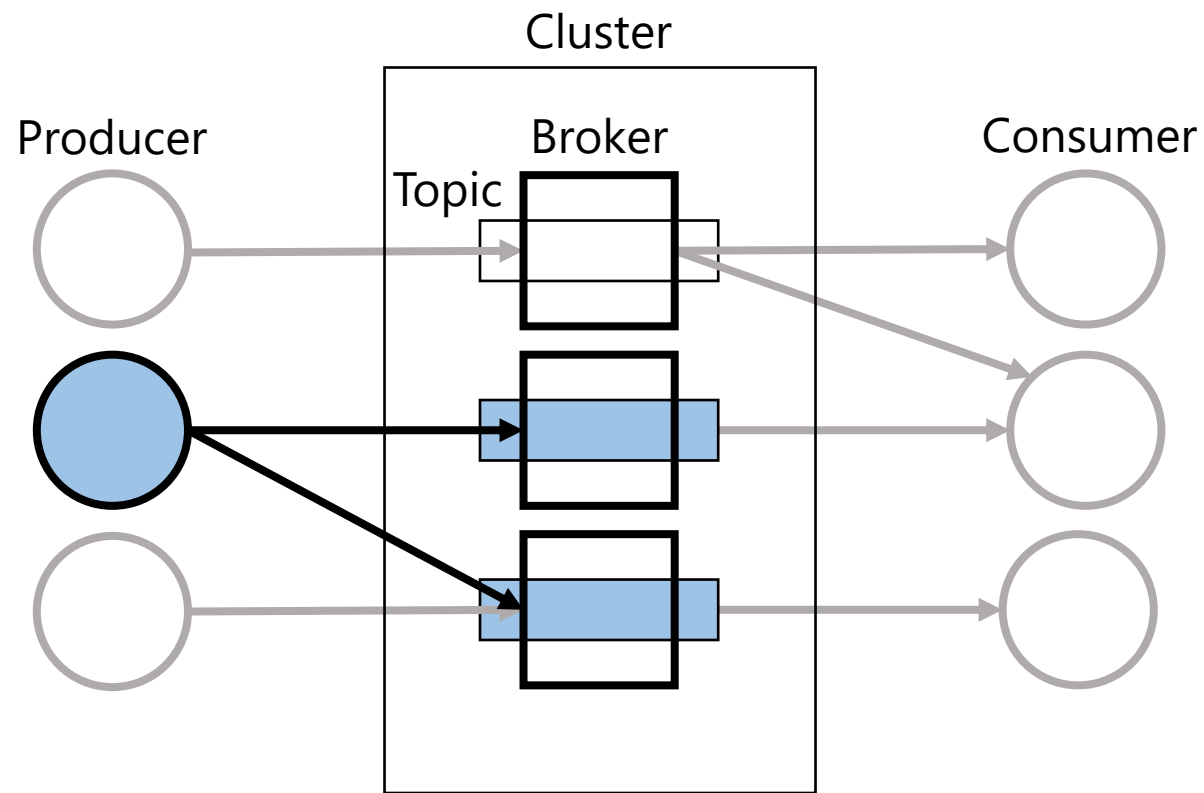
Введение в Apache Kafka

Отправка 1-to-1



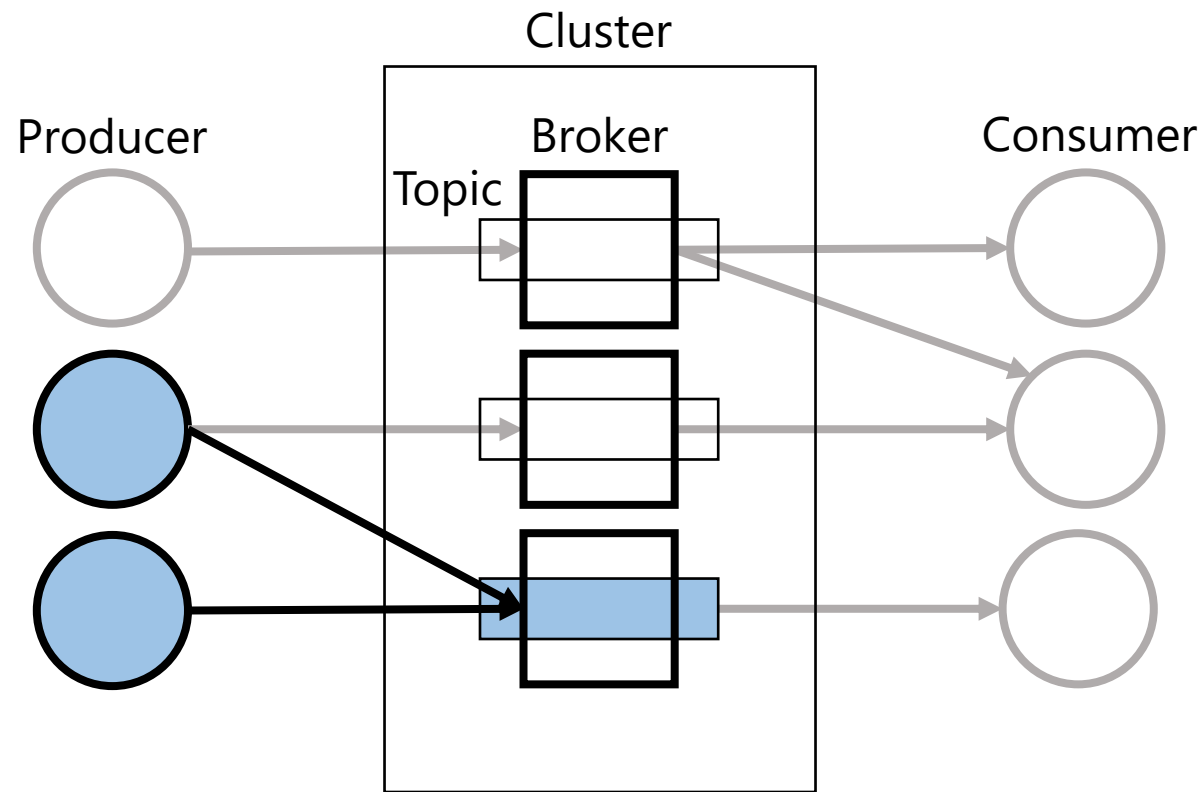
Введение в Apache Kafka

Отправка 1-to-N

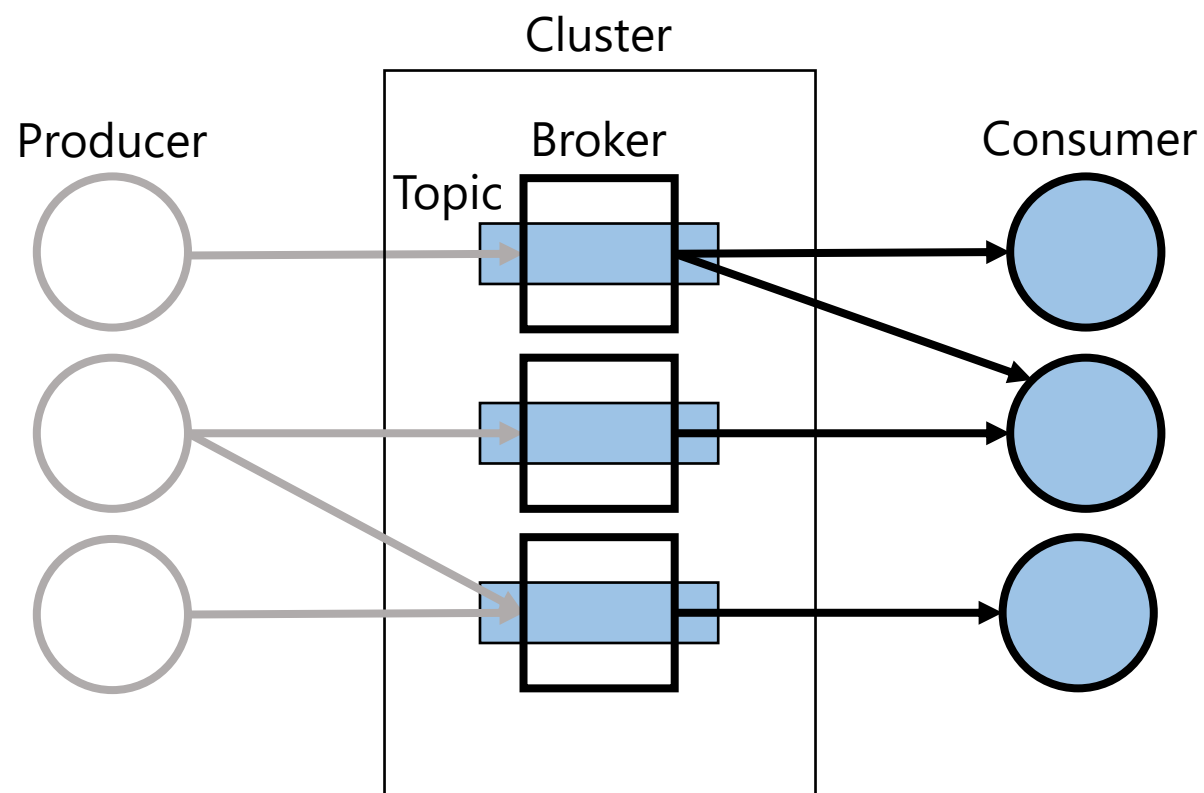


Введение в Apache Kafka

Отправка N-to-1

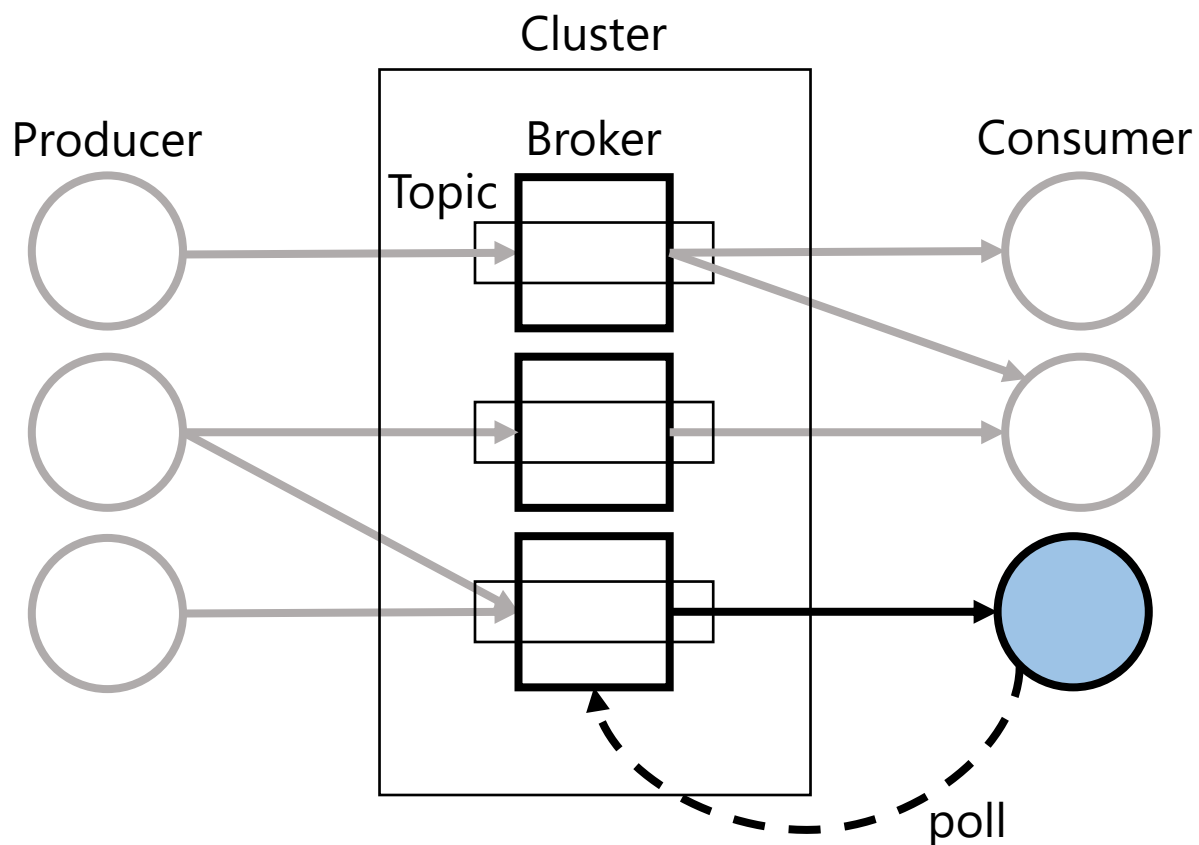


Введение в Apache Kafka



Введение в Apache Kafka

Roll-механика чтения



Архитектура Apache Kafka

- Topic
- Broker
- Producer
- Consumer

Архитектура Kafka Topic

topic = {partition}

partition 0

0	1	2	3	4	5
---	---	---	---	---	---

partition 1

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

partition 2

0	1	2	3	4	5	6
---	---	---	---	---	---	---

Архитектура Kafka Topic

topic = {partition}

partition 0

0	1	2	3	4	5	6
---	---	---	---	---	---	---

partition 1

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

partition 2

0	1	2	3	4	5	6
---	---	---	---	---	---	---

Архитектура Kafka Topic

topic = {partition}

partition 0

0	1	2	3	4	5	6
---	---	---	---	---	---	---

partition 1

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

partition 2

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

Архитектура Kafka Topic

topic = {partition}

partition 0

0	1	2	3	4	5	6
---	---	---	---	---	---	---

partition 1

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

partition 2

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

Архитектура Kafka Topic

topic = {partition}

partition 0

0	1	2	3	4	5	6
---	---	---	---	---	---	---

 offset = **0**

partition 1

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

partition 2

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

Архитектура Kafka Topic

topic = {partition}

partition 0

0	1	2	3	4	5	6
---	---	---	---	---	---	---

 offset = **1**

partition 1

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

partition 2

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

Архитектура Kafka Topic

topic = {partition}

partition 0

0	1	2	3	4	5	6
---	---	---	---	---	---	---

 offset = **2**

partition 1

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

partition 2

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

Архитектура Kafka Topic

topic = {partition}

partition 0

0	1	2	3	4	5	6
---	---	---	---	---	---	---

 offset = **3**

partition 1

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

partition 2

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

Архитектура Kafka Topic

partition = {segment}

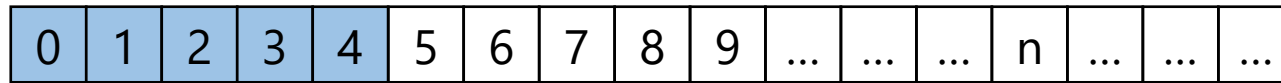
Архитектура Kafka Topic

partition = {segment}

0	1	2	3	4	5	6	7	8	9	n
---	---	---	---	---	---	---	---	---	---	-----	-----	-----	---	-----	-----	-----

Архитектура Kafka Topic

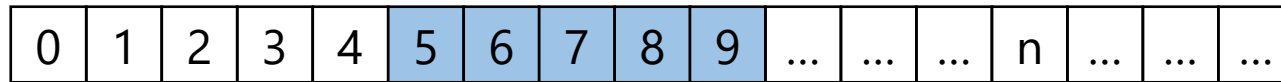
partition = {segment}



segment

Архитектура Kafka Topic

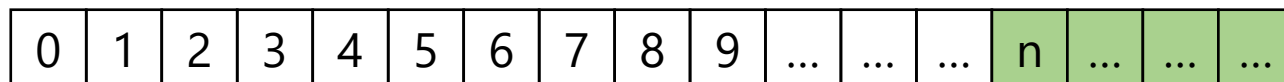
partition = {segment}



segment

Архитектура Kafka Topic

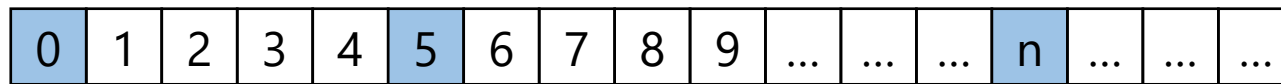
partition = {segment}



segment

Архитектура Kafka Topic

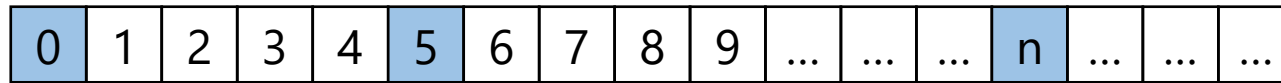
partition = {segment}



base offset

Архитектура Kafka Topic

partition = {segment}

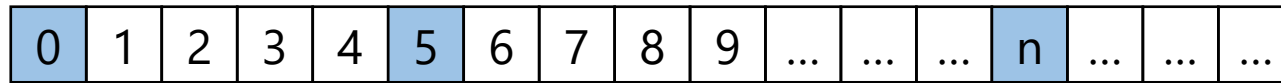


`retention.bytes=-1`

`retention.ms=604800000`

Архитектура Kafka Topic

partition = {segment}

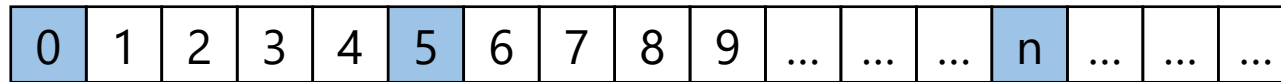


retention.bytes=-1

retention.ms=604800000

Архитектура Kafka Topic

partition = {segment}

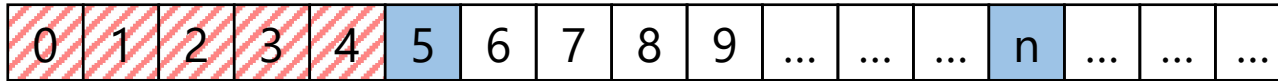


retention.bytes=-1

retention.ms=604800000

Архитектура Kafka Topic

partition = {segment}

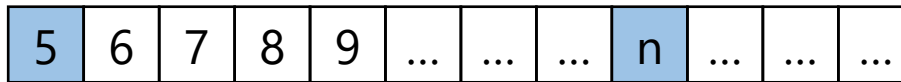


`retention.bytes=-1`

`retention.ms=604800000`

Архитектура Kafka Topic

partition = {segment}



`retention.bytes=-1`

`retention.ms=604800000`

Архитектура Kafka Topic

segment = (base_offset, data, index, timeindex)

00000000001234567890.log

00000000001234567890.index

00000000001234567890.timeindex

Архитектура Kafka Topic

segment = (**base_offset**, data, index, timeindex)

000000000001234567890.log

000000000001234567890.index

000000000001234567890.timeindex

Архитектура Kafka Topic

segment = (base_offset, **data**, index, timeindex)

00000000001234567890.log

00000000001234567890.index

00000000001234567890.timeindex

Архитектура Kafka Topic

segment = (base_offset, **data**, index, timeindex)

00000000001234567890.log

00000000001234567890.index

00000000001234567890.timeindex



log

Архитектура Kafka Topic

segment = (base_offset, data, **index**, timeindex)

00000000001234567890.log

00000000001234567890.index

00000000001234567890.timeindex



log

Index record = (relative offset, position)

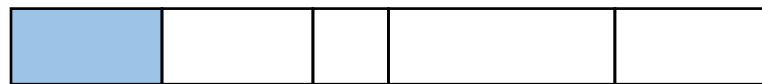
Архитектура Kafka Topic

segment = (base_offset, data, **index**, timeindex)

00000000001234567890.log

00000000001234567890.index

00000000001234567890.timeindex



log

Index record = (relative offset, position)

offset = 123456789**0** relative offset = **0**

size = 100 position = **0**

Архитектура Kafka Topic

segment = (base_offset, data, **index**, timeindex)

00000000001234567890.log

00000000001234567890.index

00000000001234567890.timeindex



log

Index record = (relative offset, position)

offset = 123456789**1** relative offset = **1**

size = 100 position = **100**

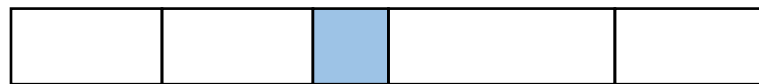
Архитектура Kafka Topic

segment = (base_offset, data, **index**, timeindex)

00000000001234567890.log

00000000001234567890.index

00000000001234567890.timeindex



log

Index record = (relative offset, position)

offset = 123456789**2** relative offset = **2**

size = 50 position = **200**

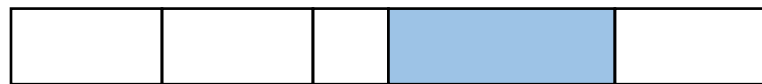
Архитектура Kafka Topic

segment = (base_offset, data, **index**, timeindex)

00000000001234567890.log

00000000001234567890.index

00000000001234567890.timeindex



log

Index record = (relative offset, position)

offset = 123456789**3** relative offset = **3**

size = 150 position = **250**

Архитектура Kafka Topic

segment = (base_offset, data, index, **timeindex**)

00000000001234567890.log

00000000001234567890.index

00000000001234567890.**timeindex**

Архитектура Kafka Topic

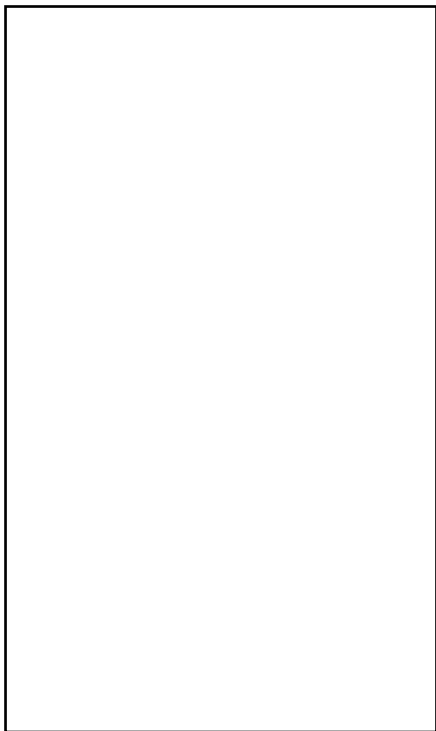
Выводы

- Топик разбит на партиции
- Партиции хранятся на диске
- Данные удаляются либо по времени, либо по размеру
- Сообщение можно быстро найти по его Offset

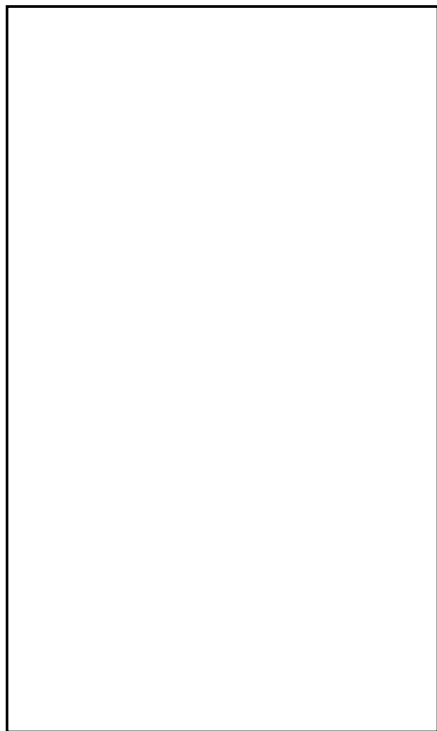
Архитектура Kafka Broker

cluster = {broker}

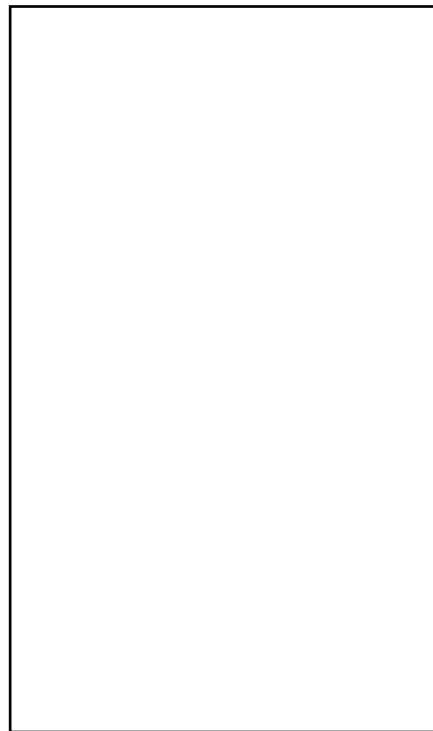
broker 1



broker 2

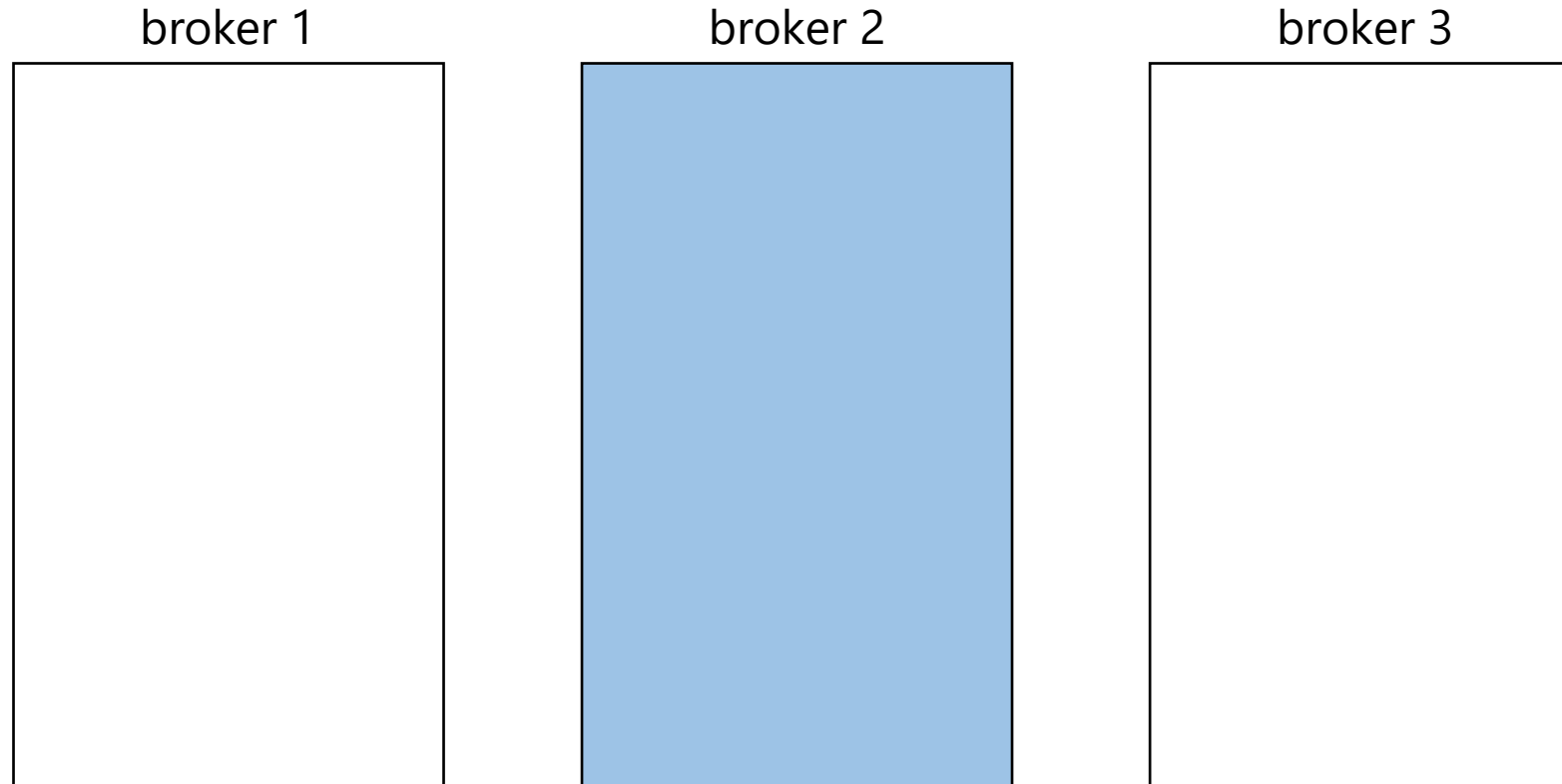


broker 3



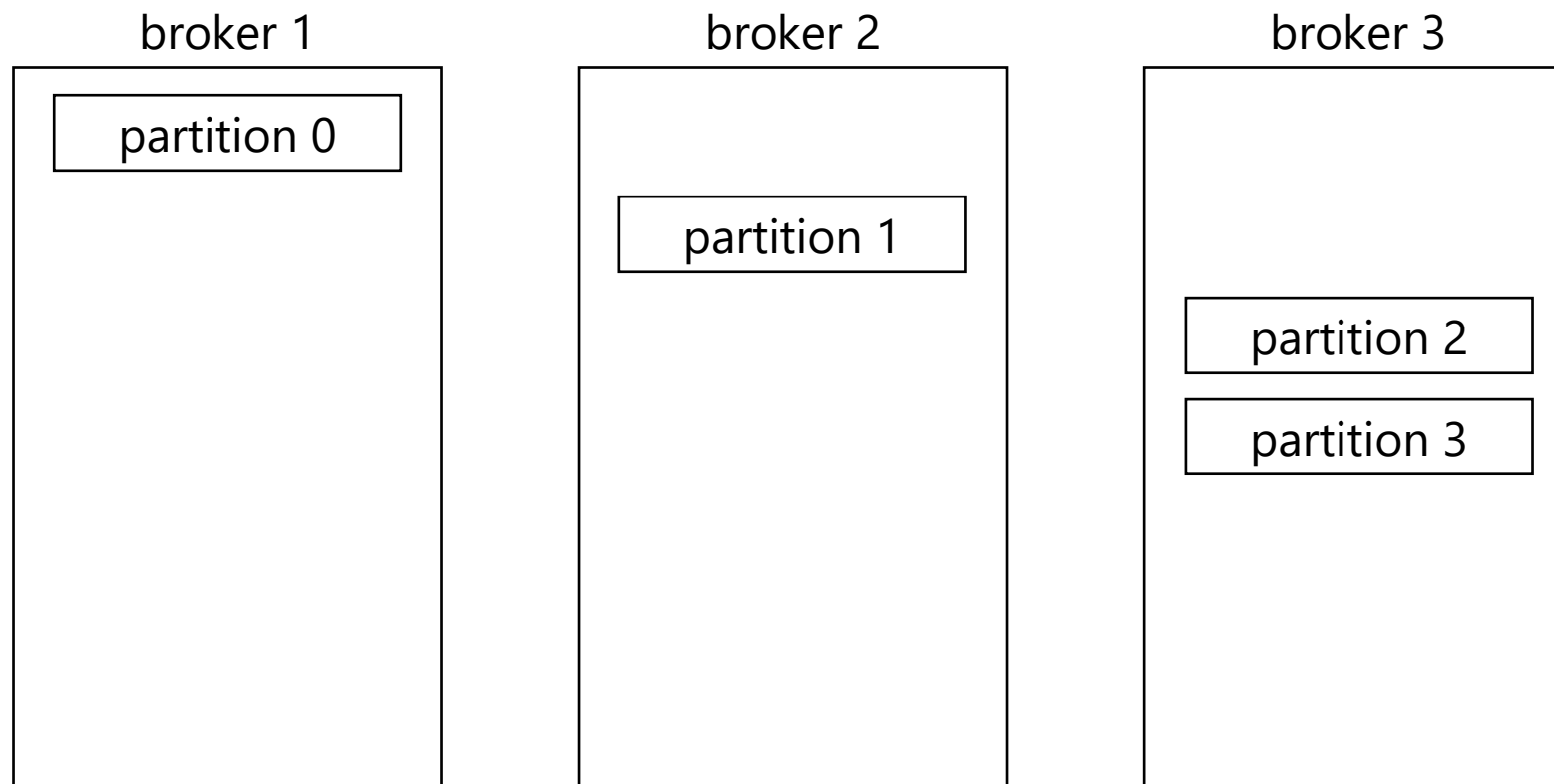
Архитектура Kafka Broker

Controller – координирует работу кластера



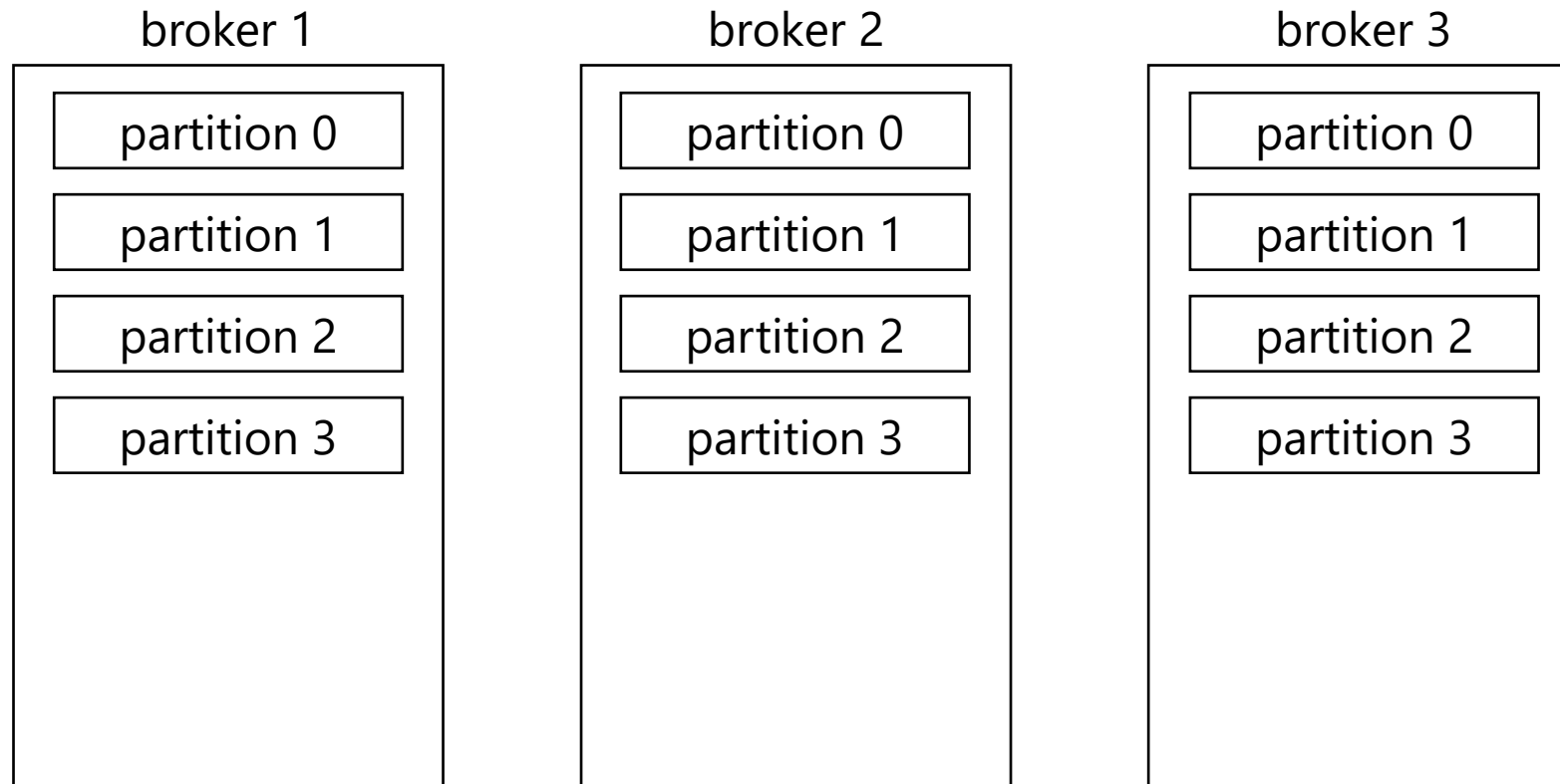
Архитектура Kafka Broker

topic = {partition}



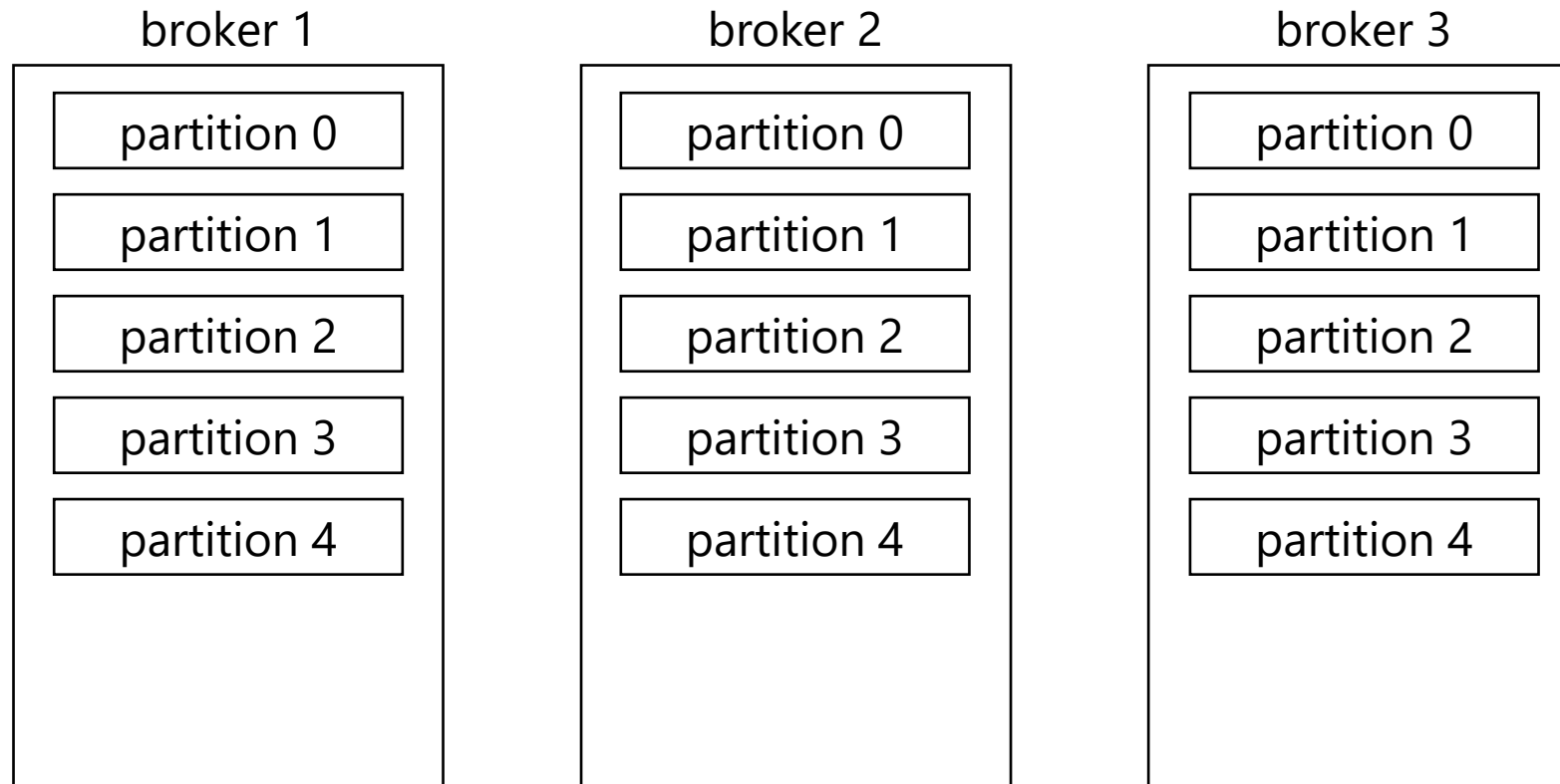
Архитектура Kafka Broker

replication factor = 3



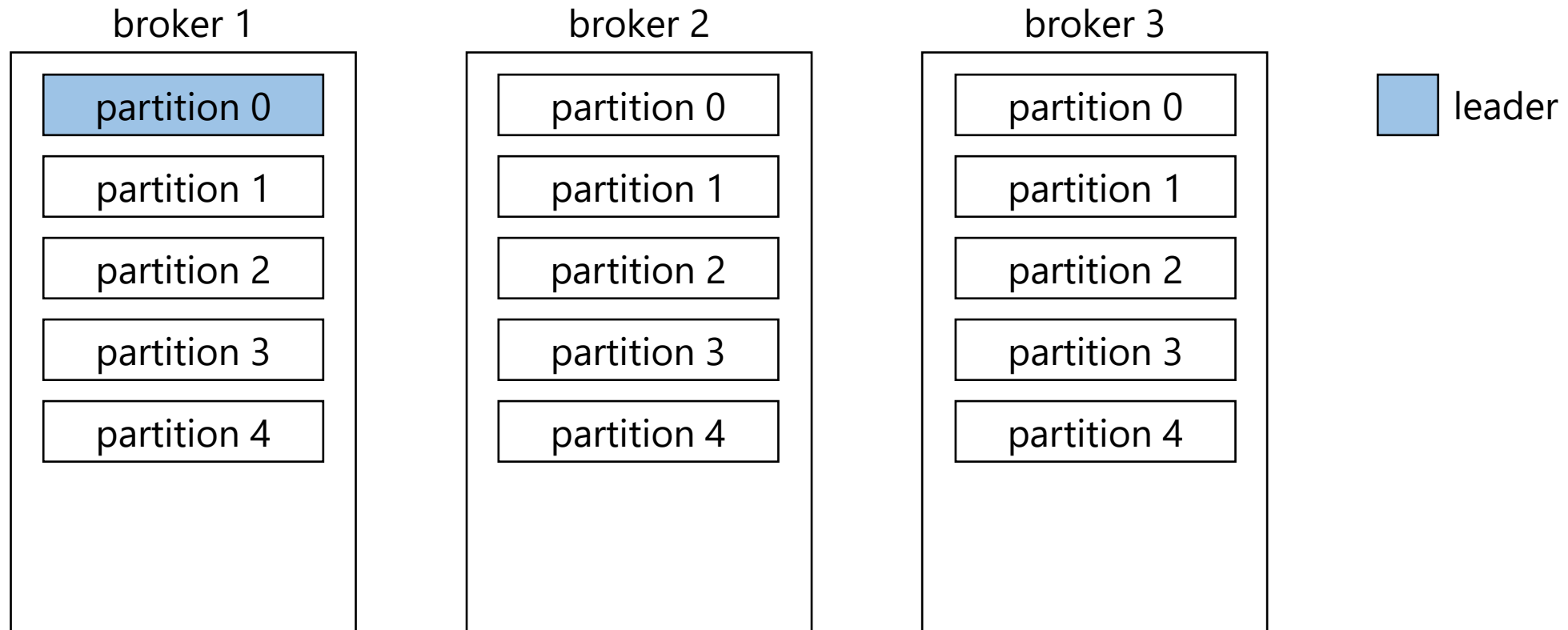
Архитектура Kafka Broker

Добавление partition



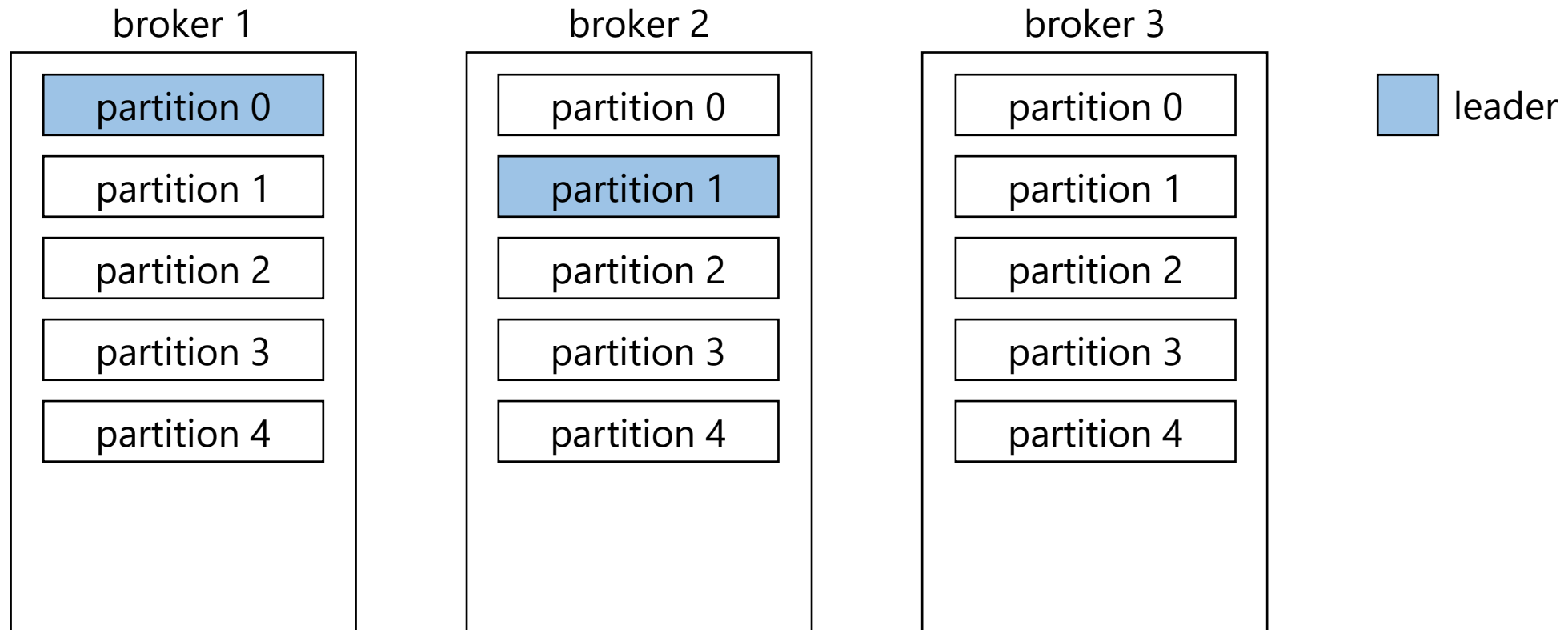
Архитектура Kafka Broker

broker 1 – leader для partition 0.



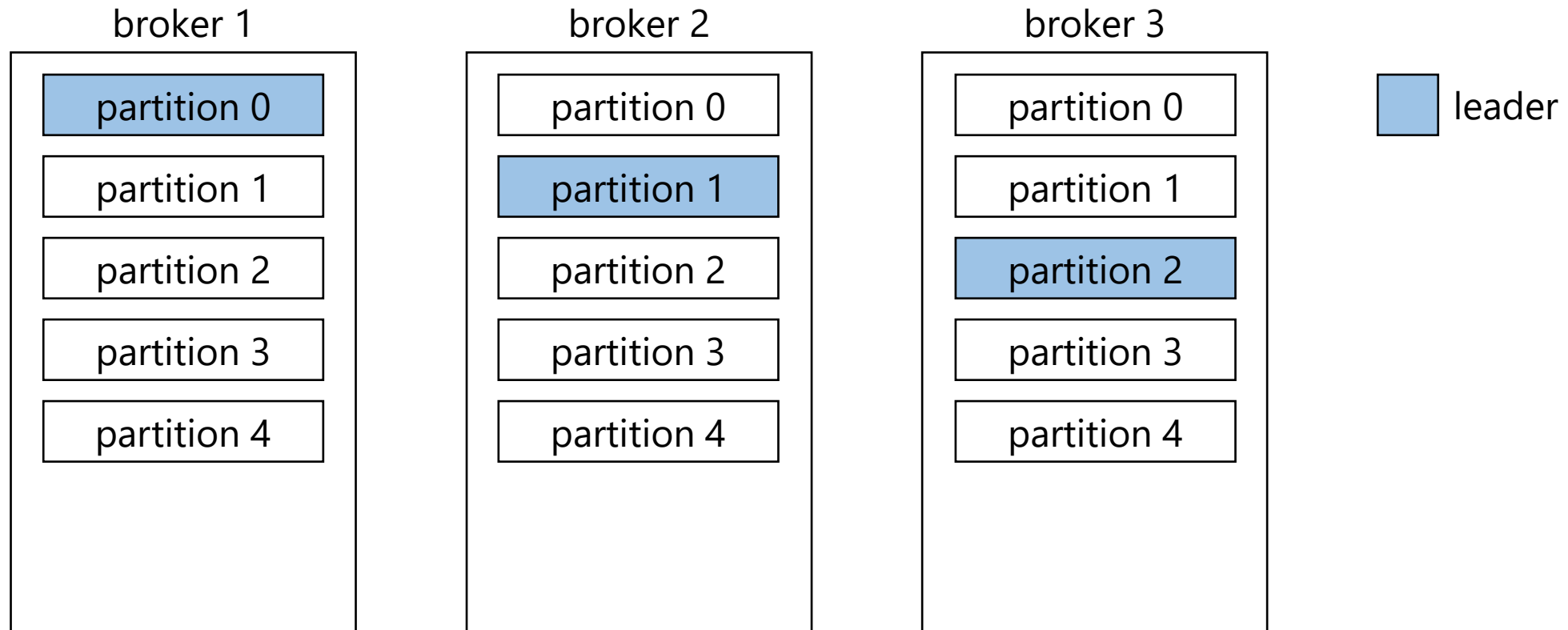
Архитектура Kafka Broker

broker 2 – leader для partition 1



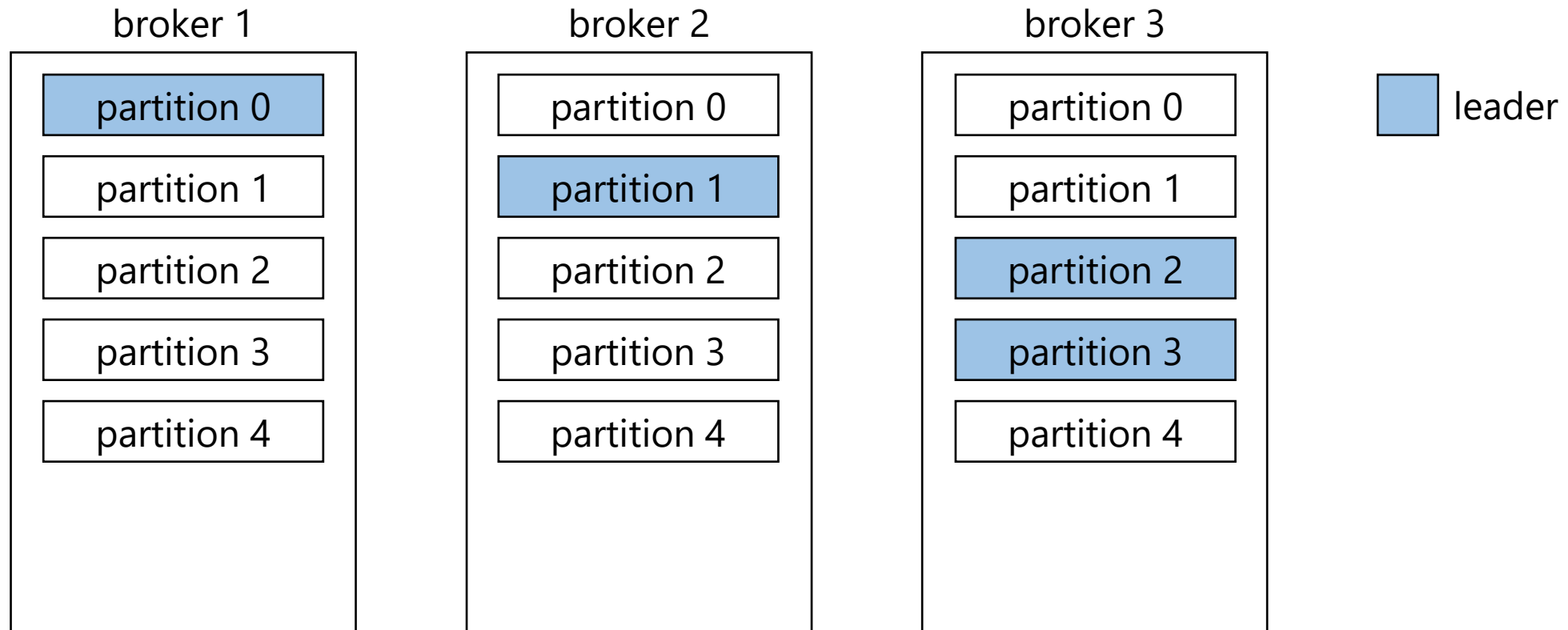
Архитектура Kafka Broker

broker 3 – leader для partition 2



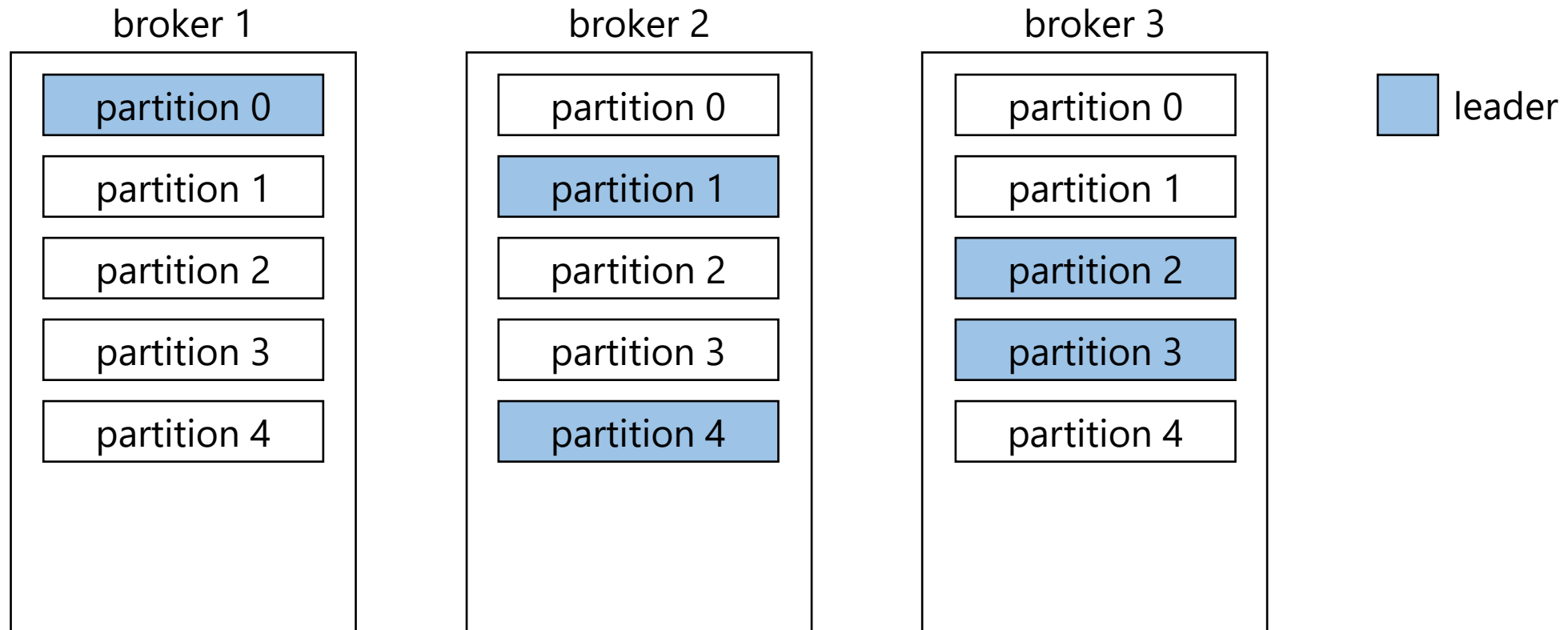
Архитектура Kafka Broker

broker 3 – leader для partition 3



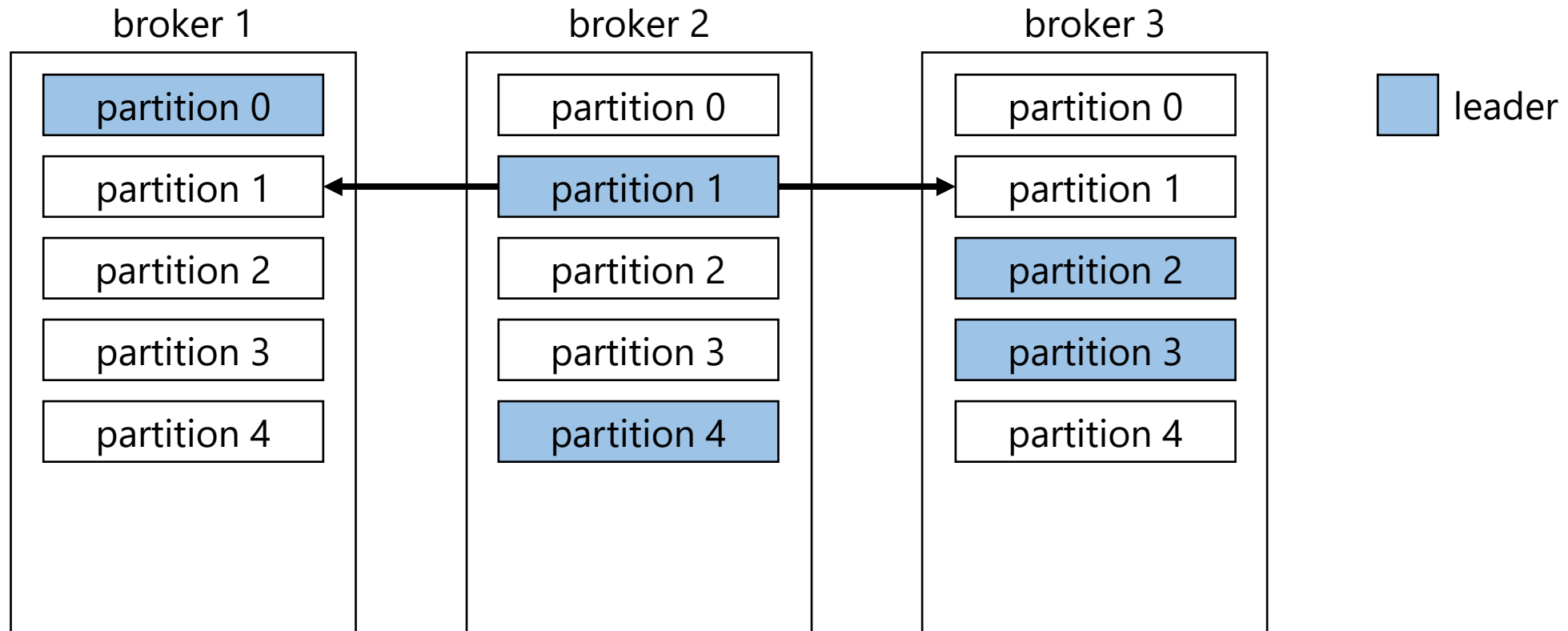
Архитектура Kafka Broker

broker 2 – leader для partition 4



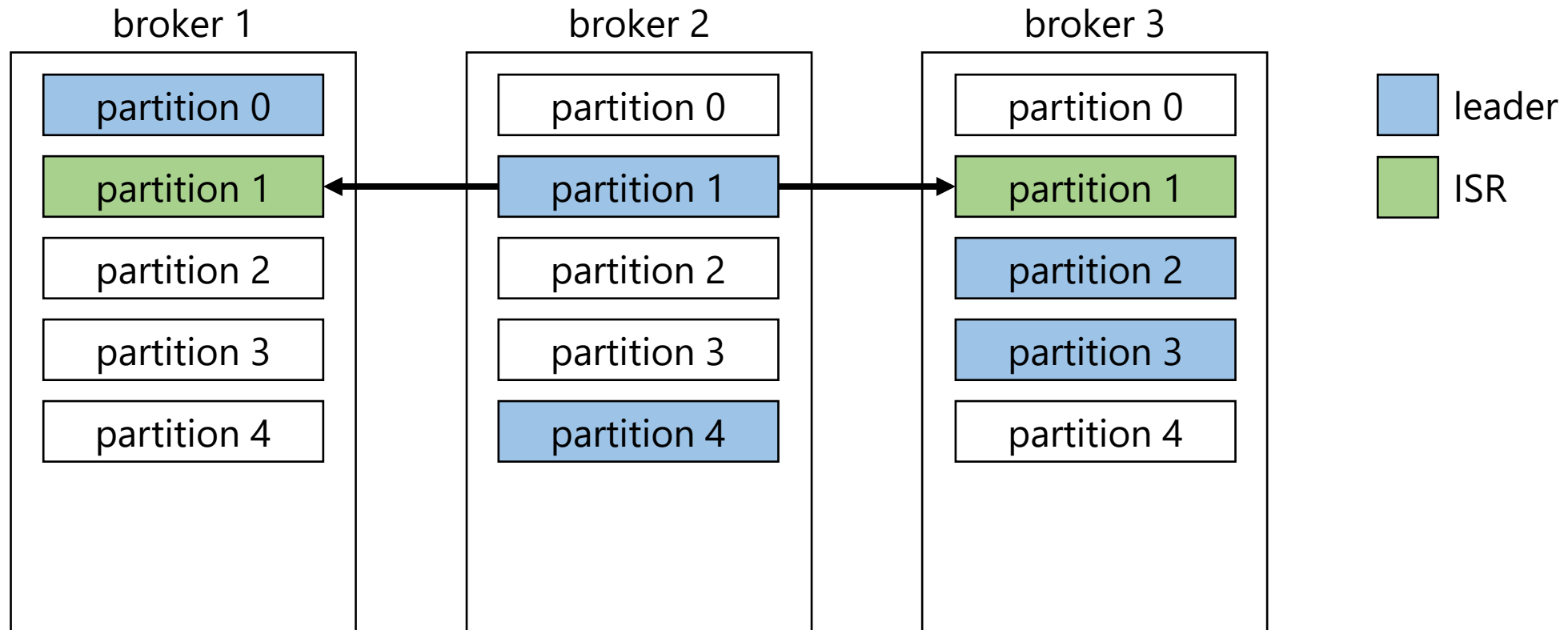
Архитектура Kafka Broker

Репликация с *лидера* на другие брокеры (*фолловеры*)



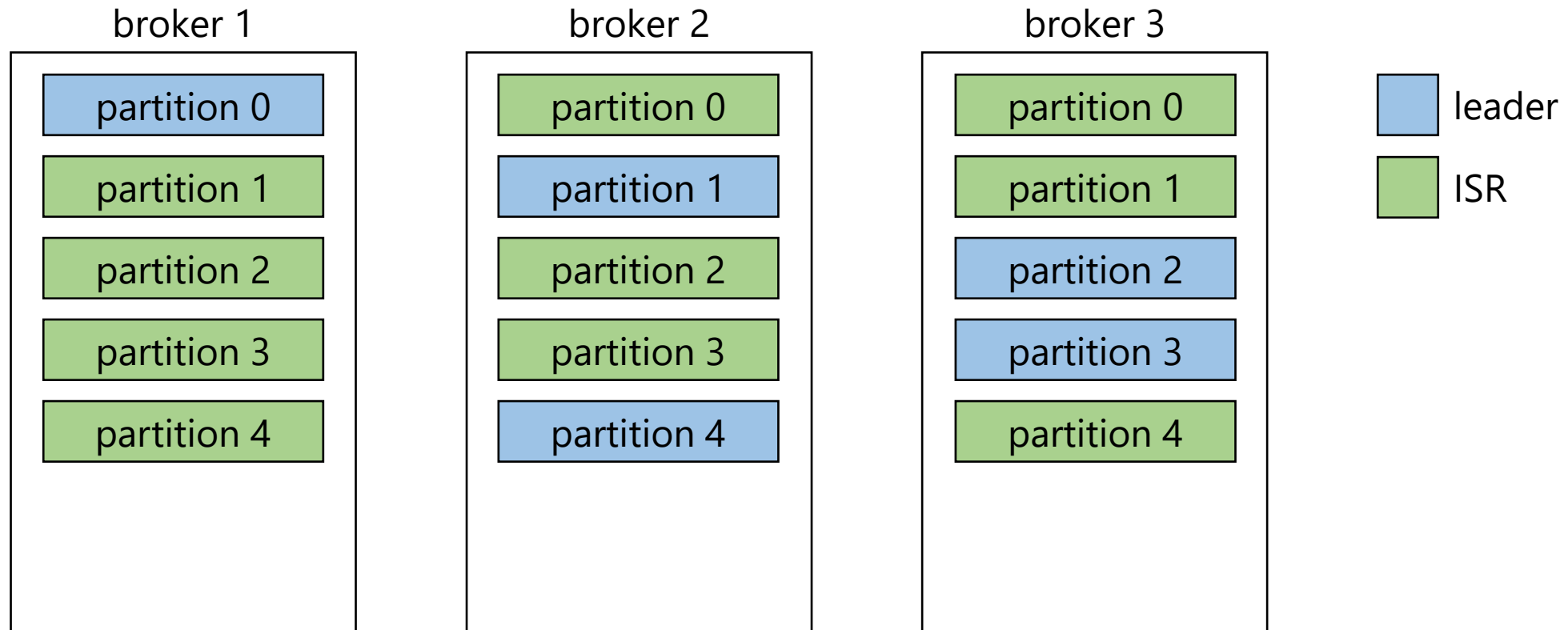
Архитектура Kafka Broker

ISR (in sync replica) – *реплика, синхронизированная с лидером*



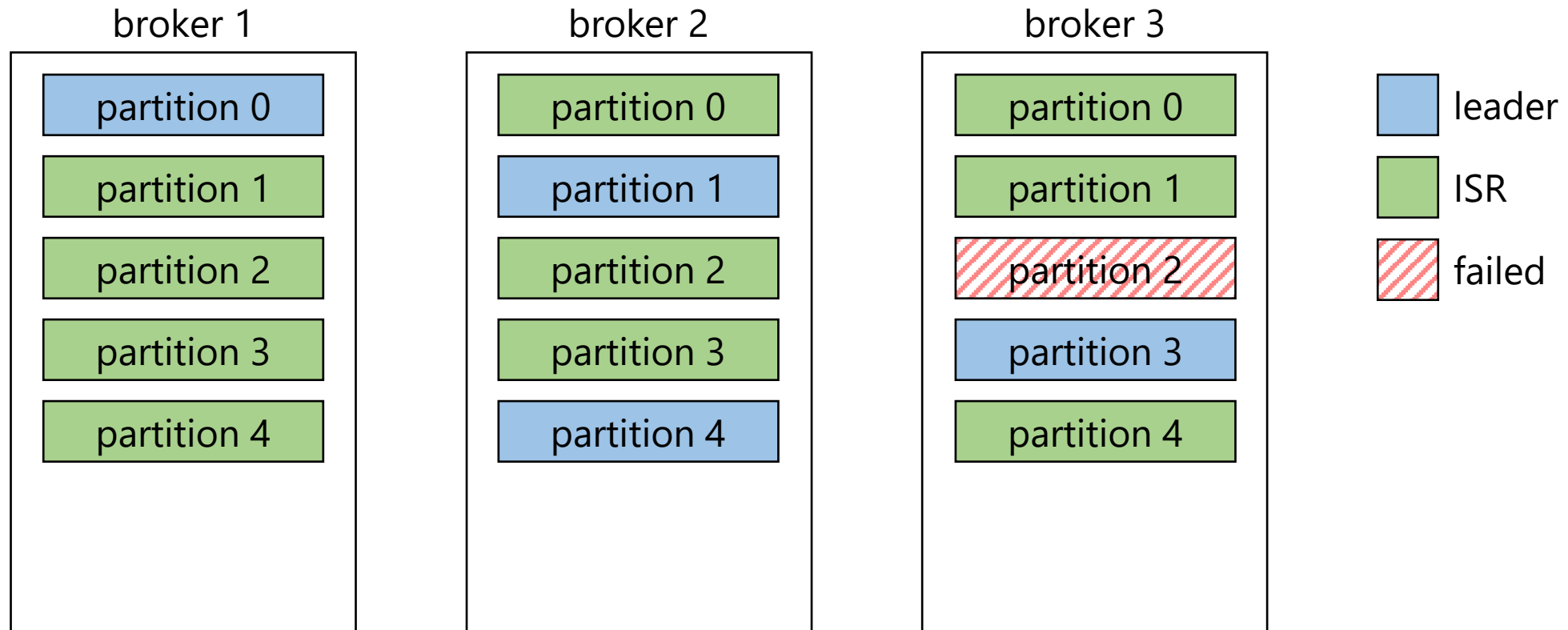
Архитектура Kafka Broker

Все реплики синхронизированы



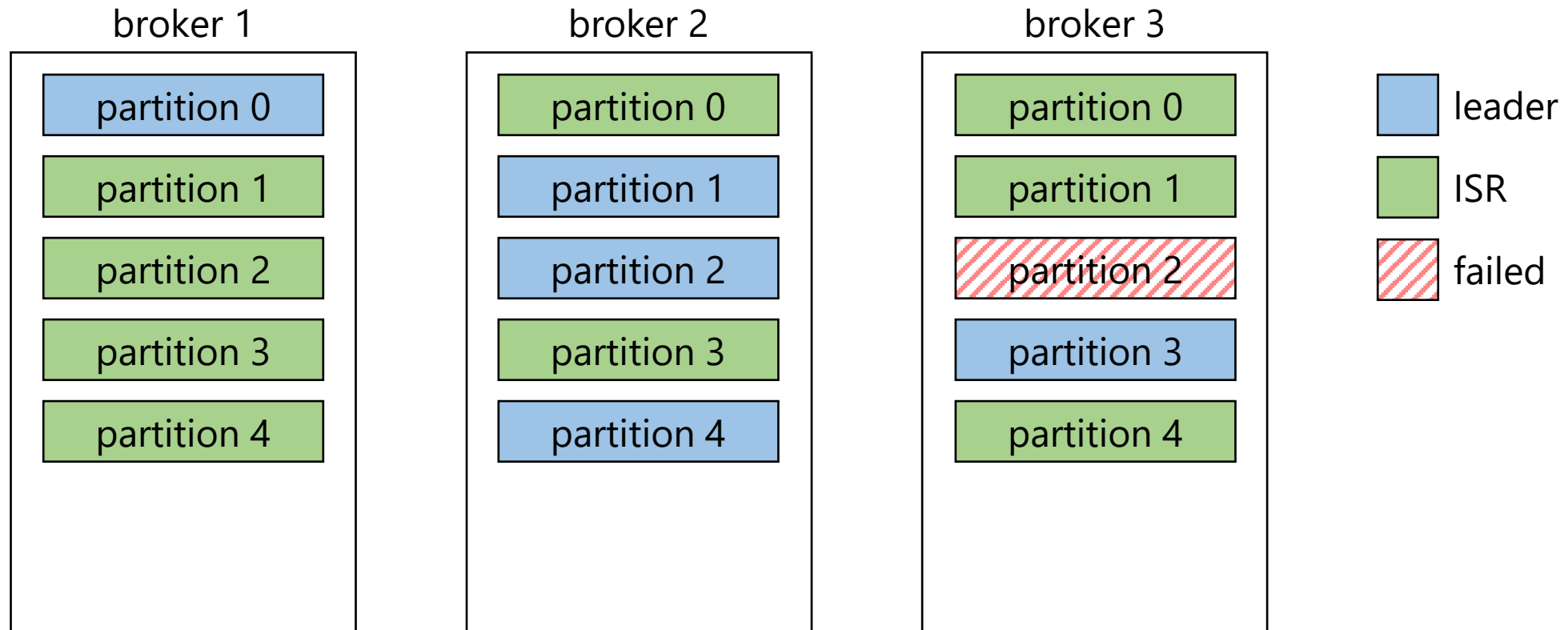
Архитектура Kafka Broker

Недоступность *лидера* у partition 2



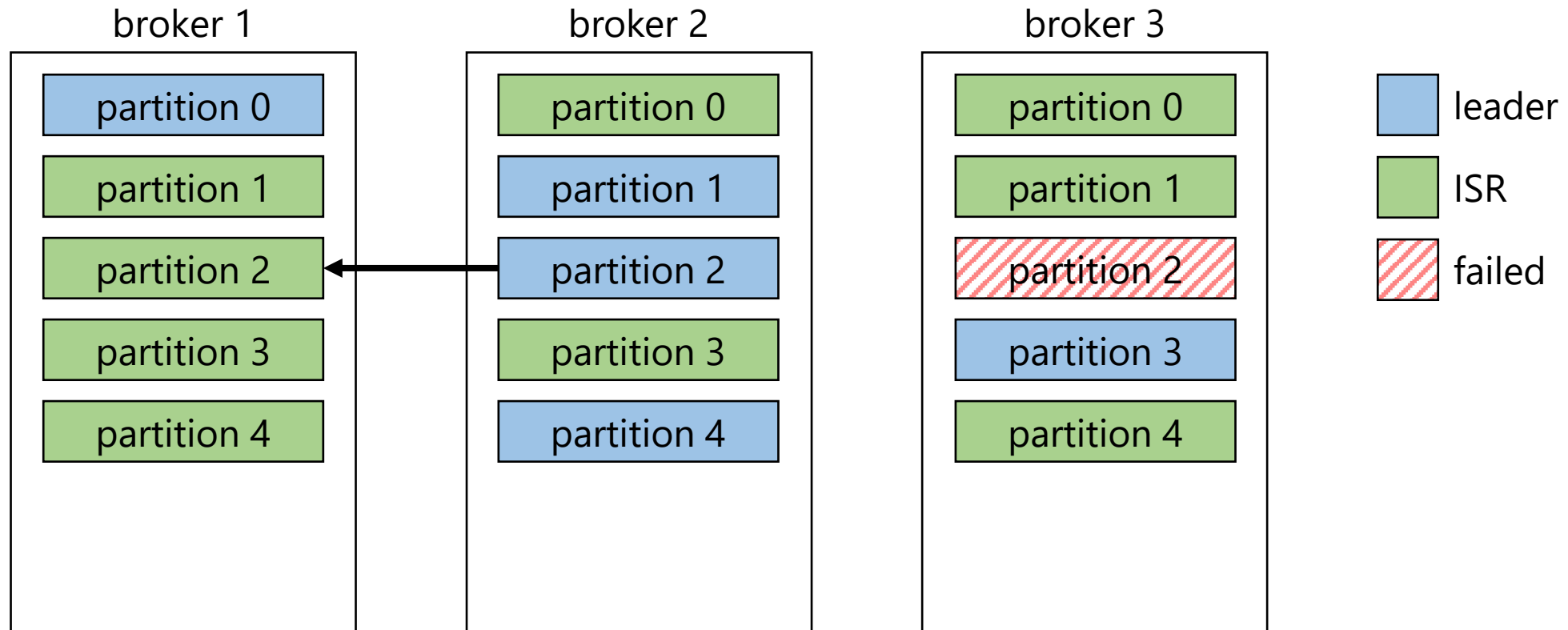
Архитектура Kafka Broker

Выбор нового *лидера* в случае недоступности

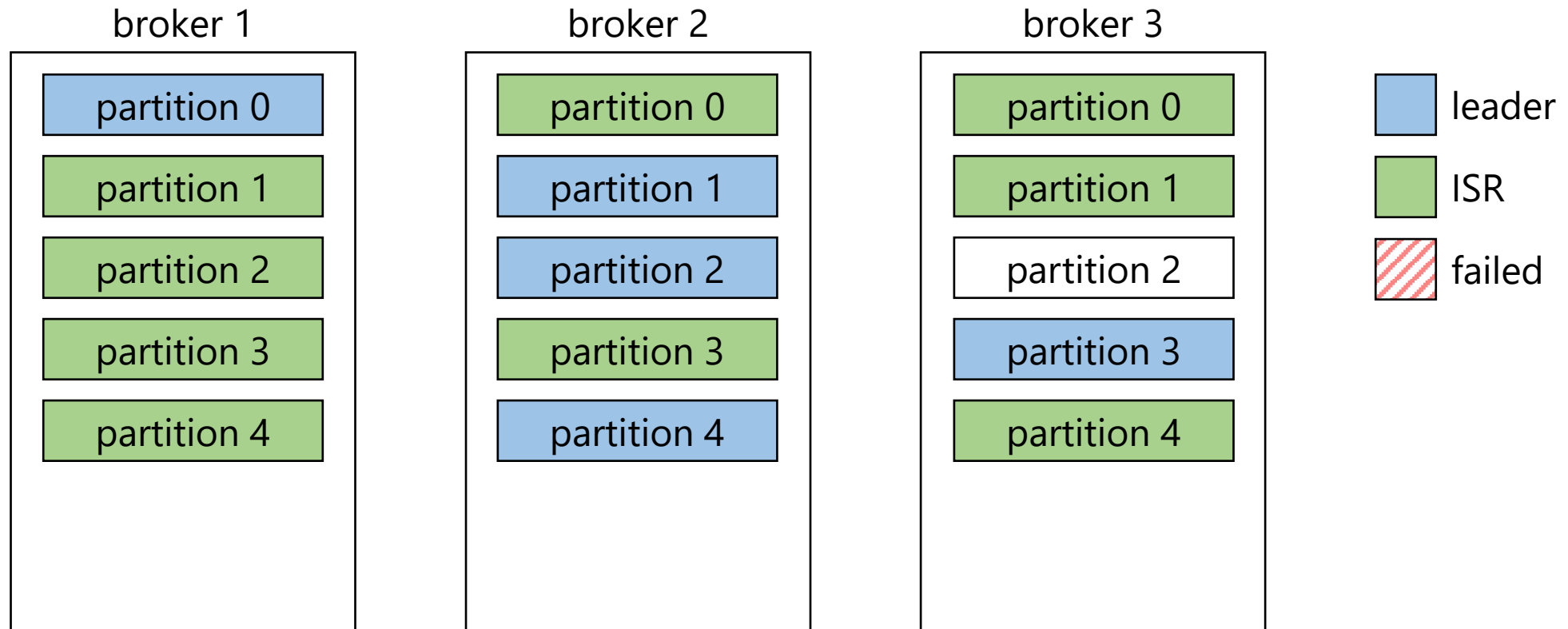


Архитектура Kafka Broker

Репликация с нового *лидера*

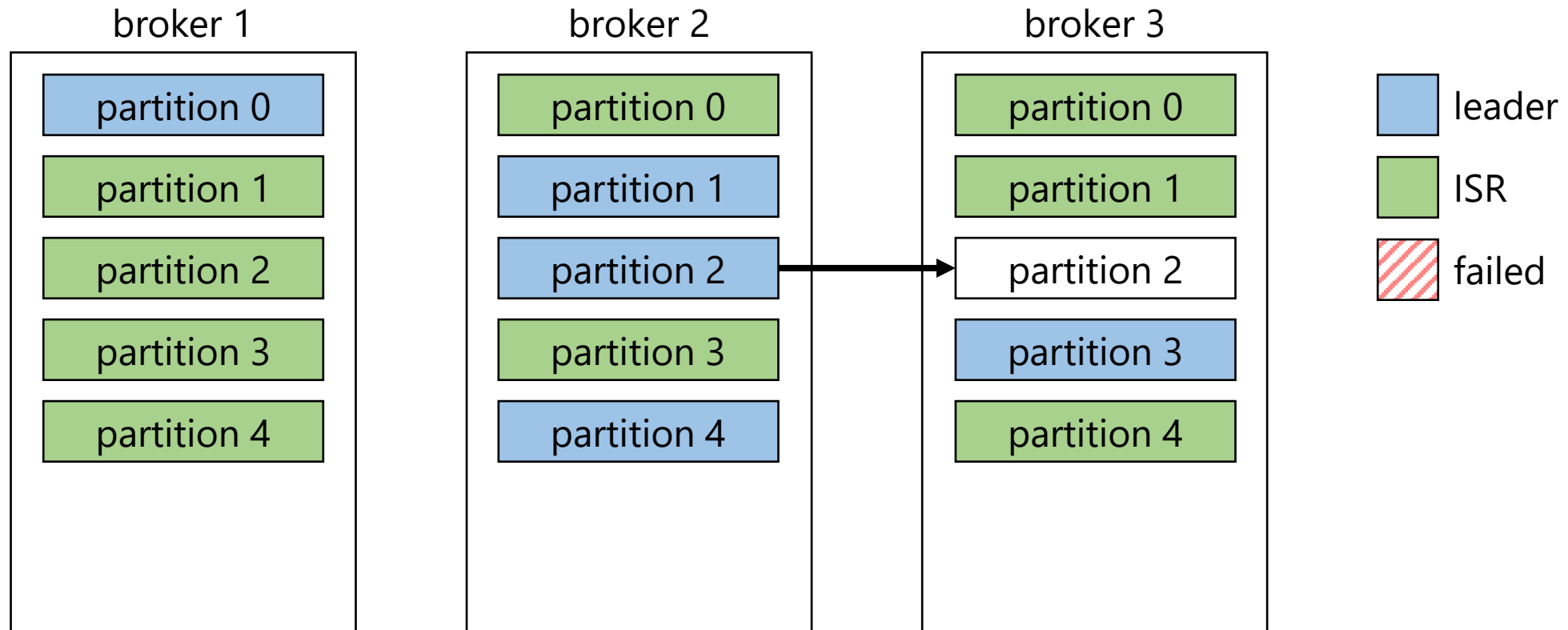


Архитектура Kafka Broker

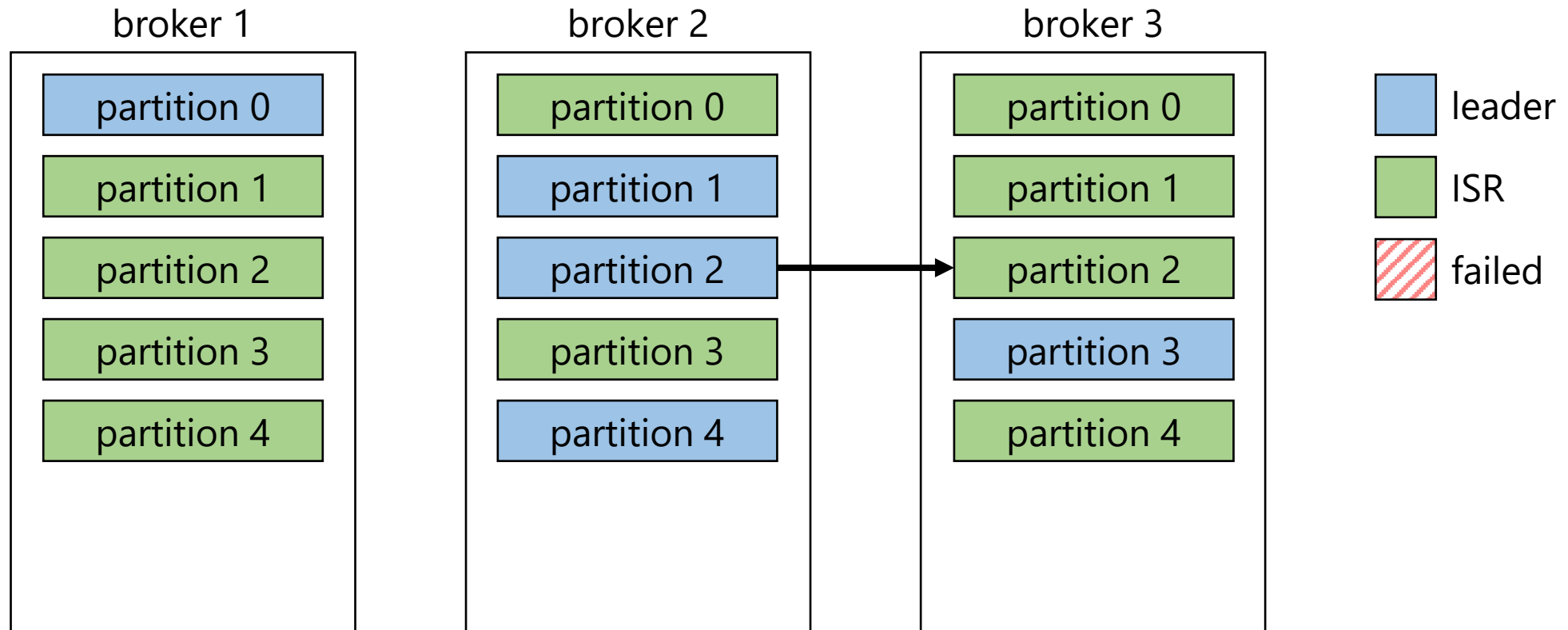


Архитектура Kafka Broker

Синхронизация реплики с *лидером* после восстановления

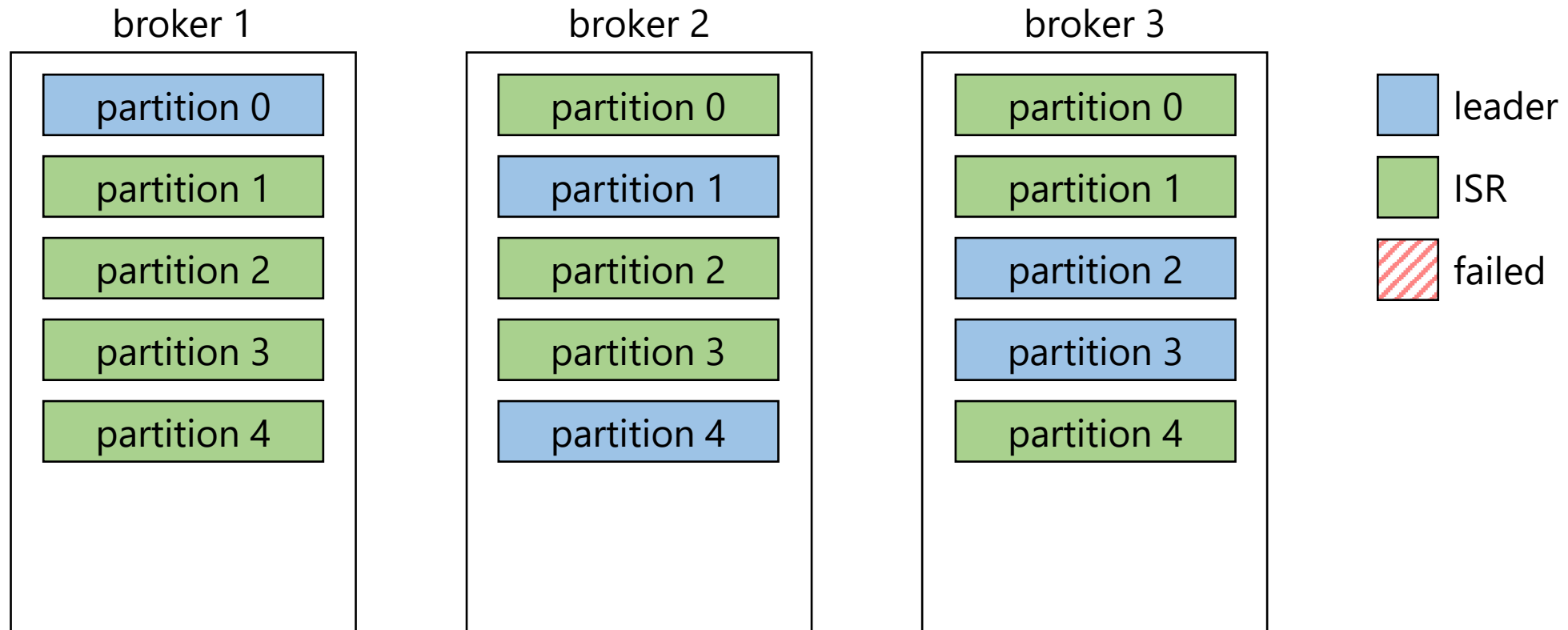


Архитектура Kafka Broker



Архитектура Kafka Broker

Перебалансировка *лидеров*

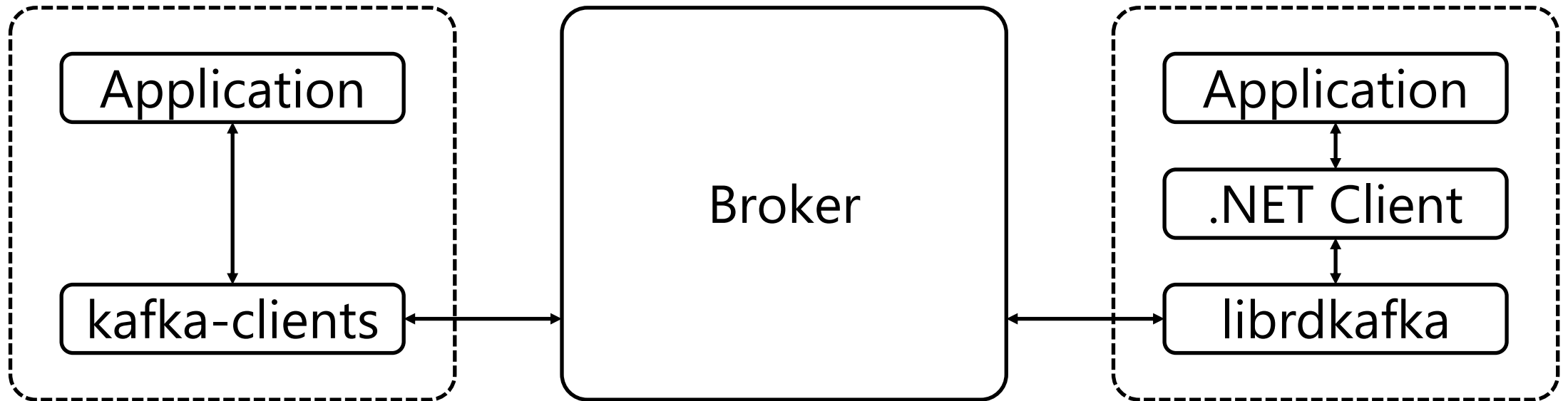


Архитектура Kafka Broker

Выводы

- У каждой партиции свой Лидер
- Сообщения пишутся в Лидера
- Данные реплицируются между брокерами
- Автоматический фейловер лидерства

Kafka Client: Java vs все остальные



Kafka Producer

Kafka Producer

```
new KafkaProducer<TKey, TValue>(
    producerProperties,
    keySerializer,
    valueSerializer
)
```

Kafka Producer

```
new KafkaProducer<TKey, TValue>(
    producerProperties,
    keySerializer,
    valueSerializer
)
```

Kafka Producer

```
# Producer properties
```

```
bootstrap.servers =  
    host1:port1,host2:port2, ...
```

```
...
```

Kafka Producer

```
new KafkaProducer<TKey, TValue>(
    producerProperties,
    keySerializer,
    valueSerializer
)
```

Kafka Producer

```
new KafkaProducer<TKey, TValue>(
    producerProperties,
    keySerializer,
    valueSerializer
)
```


Kafka Producer

```
new KafkaProducer<Long, byte[]>(
    producerProperties,
    new LongSerializer(),
    new ByteArraySerializer()
)
```

Kafka Producer

```
new ProducerRecord<TKey, TValue>(
    topic,
    partition,
    System.currentTimeMillis(),
    key,
    value,
    headers
)
```

Kafka Producer

```
new ProducerRecord<TKey, TValue>(
    topic,
    partition,
    System.currentTimeMillis(),
    key,
    value,
    headers
)
```

Kafka Producer

```
new ProducerRecord<TKey, TValue>(
    topic,
    partition,
    System.currentTimeMillis(),
    key,
    value,
    headers
)
```

Kafka Producer

```
new ProducerRecord<TKey, TValue>(
    topic,
    partition,
    System.currentTimeMillis(),
    key,
    value,
    headers
)
```

Kafka Producer

```
new ProducerRecord<TKey, TValue>(
    topic,
    partition,
    System.currentTimeMillis(),
    key,
    value,
    headers
)
```

Kafka Producer

```
new ProducerRecord<TKey, TValue>(
    topic,
    partition,
    System.currentTimeMillis(),
    key,
    value,
    headers
)
```

Kafka Producer

```
new ProducerRecord<TKey, TValue>(
    topic,
    partition,
    System.currentTimeMillis(),
    key,
    value,
    headers
)
```


Kafka Producer

```
# Producer properties
```

```
partitioner.class =  
    null |  
    o.a.k.clients.producer.RoundRobinPartitioner |  
    o.a.k.clients.producer.UniformStickyPartitioner |  
    ...
```

Kafka Producer

```
// ...  
producer.send(  
    record,  
    (metadata, exception) → {/* callback */}  
);
```

Kafka Producer

```
// ...  
producer.send(  
    record,  
    (metadata, exception) → { /* callback */ }  
);
```

Kafka Producer

```
// ...  
producer.send(  
    record,  
    (metadata, exception) → {/* callback */}  
);
```

Kafka Producer

Kafka Producer

Producer properties

acks = 0 | 1 | all

Kafka Producer

replica 0

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

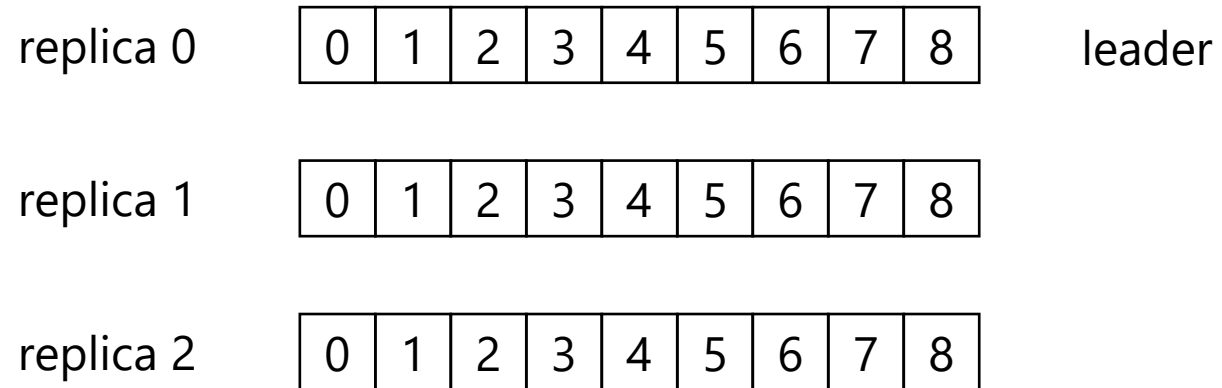
replica 1

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

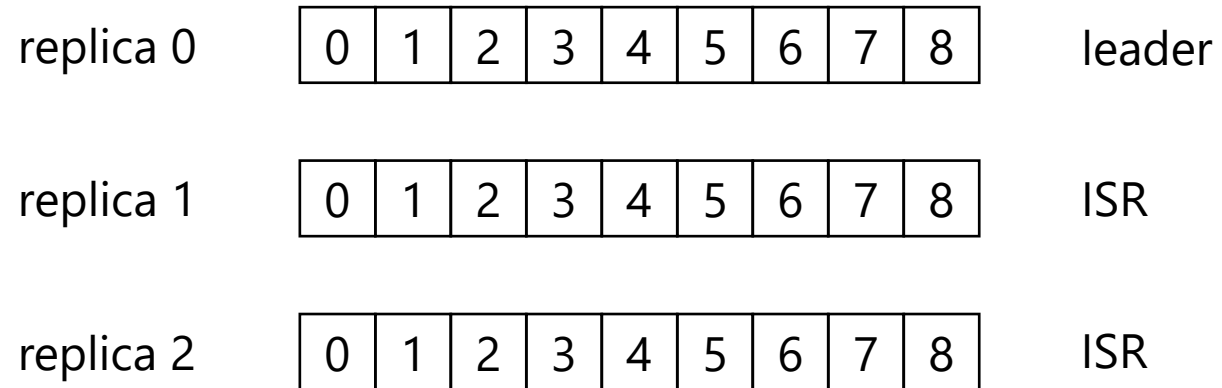
replica 2

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

Kafka Producer



Kafka Producer



Kafka Producer

replica 0

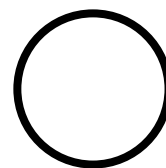
0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

replica 1

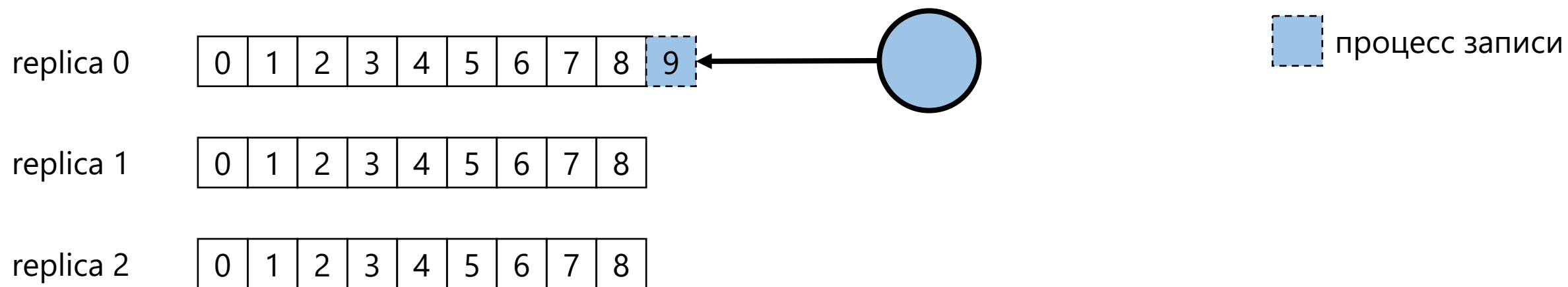
0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

replica 2

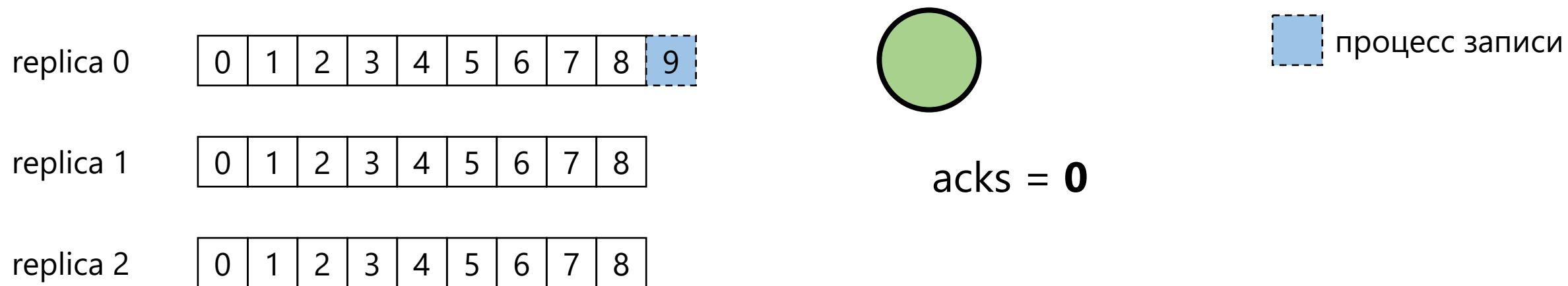
0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---



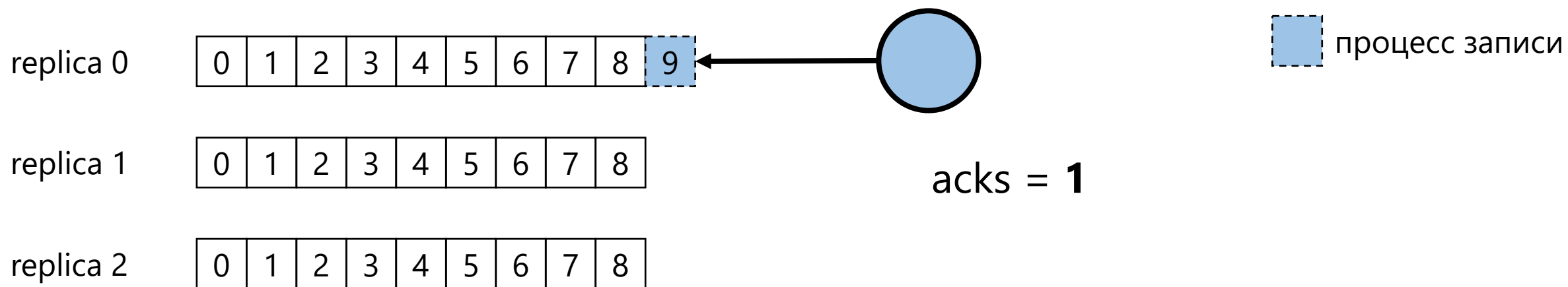
Kafka Producer



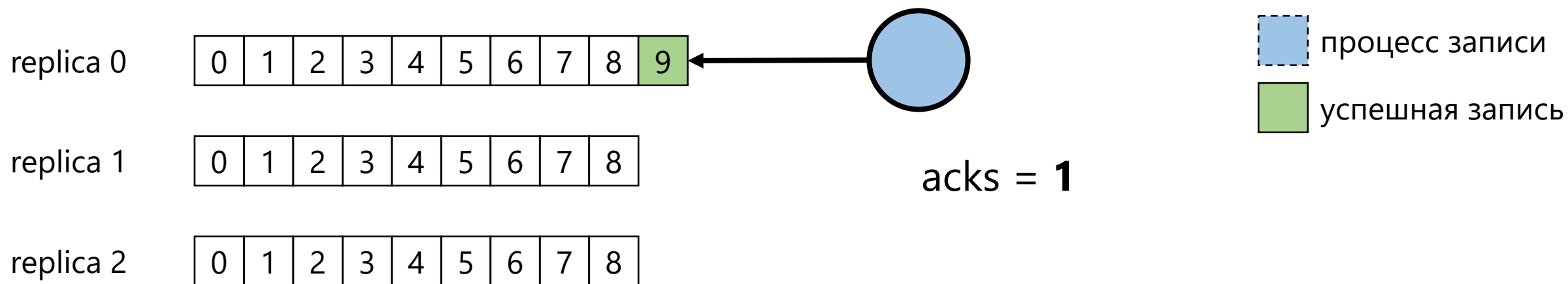
Kafka Producer



Kafka Producer



Kafka Producer



Kafka Producer

replica 0

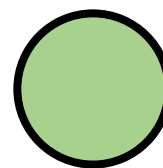
0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

replica 1

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

replica 2

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---



acks = **1**

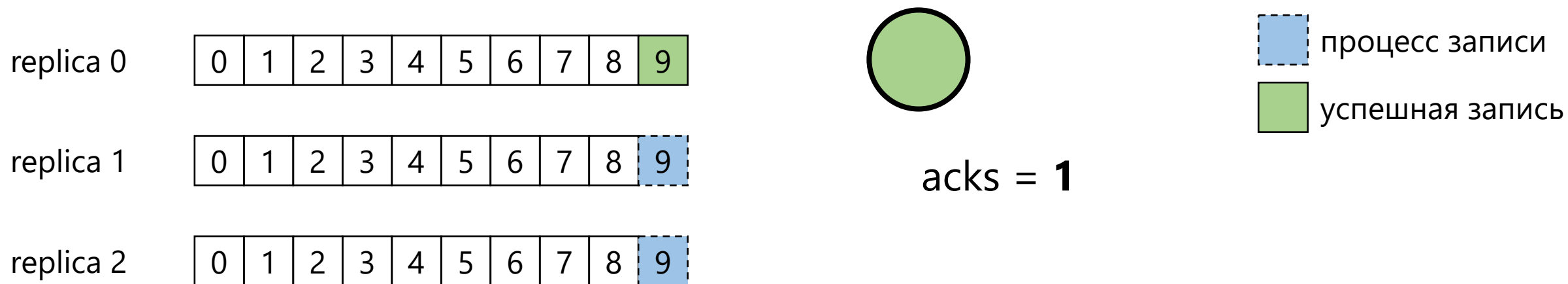


процесс записи

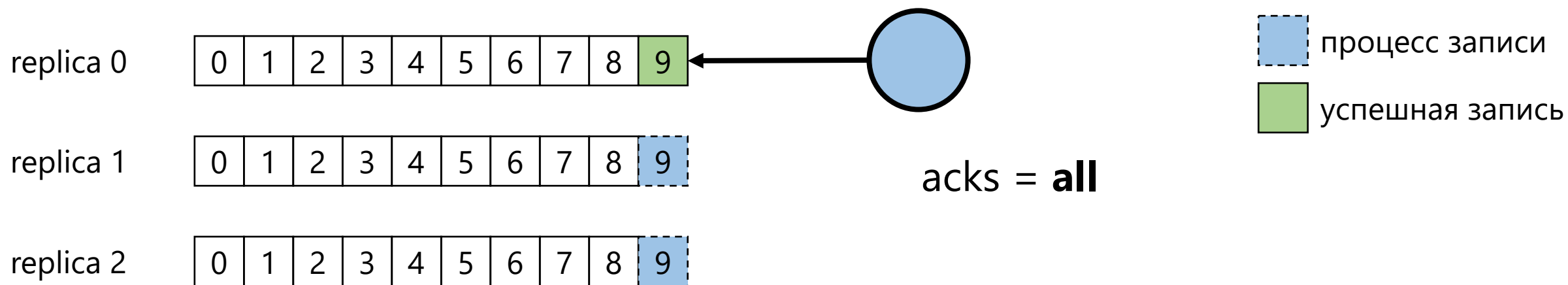


успешная запись

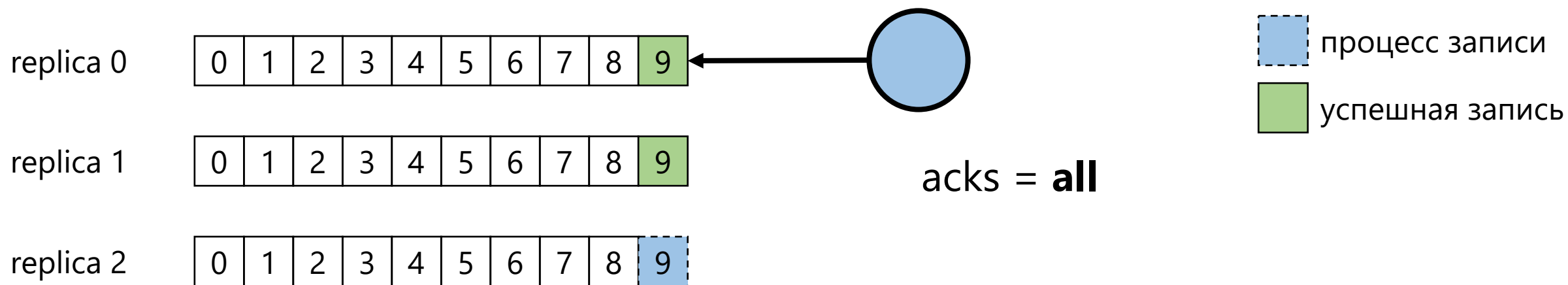
Kafka Producer



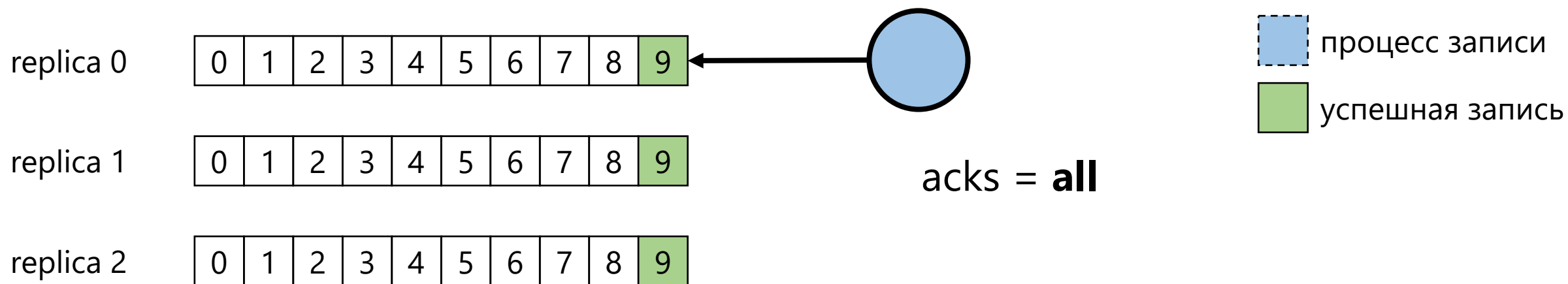
Kafka Producer



Kafka Producer



Kafka Producer



Kafka Producer

replica 0

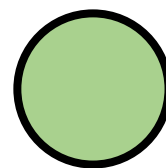
0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

replica 1

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

replica 2

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---



acks = **all**



процесс записи



успешная запись

Kafka Producer

replica 0

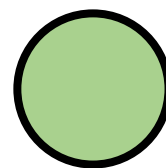
0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

replica 1

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

replica 2

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---



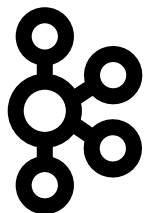
acks = **all**



процесс записи

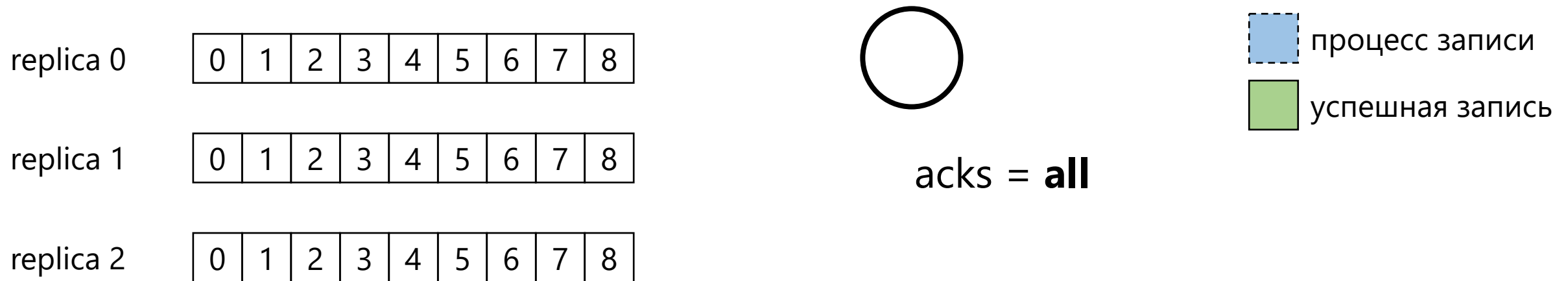


успешная запись



Kafka Producer

Topic config: min.insync.replicas = 2



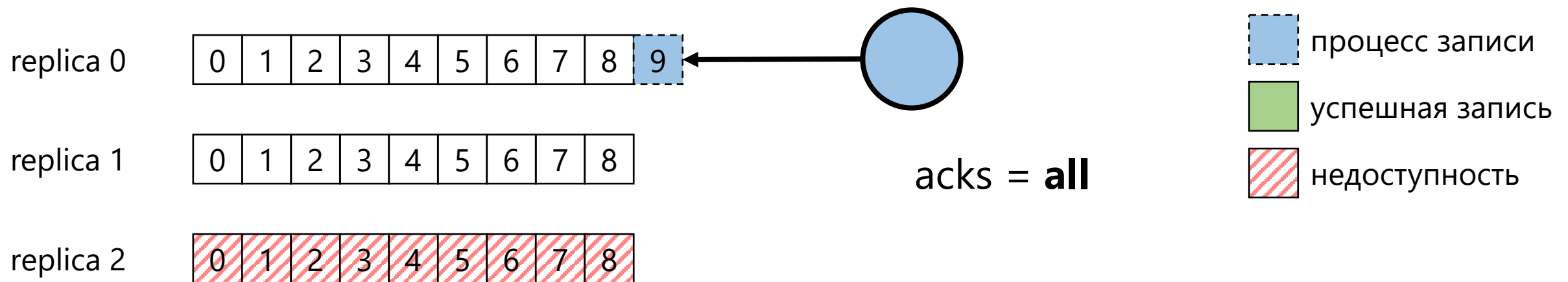
Kafka Producer

Topic config: min.insync.replicas = 2



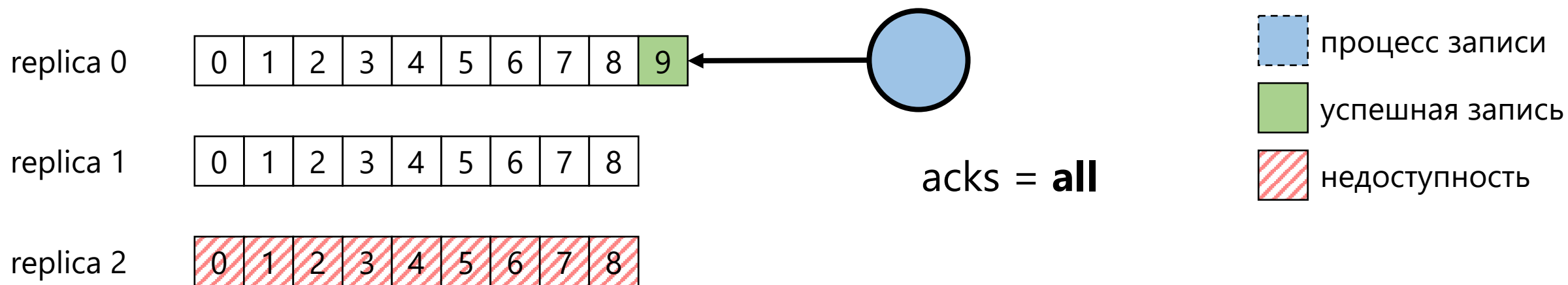
Kafka Producer

Topic config: min.insync.replicas = 2



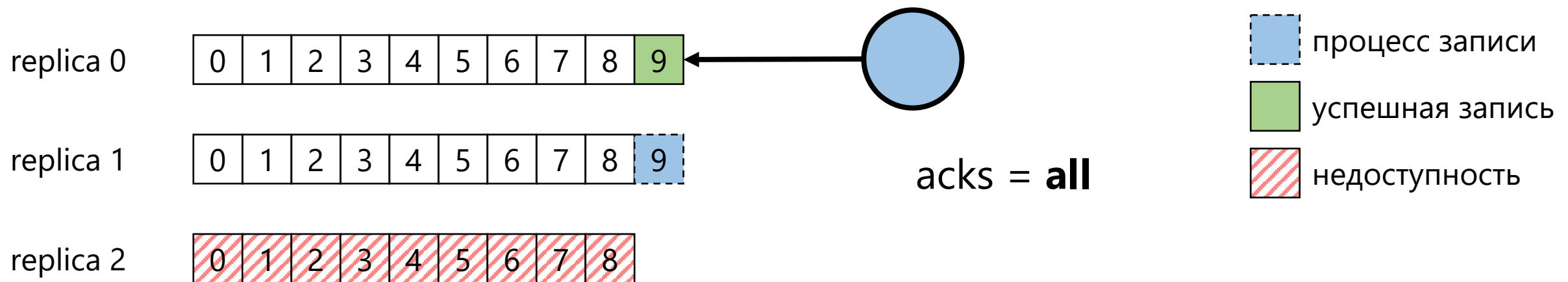
Kafka Producer

Topic config: min.insync.replicas = 2



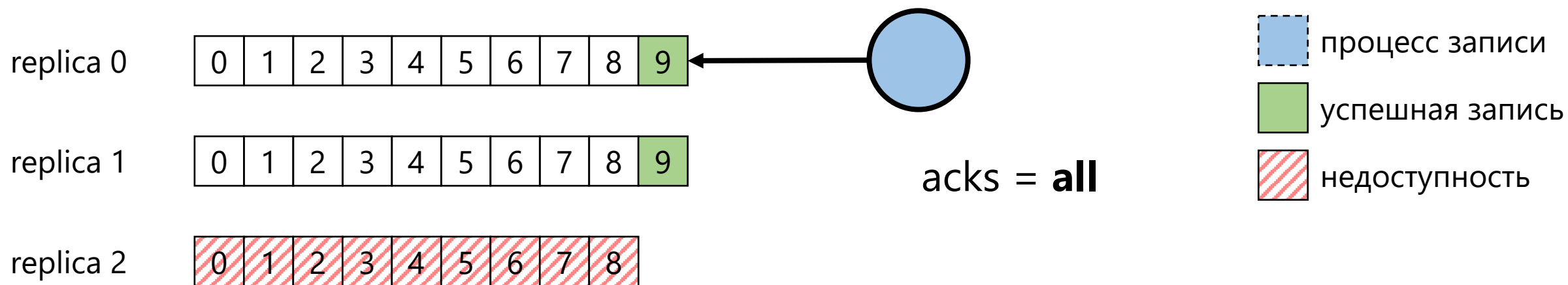
Kafka Producer

Topic config: min.insync.replicas = 2



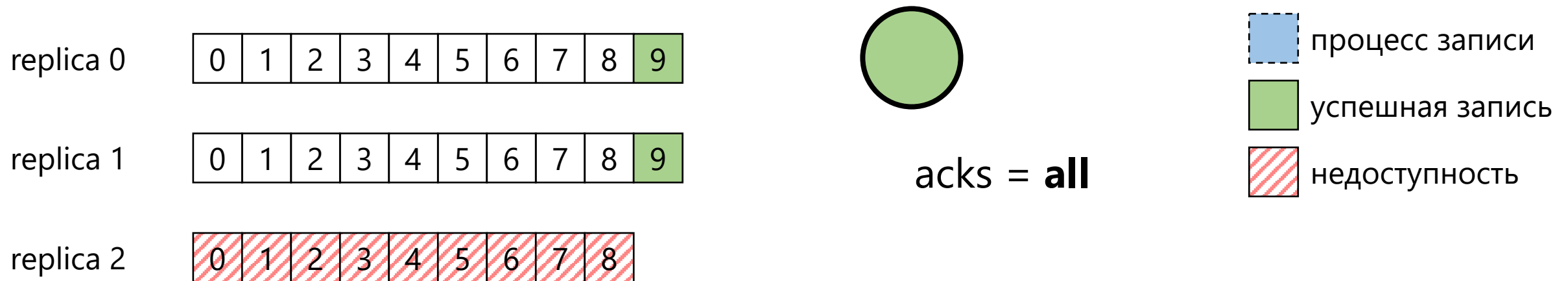
Kafka Producer

Topic config: min.insync.replicas = 2



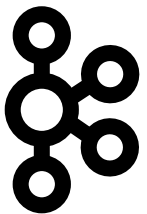
Kafka Producer

Topic config: min.insync.replicas = 2



Kafka Producer

Производительность



Kafka Producer

Производительность

- Низкая задержка
- Высокая пропускная способность

Kafka Producer

Производительность

— Низкая задержка

— Высокая пропускная способность



Broker

Kafka Producer

Производительность

— Низкая задержка

— Высокая пропускная способность

msg

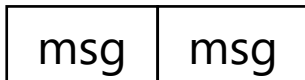
Broker

Kafka Producer

Производительность

— Низкая задержка

— Высокая пропускная способность

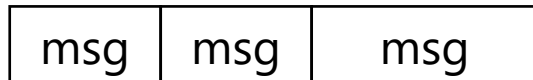


Kafka Producer

Производительность

— Низкая задержка

— Высокая пропускная способность



Kafka Producer

Производительность

— Низкая задержка

— Высокая пропускная способность

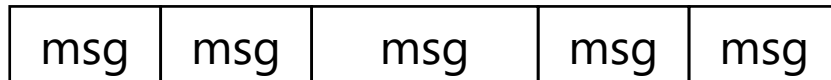


Kafka Producer

Производительность

— Низкая задержка

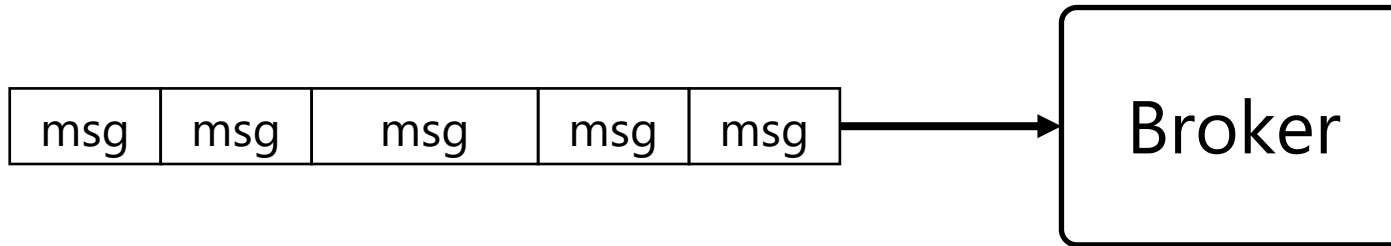
— Высокая пропускная способность



Kafka Producer

Производительность

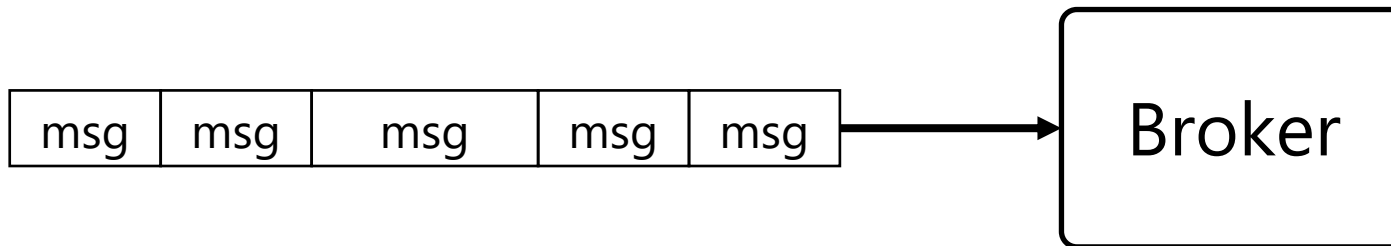
- Низкая задержка
- Высокая пропускная способность



Kafka Producer

Производительность

- Низкая задержка
- Высокая пропускная способность

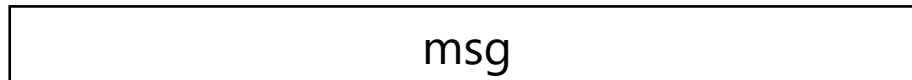


`batch.size = 100000`

Kafka Producer

Производительность

- Низкая задержка
- Высокая пропускная способность

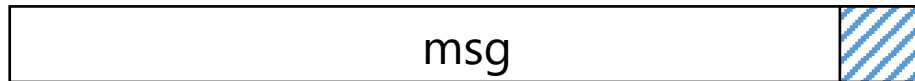


`batch.size = 100000`

Kafka Producer

Производительность

- Низкая задержка
- Высокая пропускная способность

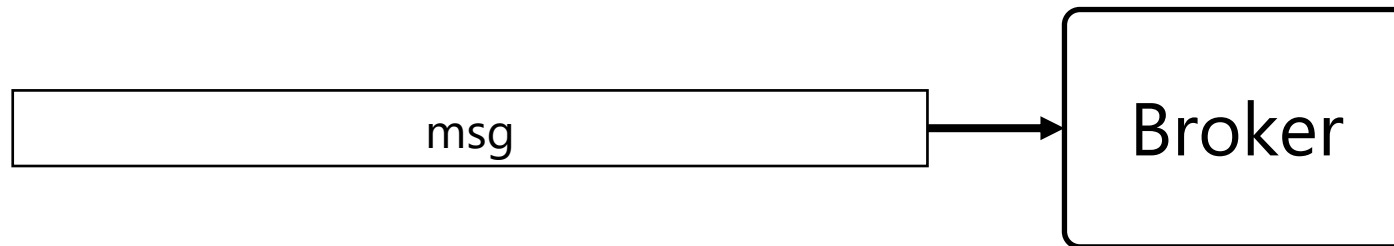


```
batch.size = 100000
```


Kafka Producer

Производительность

- Низкая задержка
- Высокая пропускная способность



`batch.size = 100000`

Kafka Producer

Производительность

- Низкая задержка
- Высокая пропускная способность

```
batch.size = 100000  
linger.ms = 5
```



Broker

Kafka Producer

Производительность

- Низкая задержка
- Высокая пропускная способность

msg

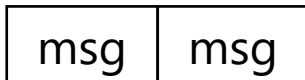
```
batch.size = 100000  
linger.ms = 5
```

Broker

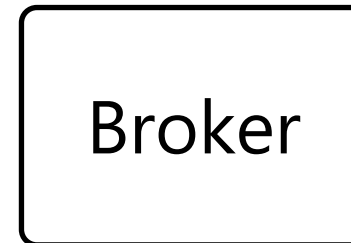
Kafka Producer

Производительность

- Низкая задержка
- Высокая пропускная способность



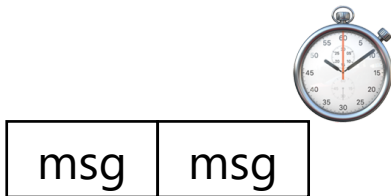
```
batch.size = 100000  
linger.ms = 5
```



Kafka Producer

Производительность

- Низкая задержка
- Высокая пропускная способность



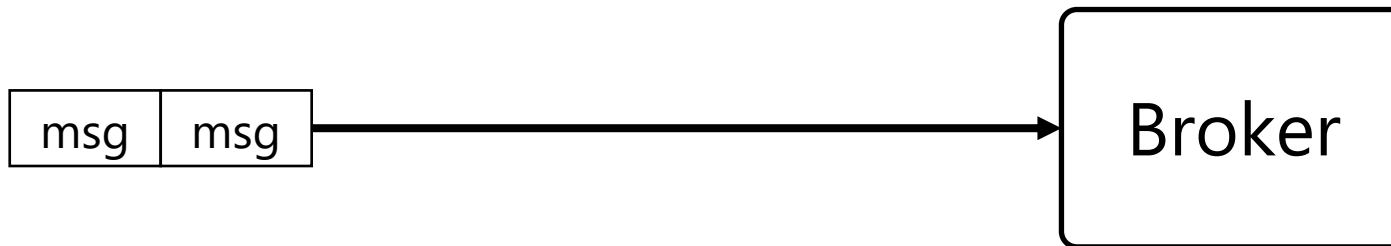
```
batch.size = 100000  
linger.ms = 5
```



Kafka Producer

Производительность

- Низкая задержка
- Высокая пропускная способность

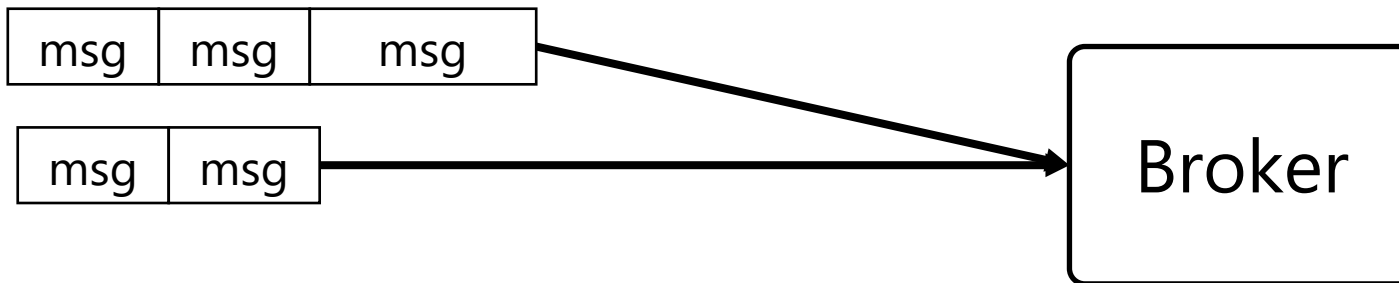


```
batch.size = 100000  
linger.ms = 5
```

Kafka Producer

Производительность

- Низкая задержка
- Высокая пропускная способность



```
batch.size = 100000
```

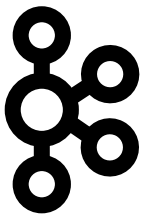
```
linger.ms = 5
```

```
max.request.size = 1000000
```

Kafka Producer

Производительность

- Низкая задержка
- Высокая пропускная способность



Kafka Producer

```
# Producer properties
```

```
compression.type =  
    none | gzip | snappy | lz4 | zstd
```

Kafka Producer

```
# Producer properties
```

```
compression.type =  
    none | gzip | snappy | lz4 | zstd
```

Kafka Producer

Выводы

- Producer выбирает партицию для сообщения
- Producer определяет уровень гарантии доставки
- В Producer можно тюнить производительность

Kafka Consumer

Kafka Consumer

```
new KafkaConsumer<TKey, TValue>(
    consumerProperties,
    keyDeserializer,
    valueDeserializer
)
```

Kafka Consumer

```
new KafkaConsumer<TKey, TValue>(
    consumerProperties,
    keyDeserializer,
    valueDeserializer
)
```

Kafka Consumer

```
# Consumer properties
```

```
bootstrap.servers =  
    host1:port1,host2:port2,...
```

```
...
```

Kafka Consumer

```
new KafkaConsumer<TKey, TValue>(
    consumerProperties,
    keyDeserializer,
    valueDeserializer
)
```


Kafka Consumer

```
new KafkaConsumer<TKey, TValue>(
    consumerProperties,
    keyDeserializer,
    valueDeserializer
)
```

Kafka Consumer

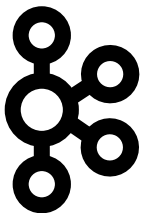
```
// Подписаться на список топиков  
consumer.subscribe(topics);
```

```
// Подписаться на топики по шаблону имени  
consumer.subscribe(pattern);
```

Kafka Consumer

```
// Подписаться на список топиков  
consumer.subscribe(topics);
```

```
// Подписаться на топики по шаблону имени  
consumer.subscribe(pattern);
```



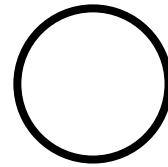
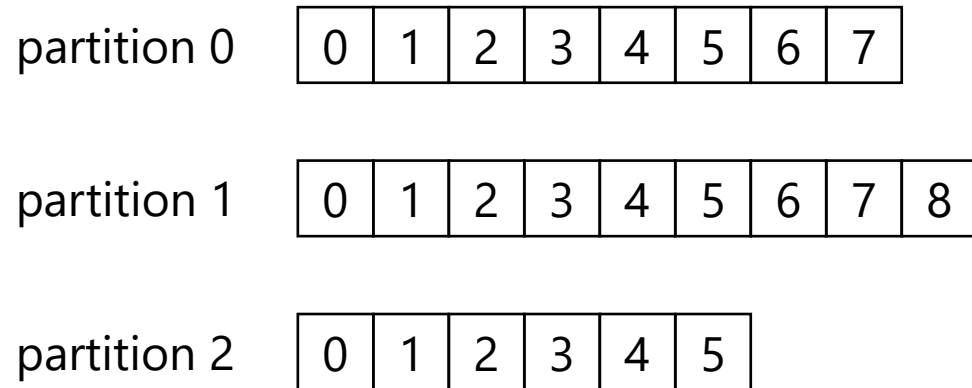
Kafka Consumer

```
ConsumerRecords<UUID, Event> records =  
    consumer.poll(Duration.ofSeconds(5));
```

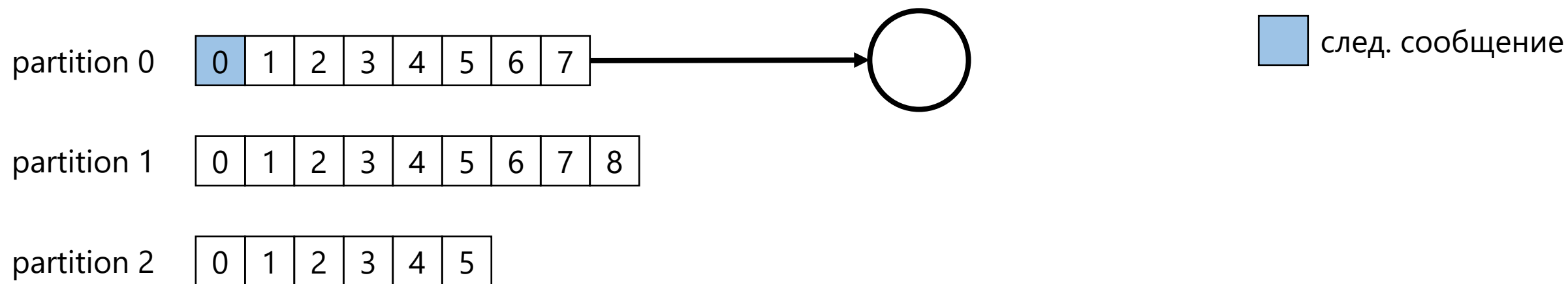
Kafka Consumer

```
ConsumerRecords<UUID, Event> records =  
    consumer.poll(Duration.ofSeconds(5));
```

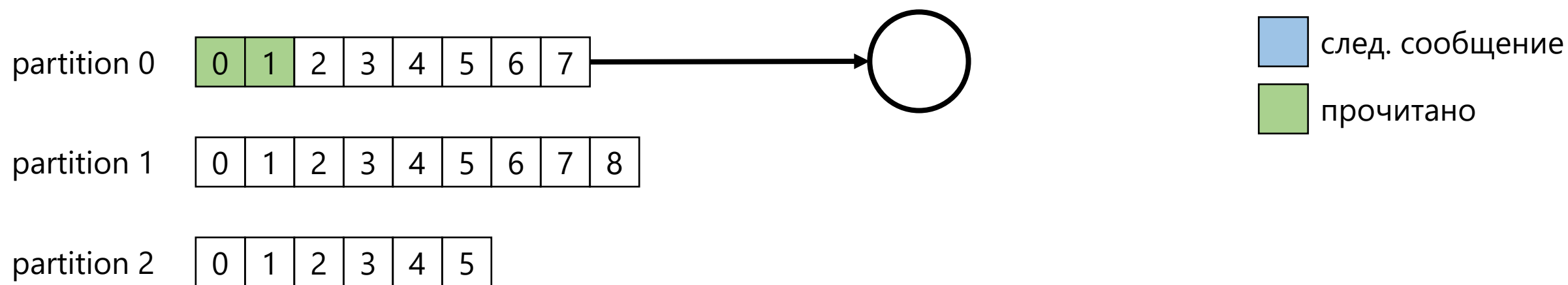
Kafka Consumer



Kafka Consumer

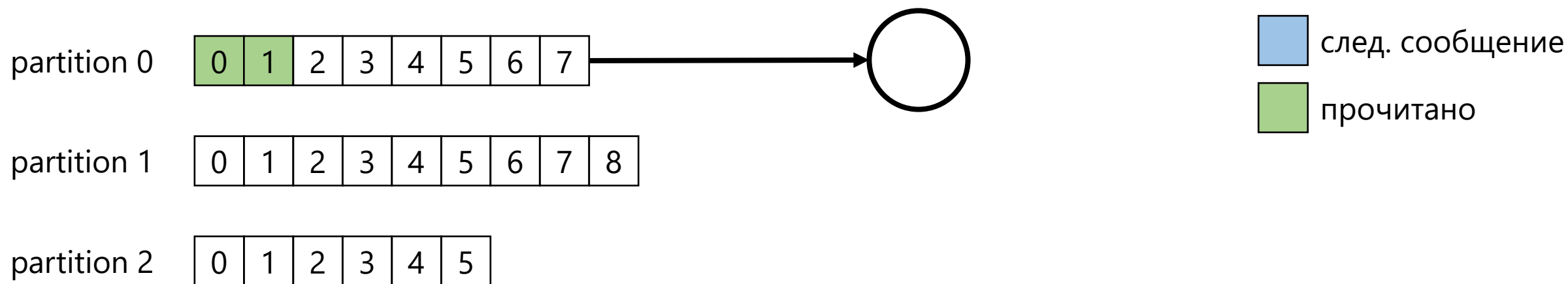


Kafka Consumer

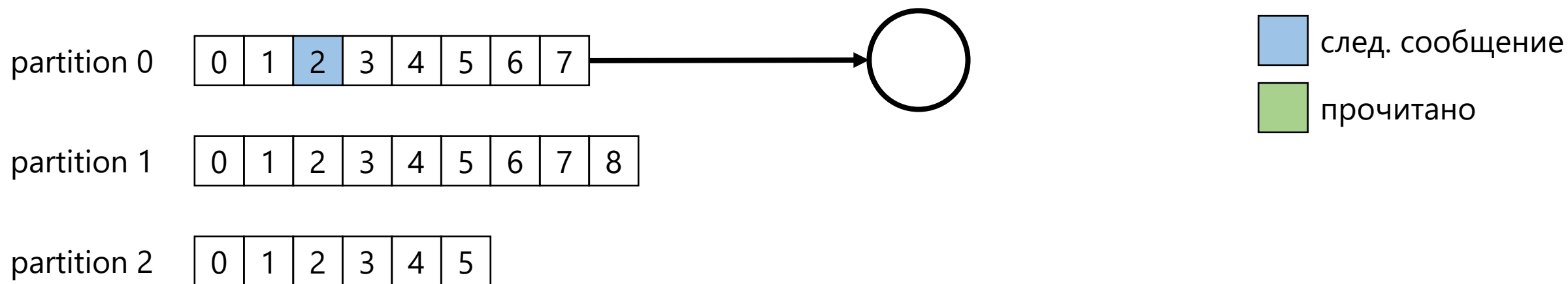


Kafka Consumer

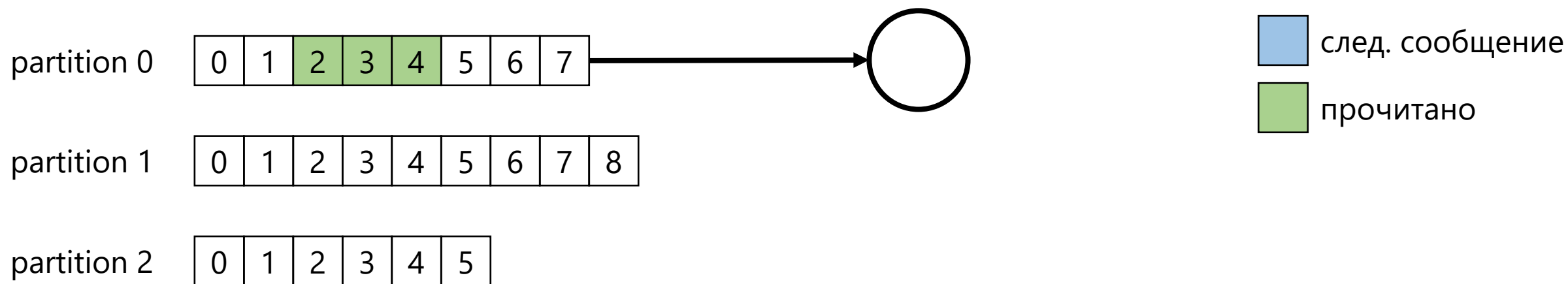
`max.poll.records = 500`



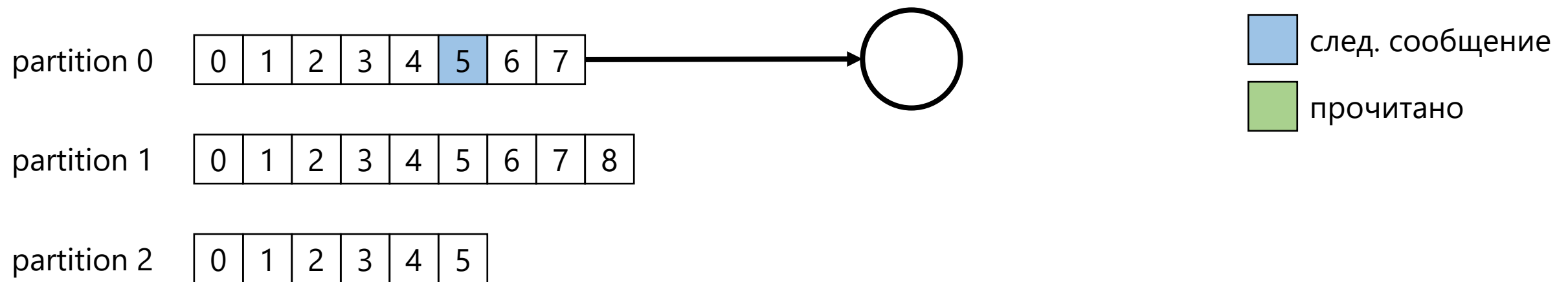
Kafka Consumer



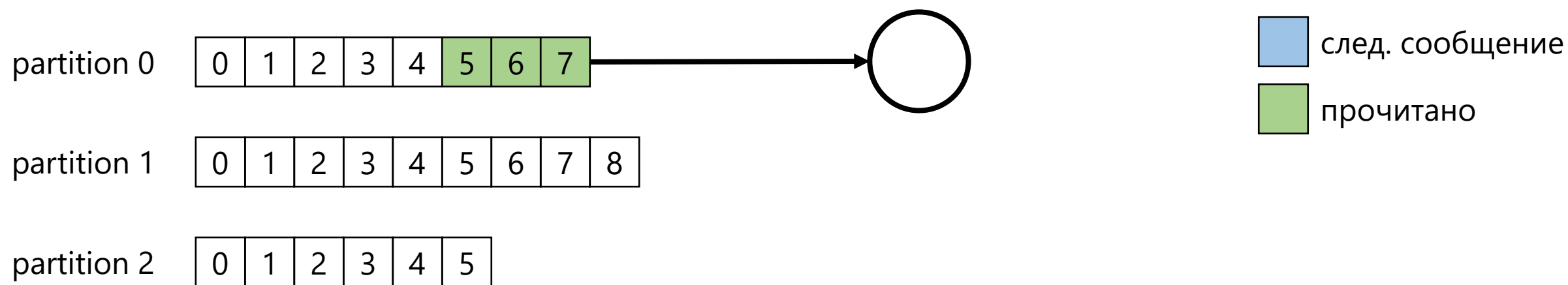
Kafka Consumer



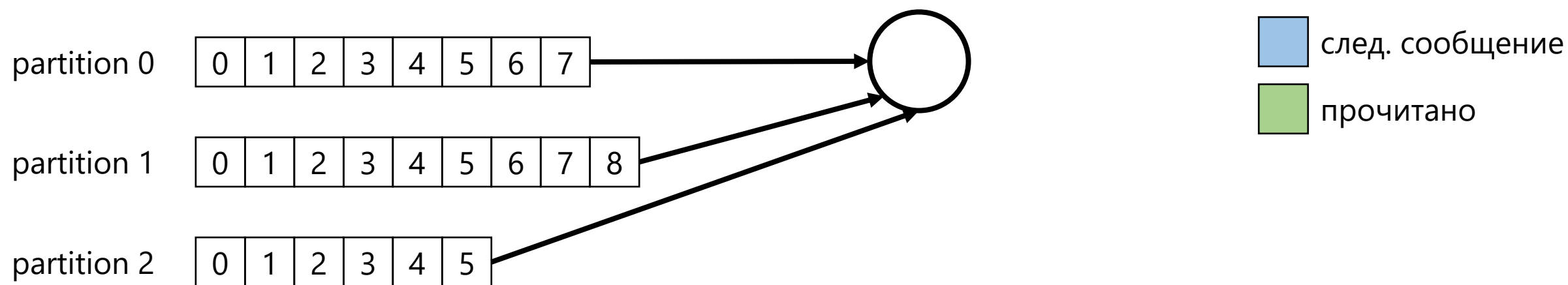
Kafka Consumer



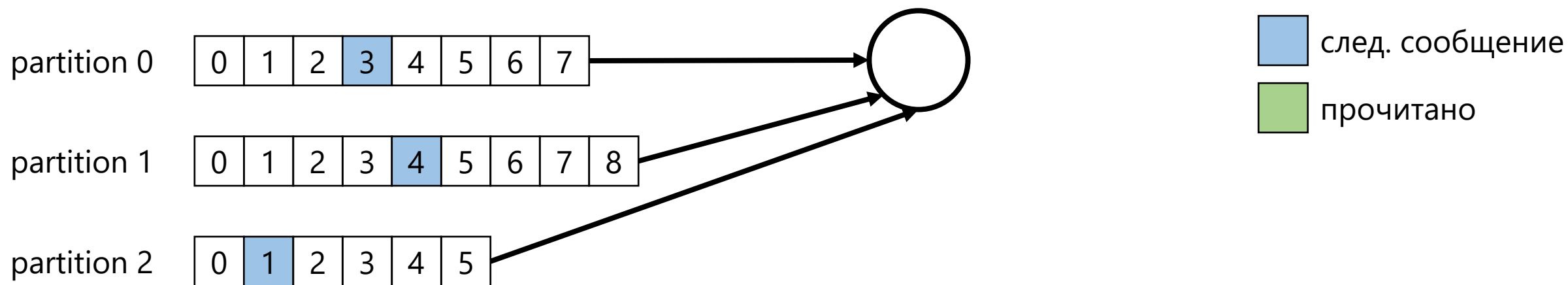
Kafka Consumer



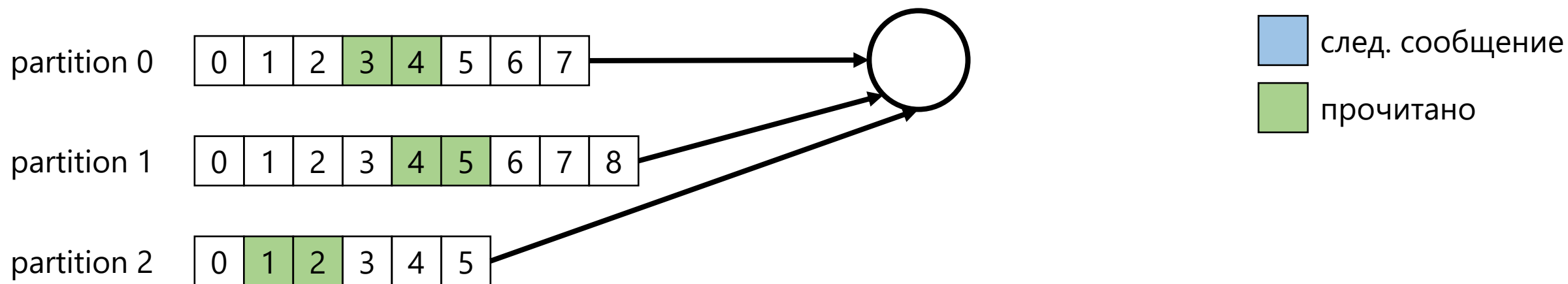
Kafka Consumer



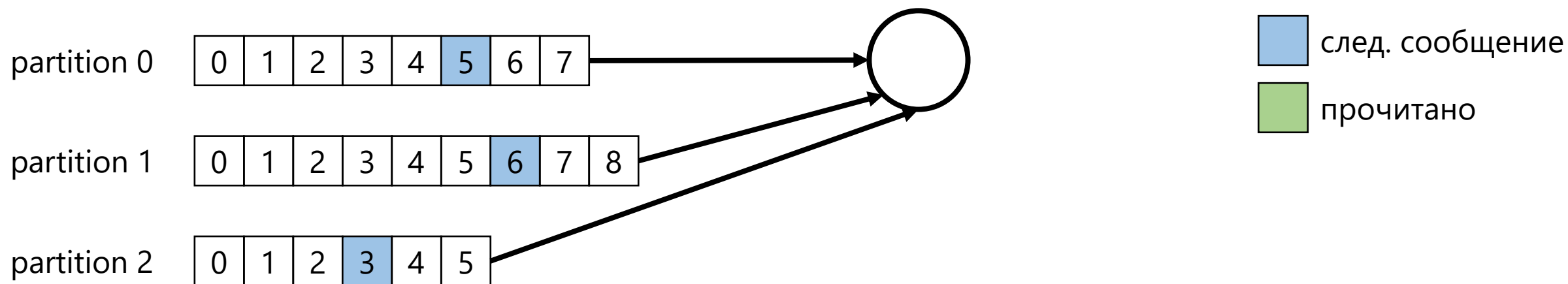
Kafka Consumer



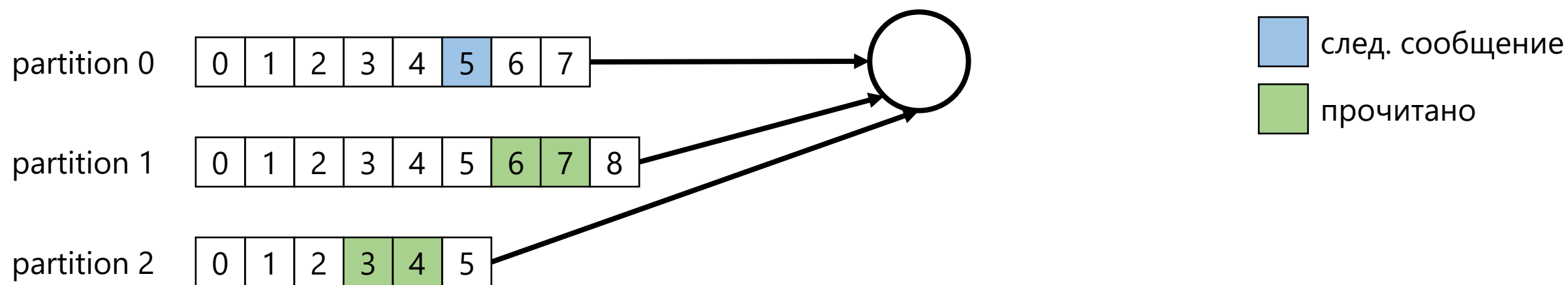
Kafka Consumer



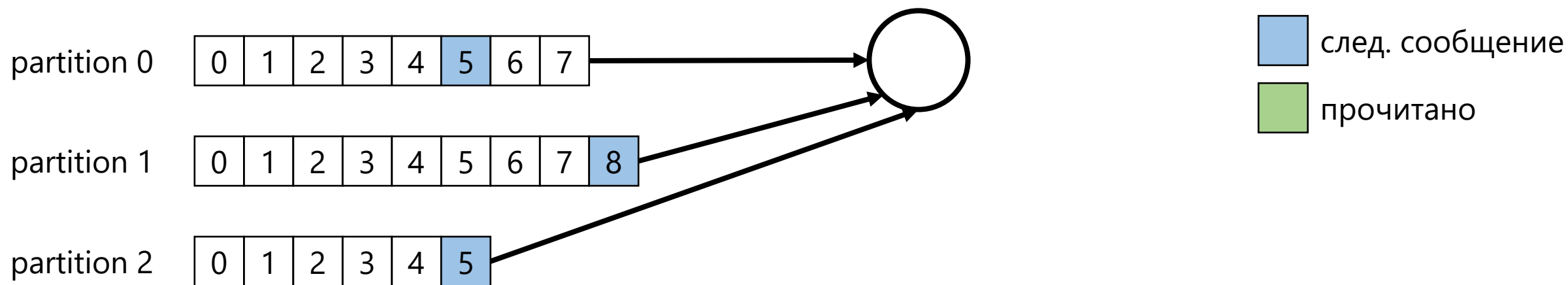
Kafka Consumer



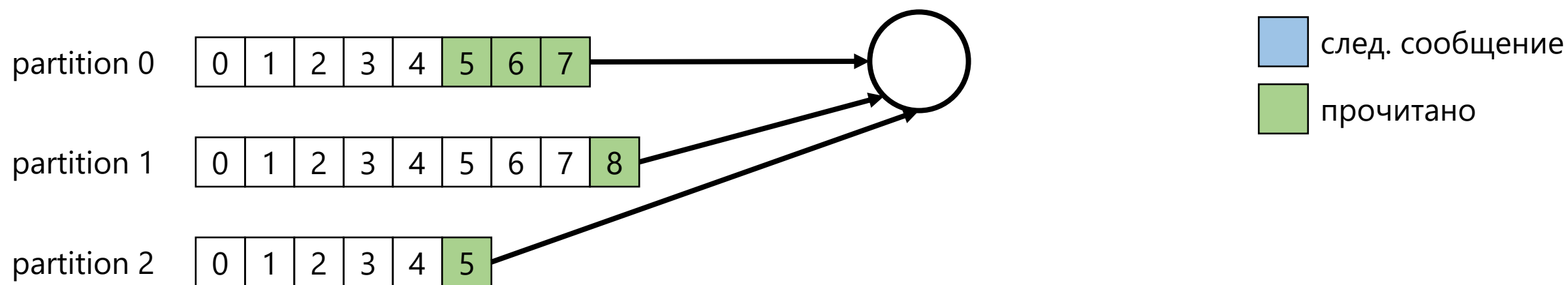
Kafka Consumer



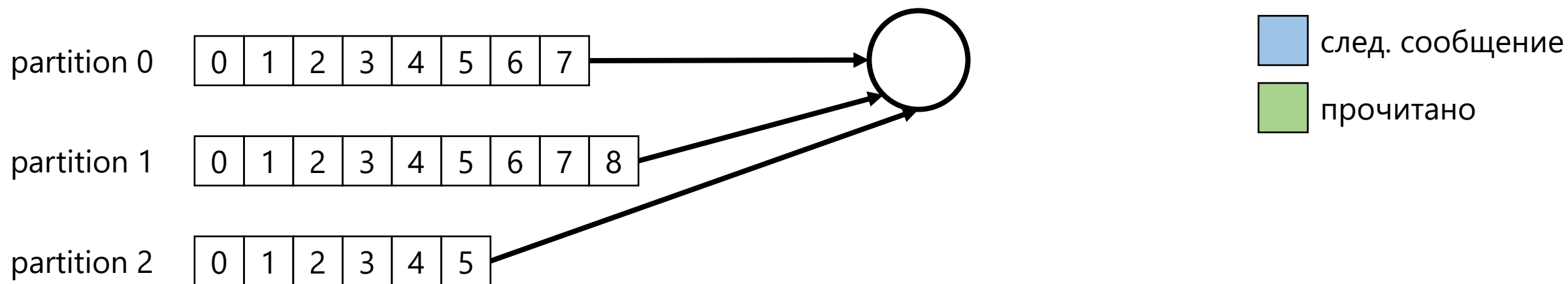
Kafka Consumer



Kafka Consumer



Kafka Consumer



Kafka Consumer

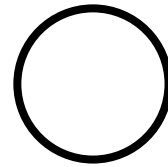
Commit offset

Kafka Consumer

Commit offset

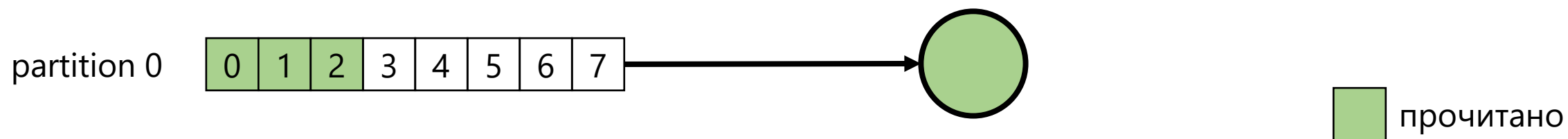
partition 0

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---



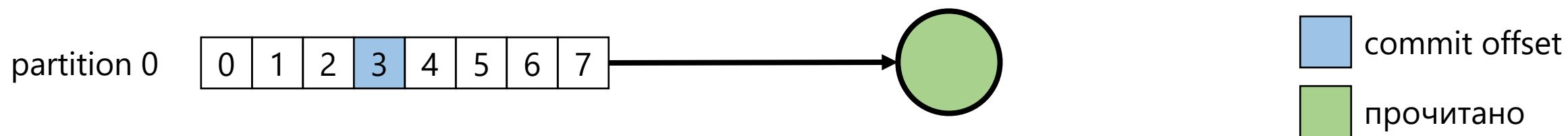
Kafka Consumer

Commit offset



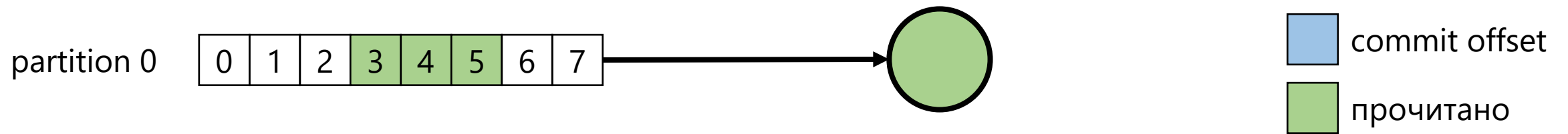
Kafka Consumer

Commit offset



Kafka Consumer

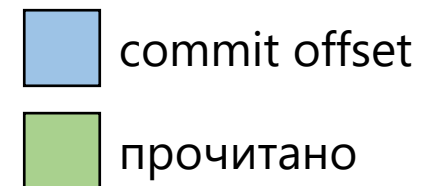
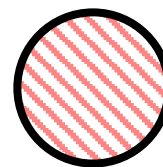
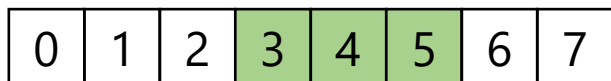
Commit offset



Kafka Consumer

Commit offset

partition 0



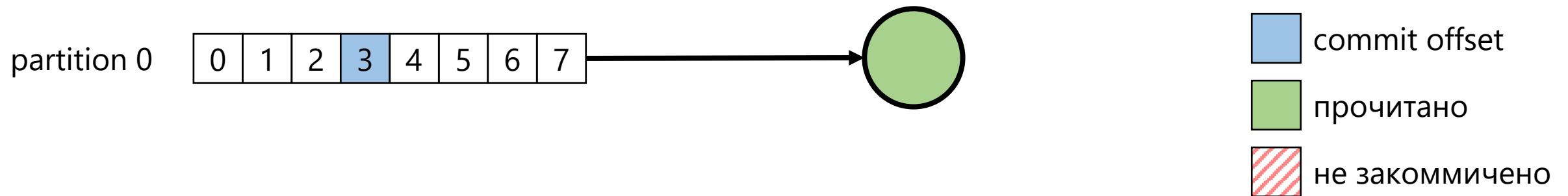
Kafka Consumer

Commit offset



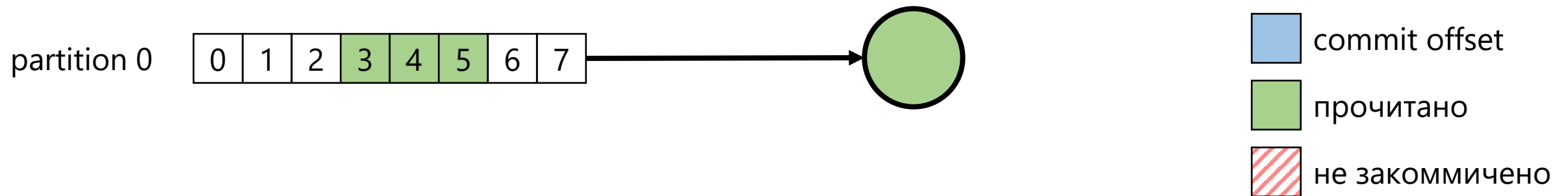
Kafka Consumer

Commit offset



Kafka Consumer

Commit offset



Kafka Consumer

Гарантии обработки

- at least once
- mostly once

Kafka Consumer

```
while (true) {  
    /* */  
    ConsumerRecords<UUID, Event> records =  
        consumer.poll(timer.toDuration());  
  
    consumer.commitAsync();  
    process(records);  
    /* */  
}
```


Kafka Consumer

```
while (true) {  
    /* */  
    ConsumerRecords<UUID, Event> records =  
        consumer.poll(timer.toDuration());  
  
    consumer.commitAsync();  
    process(records);  
    /* */  
}
```

mostly once

Commit до обработки

Kafka Consumer

```
while (true) {  
    /* */  
    ConsumerRecords<UUID, Event> records =  
        consumer.poll(timer.toDuration());  
  
    process(records);  
    consumer.commitAsync();  
    /* */  
}
```

Kafka Consumer

```
while (true) {  
    /* */  
    ConsumerRecords<UUID, Event> records =  
        consumer.poll(timer.toDuration());  
  
    process(records);  
    consumer.commitAsync();  
    /* */  
}
```

at least once

Commit после обработки

Kafka Consumer

```
while (true) {  
    /* */  
    ConsumerRecords<UUID, Event> records =  
        consumer.poll(timer.toDuration());  
  
    process(records);  
    consumer.commitAsync();  
    /* */  
}
```



Kafka Consumer

- 1 коммит на 1 сообщение
- 1 коммит на N сообщений

Kafka Consumer

- 1 коммит на 1 сообщение
- 1 коммит на N сообщений ✓

↑ производительность

Kafka Consumer

```
# Consumer properties
```

```
enable.auto.commit = true | false  
auto.commit.interval.ms = 5000
```

Kafka Consumer

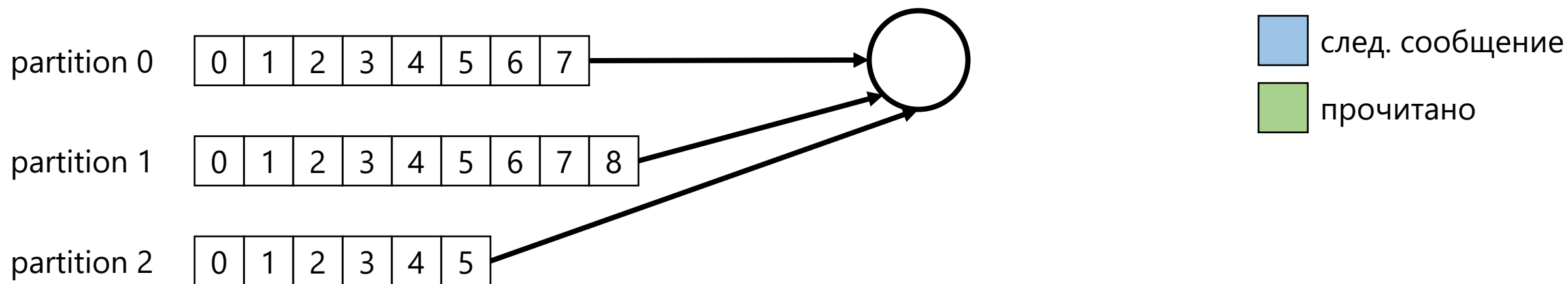
```
# Consumer properties
```

```
enable.auto.commit = true | false  
auto.commit.interval.ms = 5000
```

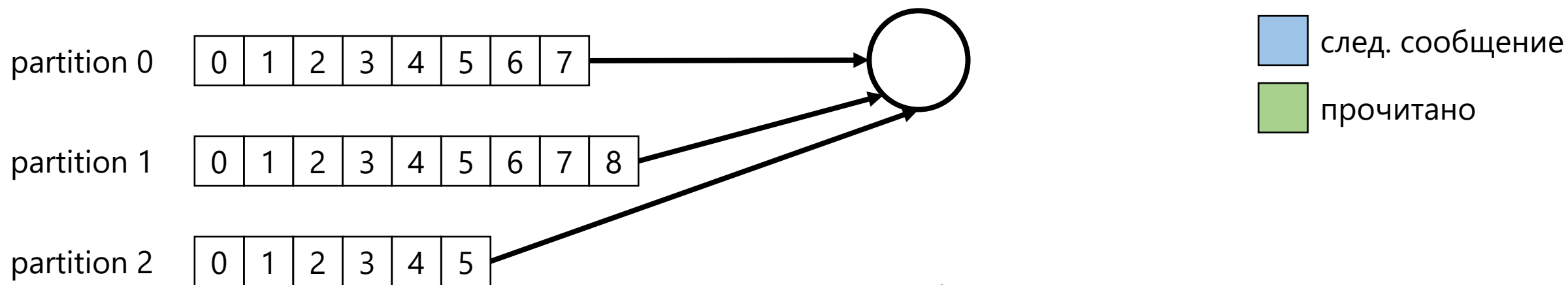

Kafka Consumer

```
while (true) {  
    /* */  
    ConsumerRecords<UUID, Event> records =  
        consumer.poll(timer.toDuration());  
  
    process(records);  
    /* */  
}
```

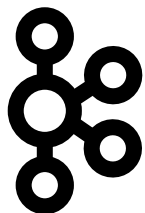
Kafka Consumer



Kafka Consumer



Какое сообщение читать следующим, если ещё ничего не коммитили?



Kafka Consumer

```
# Consumer properties
```

```
auto.offset.reset =  
    earliest |  
    latest |  
    none
```

Kafka Consumer

```
# Consumer properties
```

```
auto.offset.reset =  
    earliest |  
    latest |  
    none
```

Kafka Consumer

```
# Consumer properties
```

```
auto.offset.reset =  
    earliest |  
    latest |  
    none
```

Kafka Consumer

```
# Consumer properties
```

```
auto.offset.reset =  
    earliest |  
    latest |  
    none
```



Значение по умолчанию

Kafka Consumer

```
# Consumer properties
```

```
auto.offset.reset =  
    earliest |  
    latest |  
    none
```


Kafka Consumer

```
# Consumer properties
```

```
auto.offset.reset =  
    earliest |  
    latest |  
    none
```



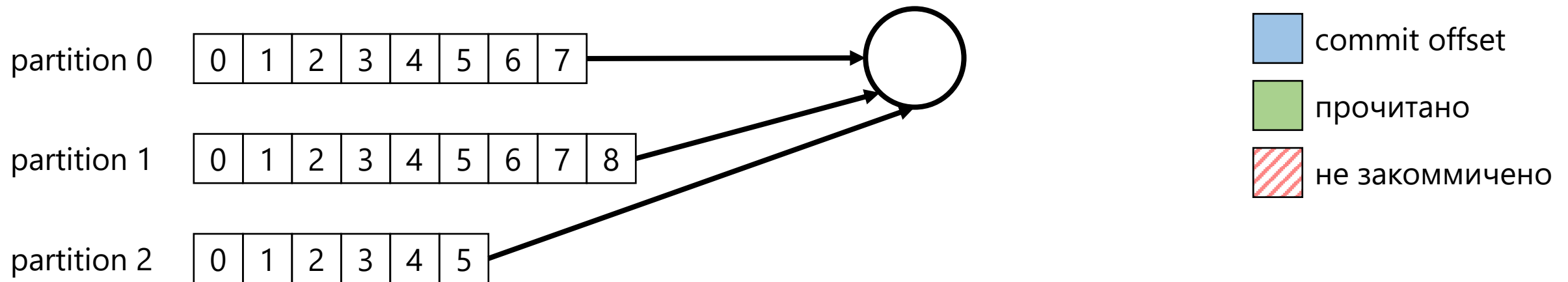
Если нет сохранённого Commit Offset
— будет exception

Kafka Consumer

Consumer Group

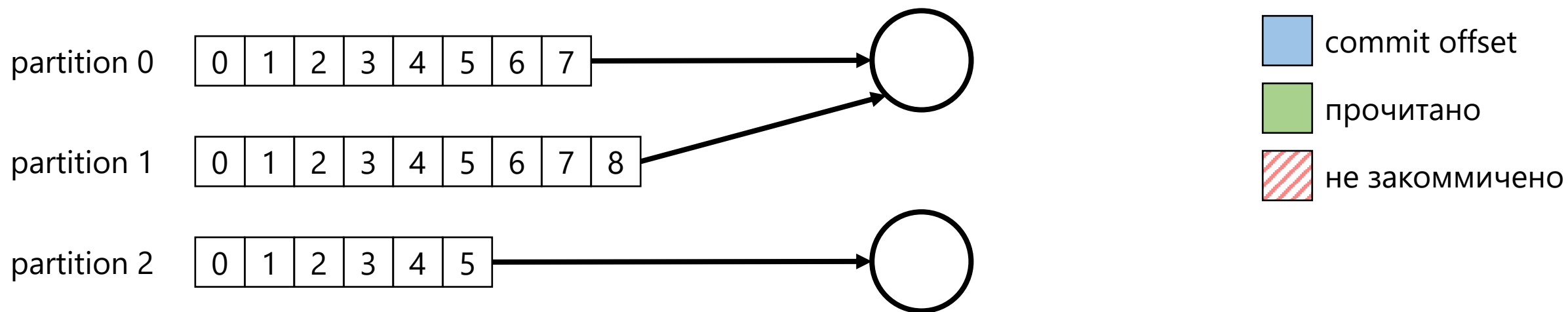
Kafka Consumer

Consumer Group



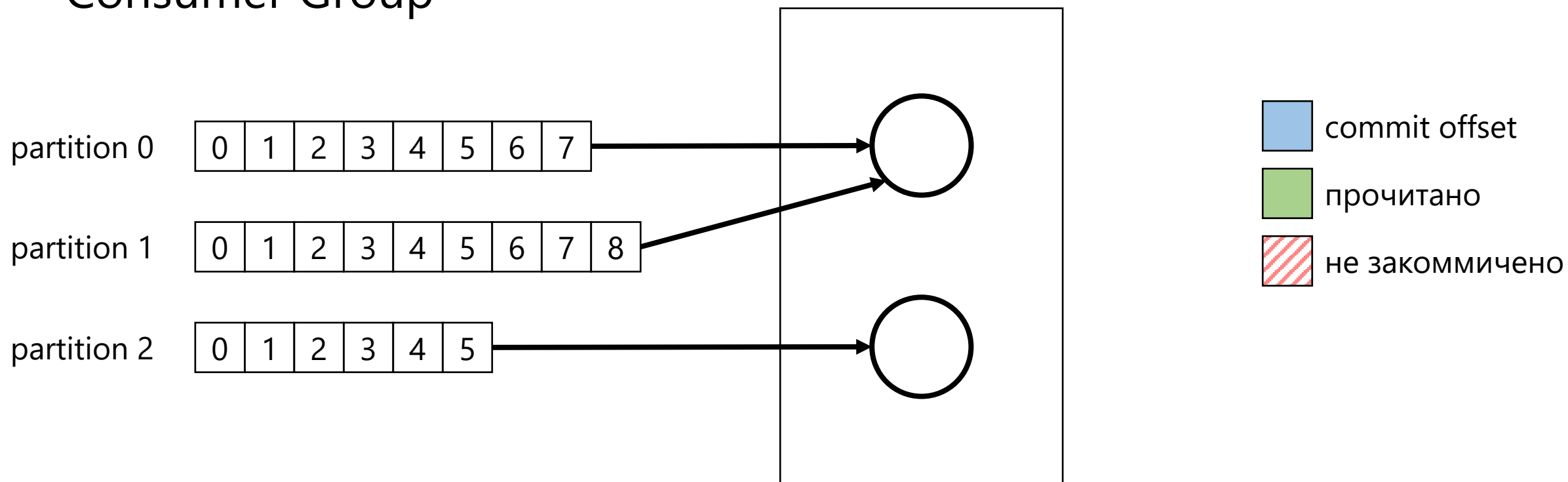
Kafka Consumer

Consumer Group



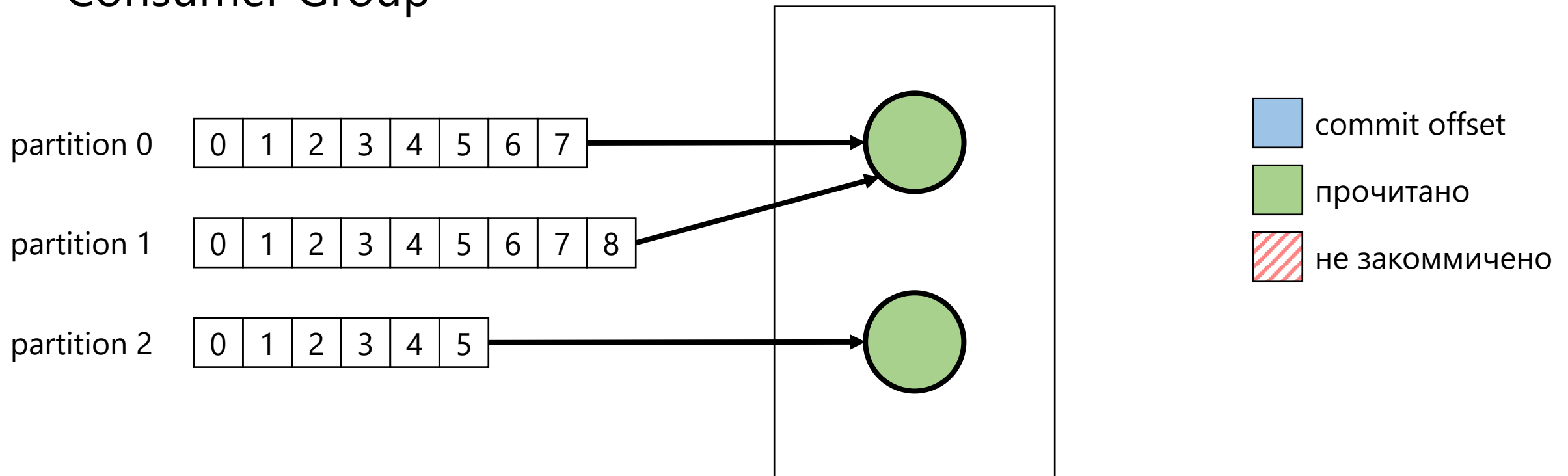
Kafka Consumer

Consumer Group



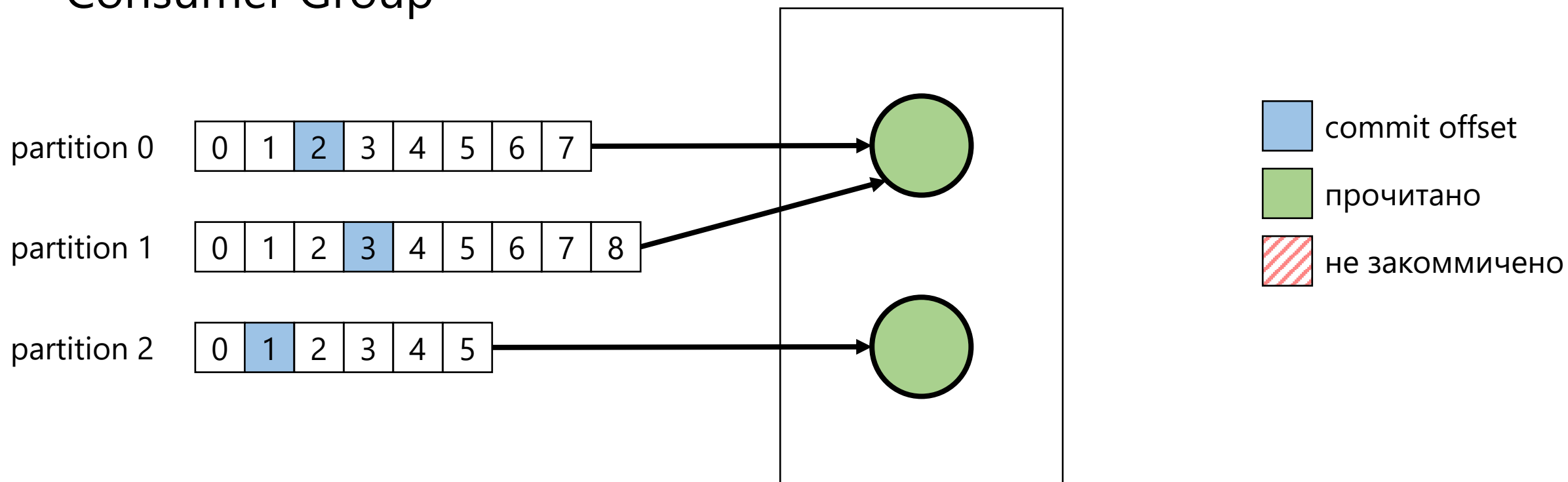
Kafka Consumer

Consumer Group



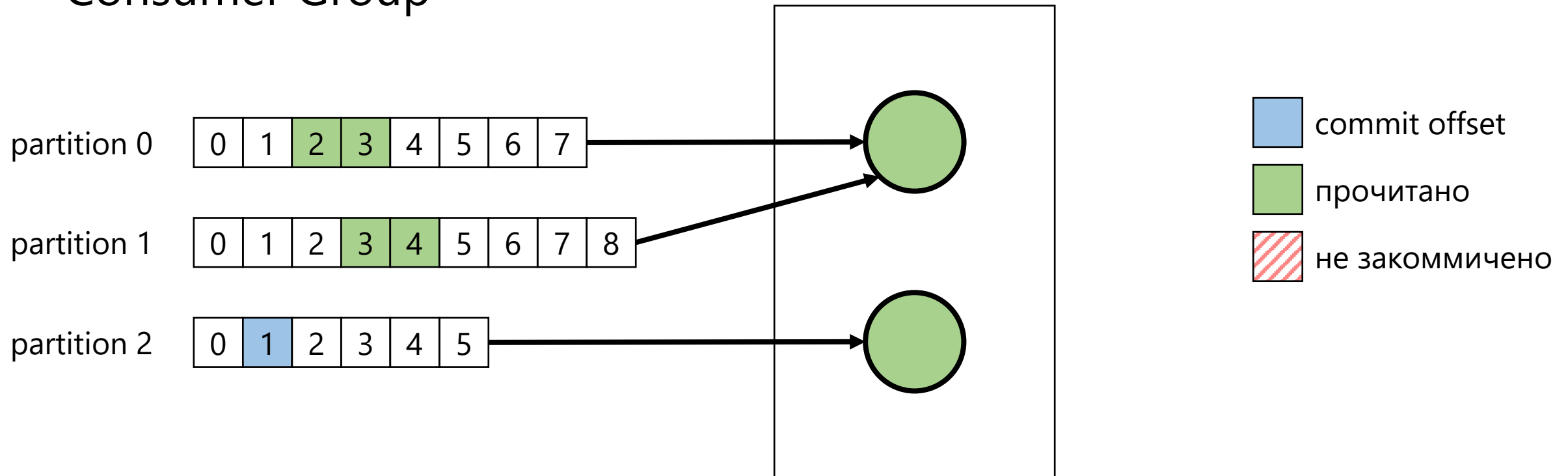
Kafka Consumer

Consumer Group



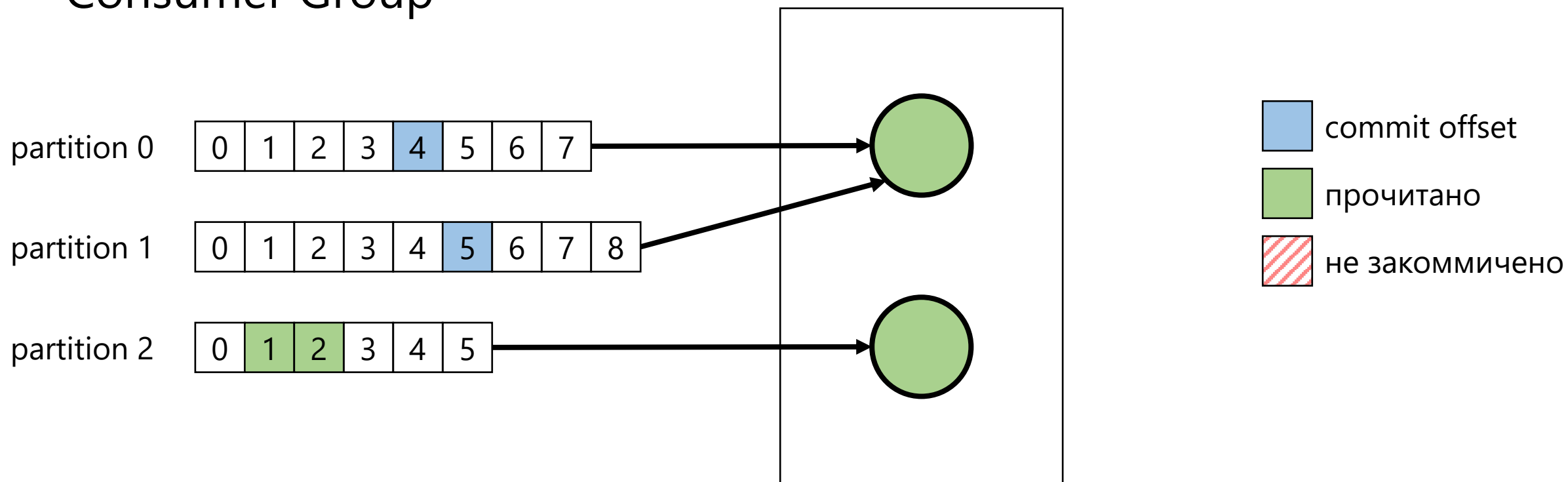
Kafka Consumer

Consumer Group



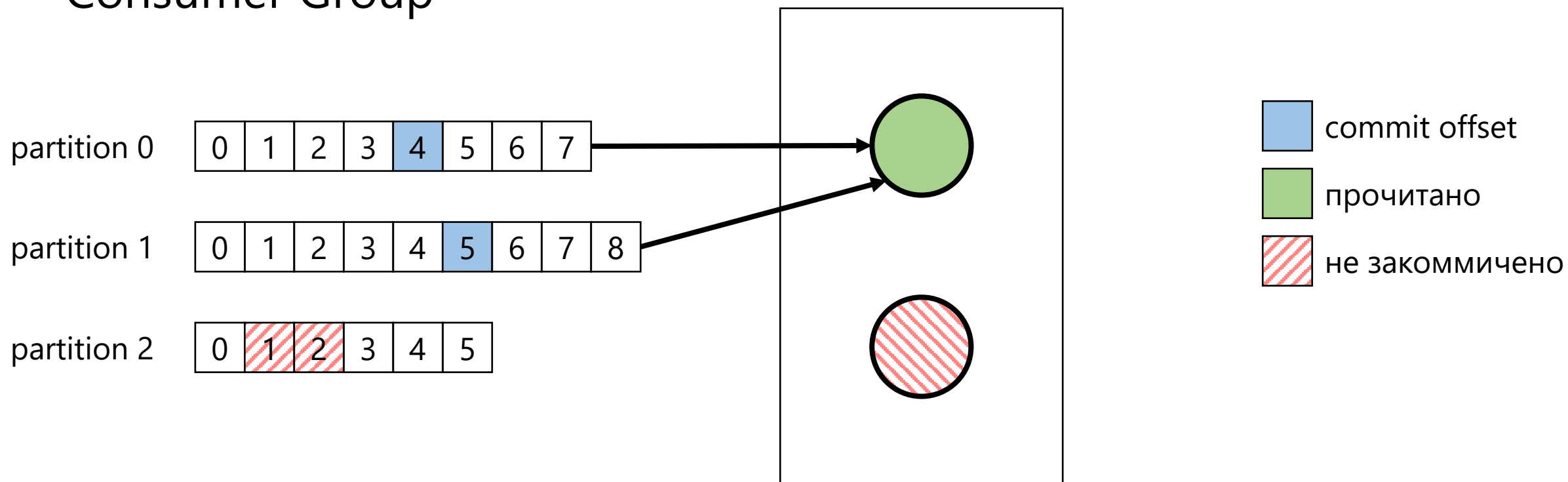
Kafka Consumer

Consumer Group



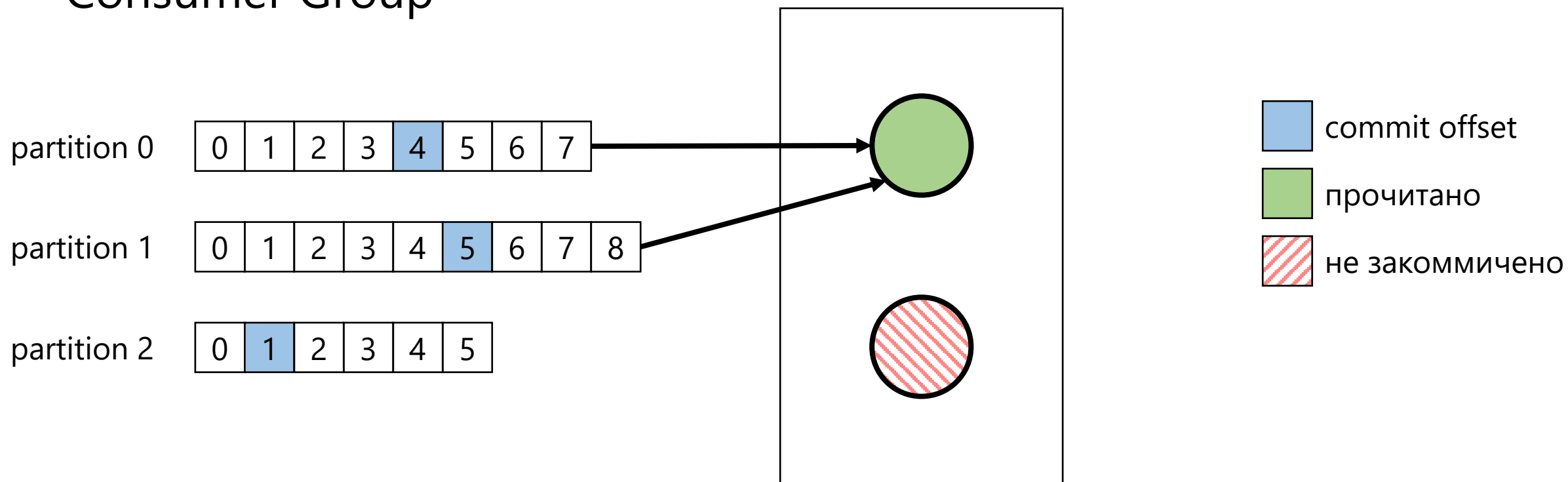
Kafka Consumer

Consumer Group



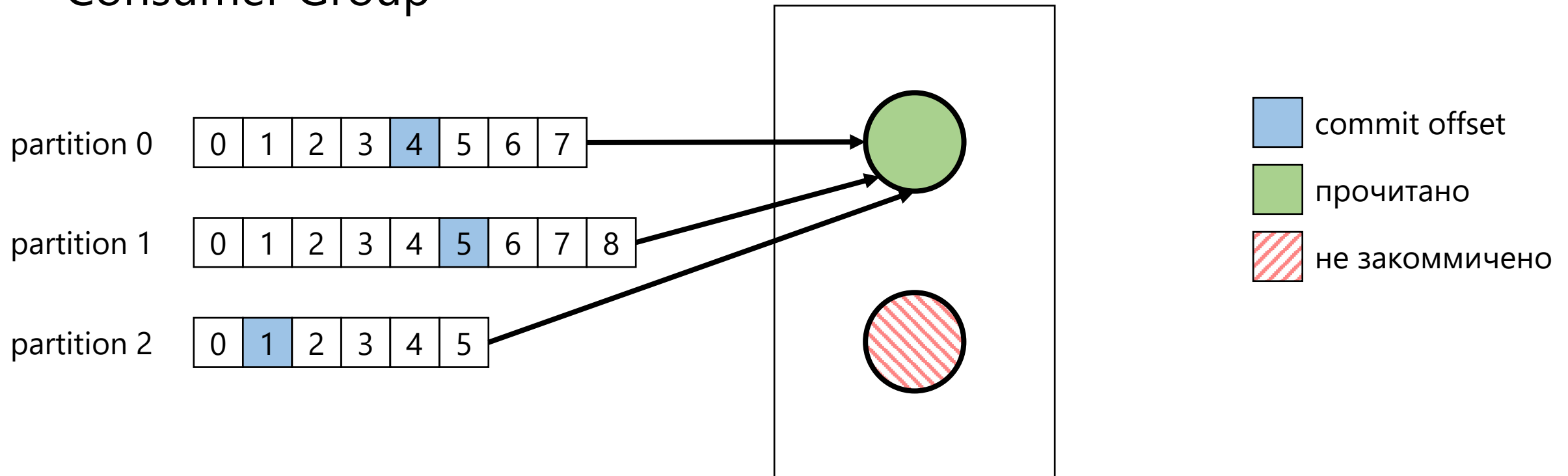
Kafka Consumer

Consumer Group



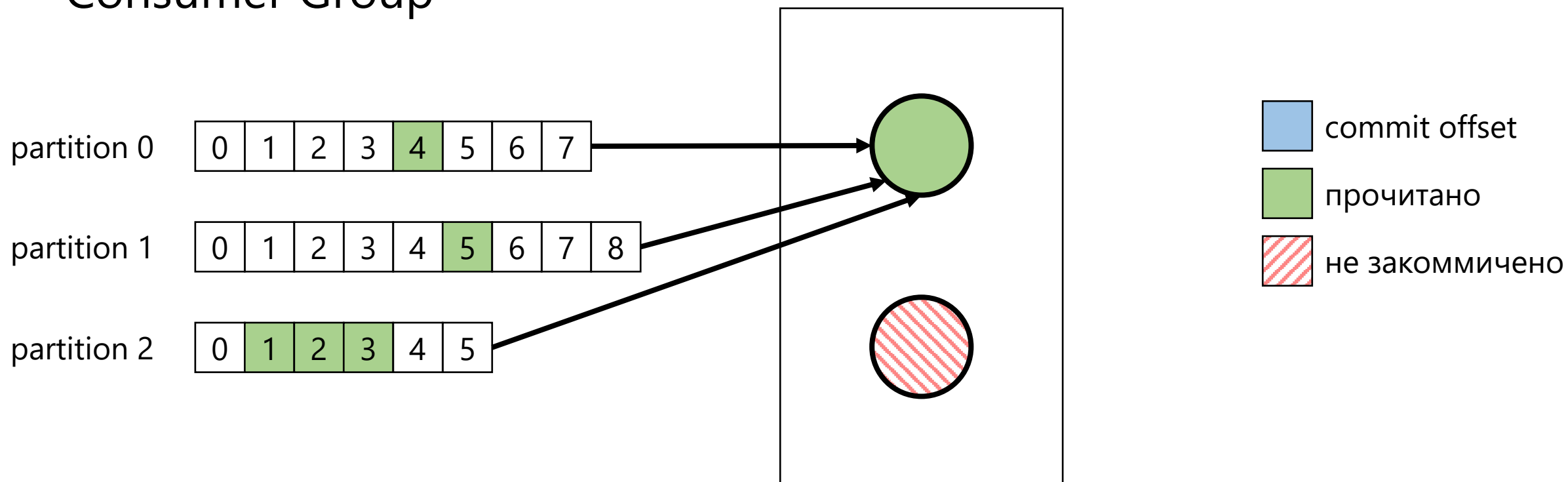
Kafka Consumer

Consumer Group



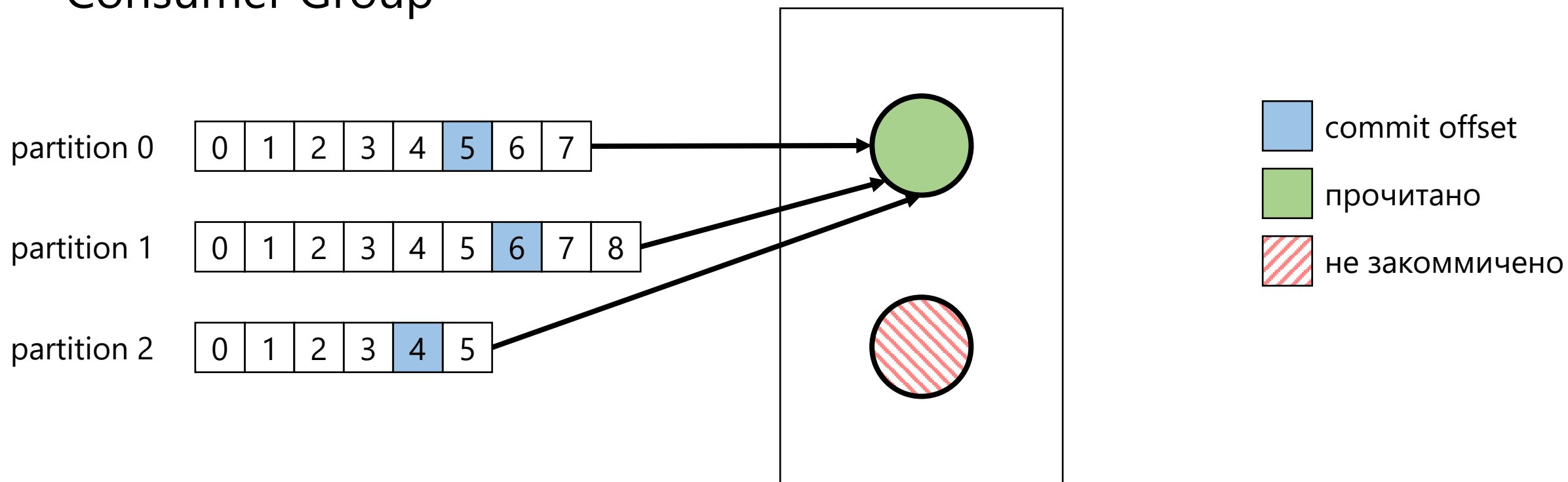
Kafka Consumer

Consumer Group



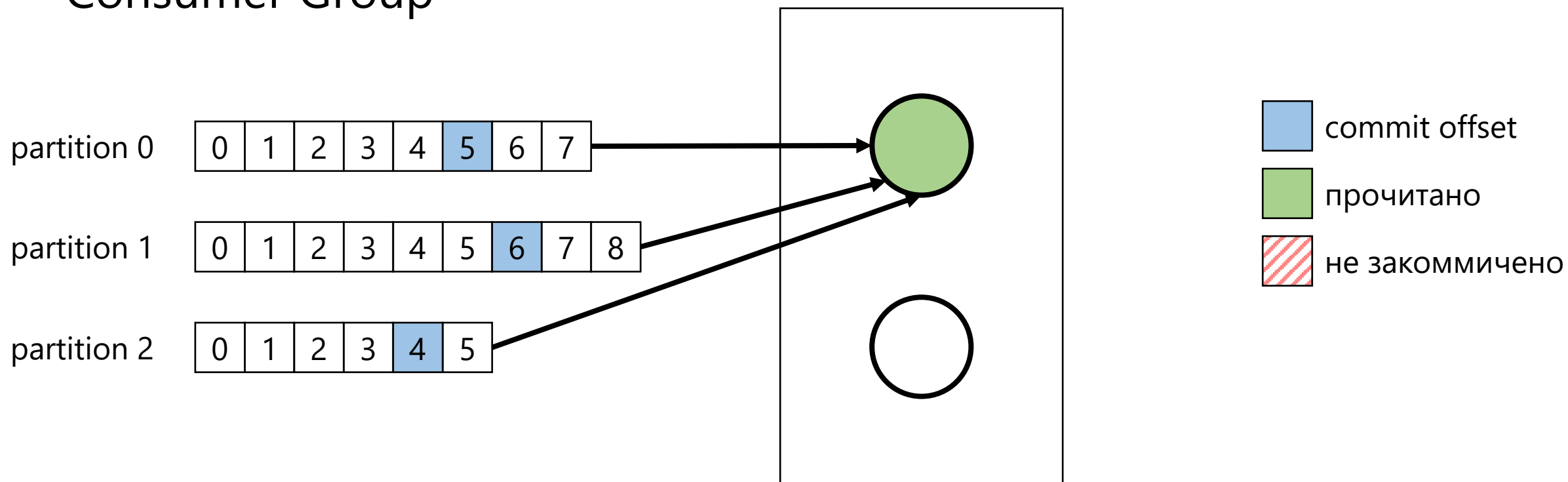
Kafka Consumer

Consumer Group



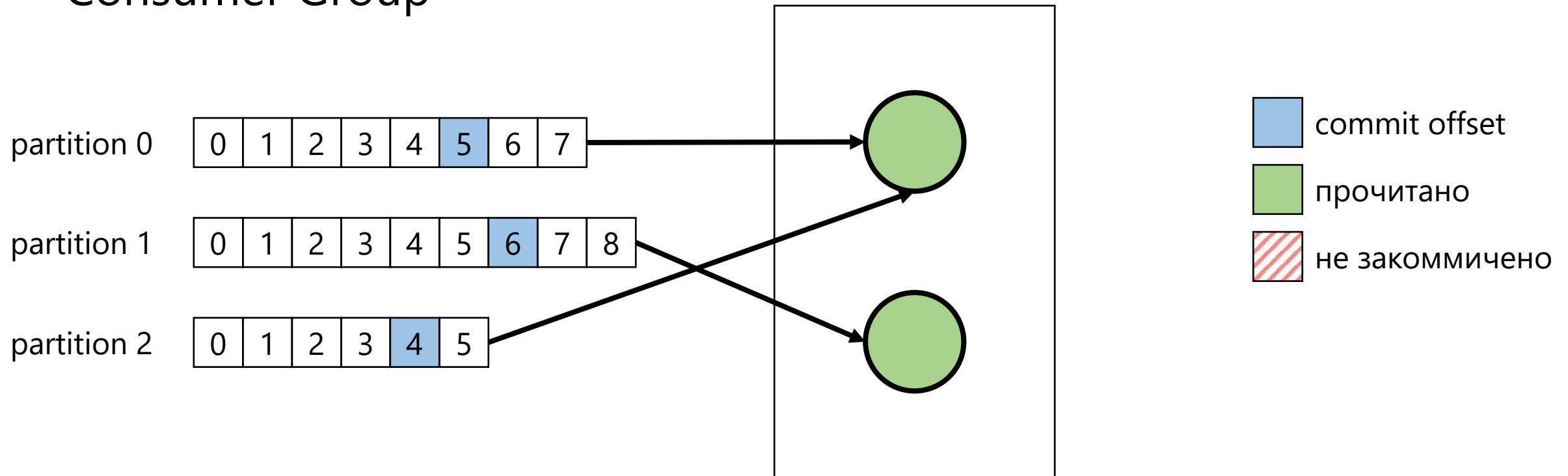
Kafka Consumer

Consumer Group



Kafka Consumer

Consumer Group

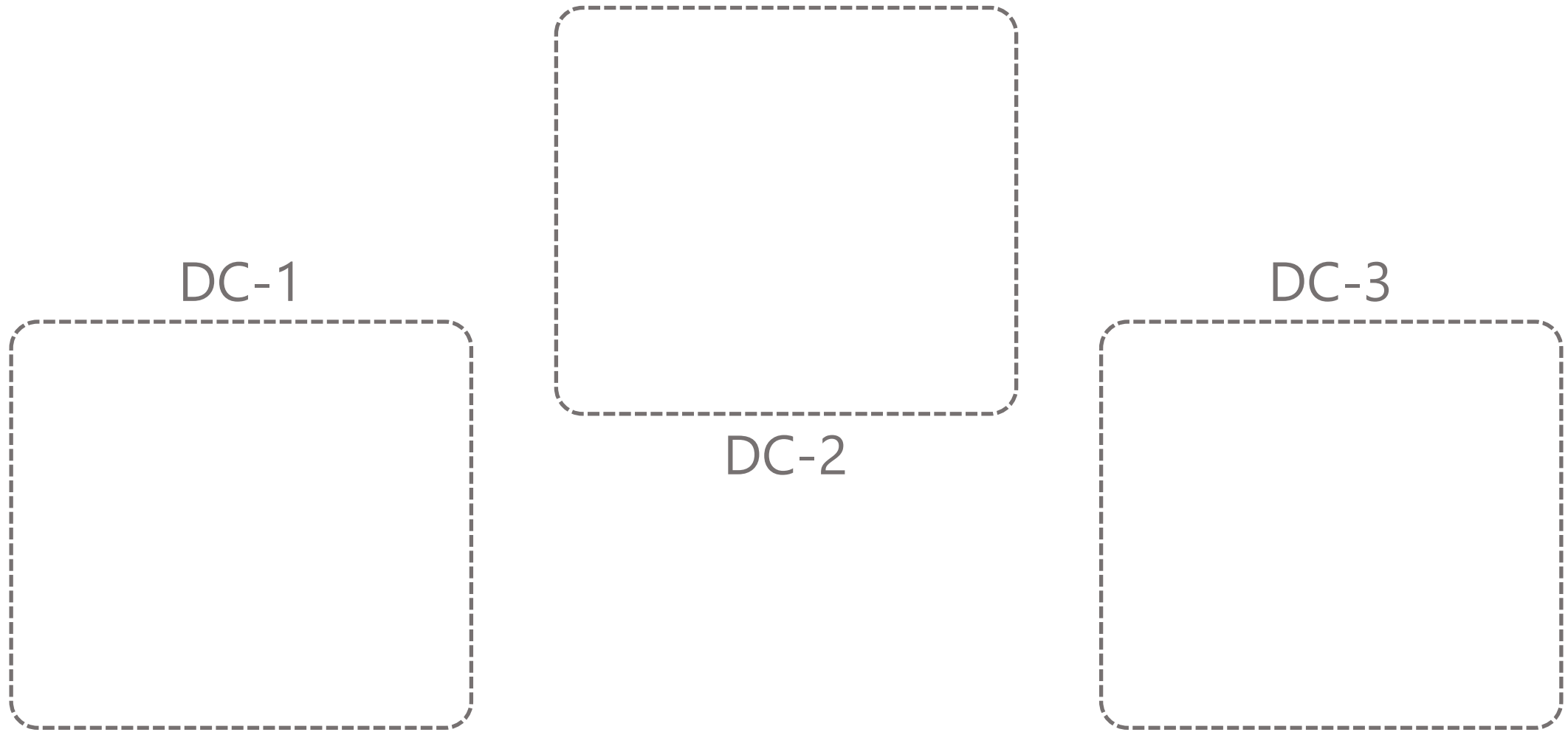


Kafka Consumer

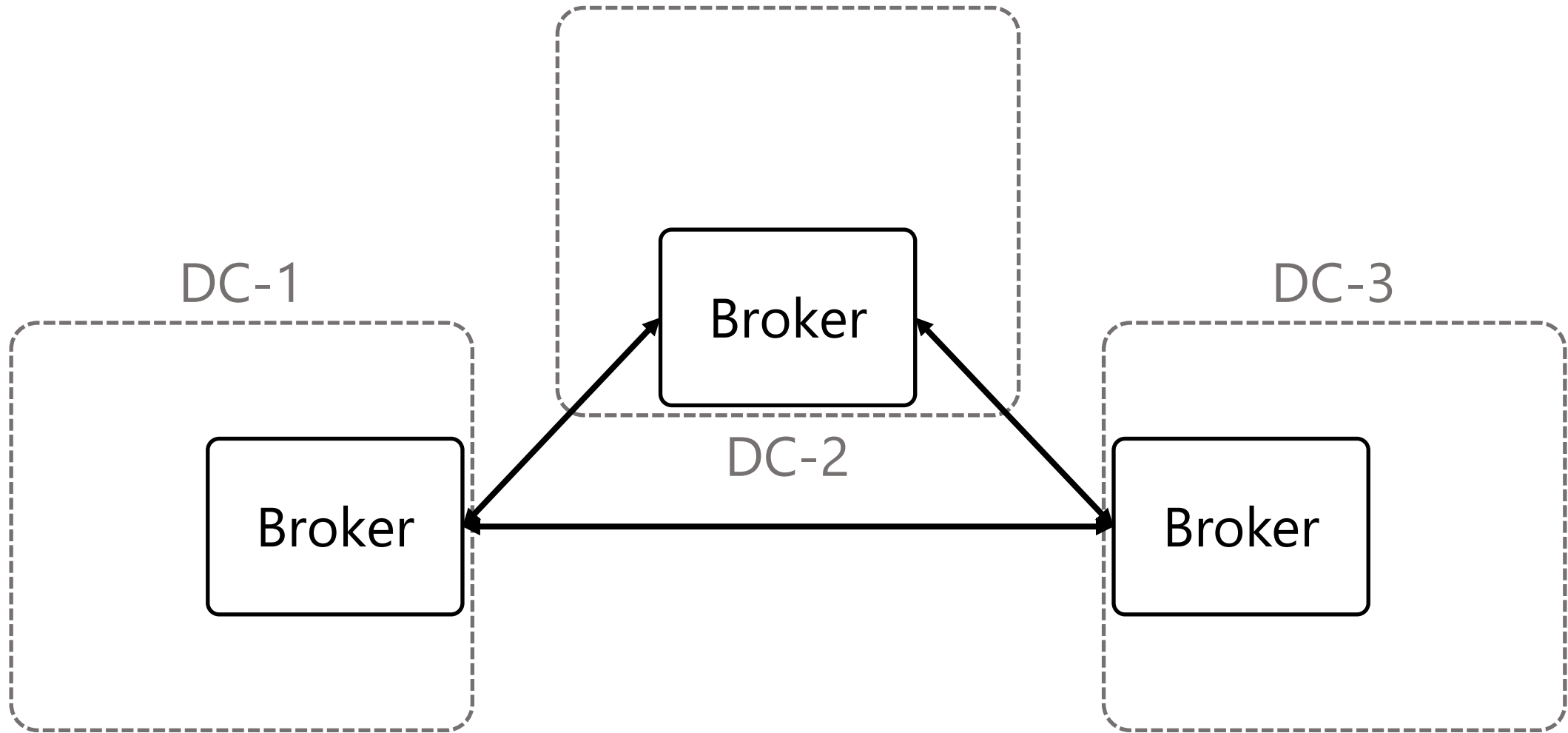
```
# Consumer properties
```

```
group.id =  
    consumer-group-name
```

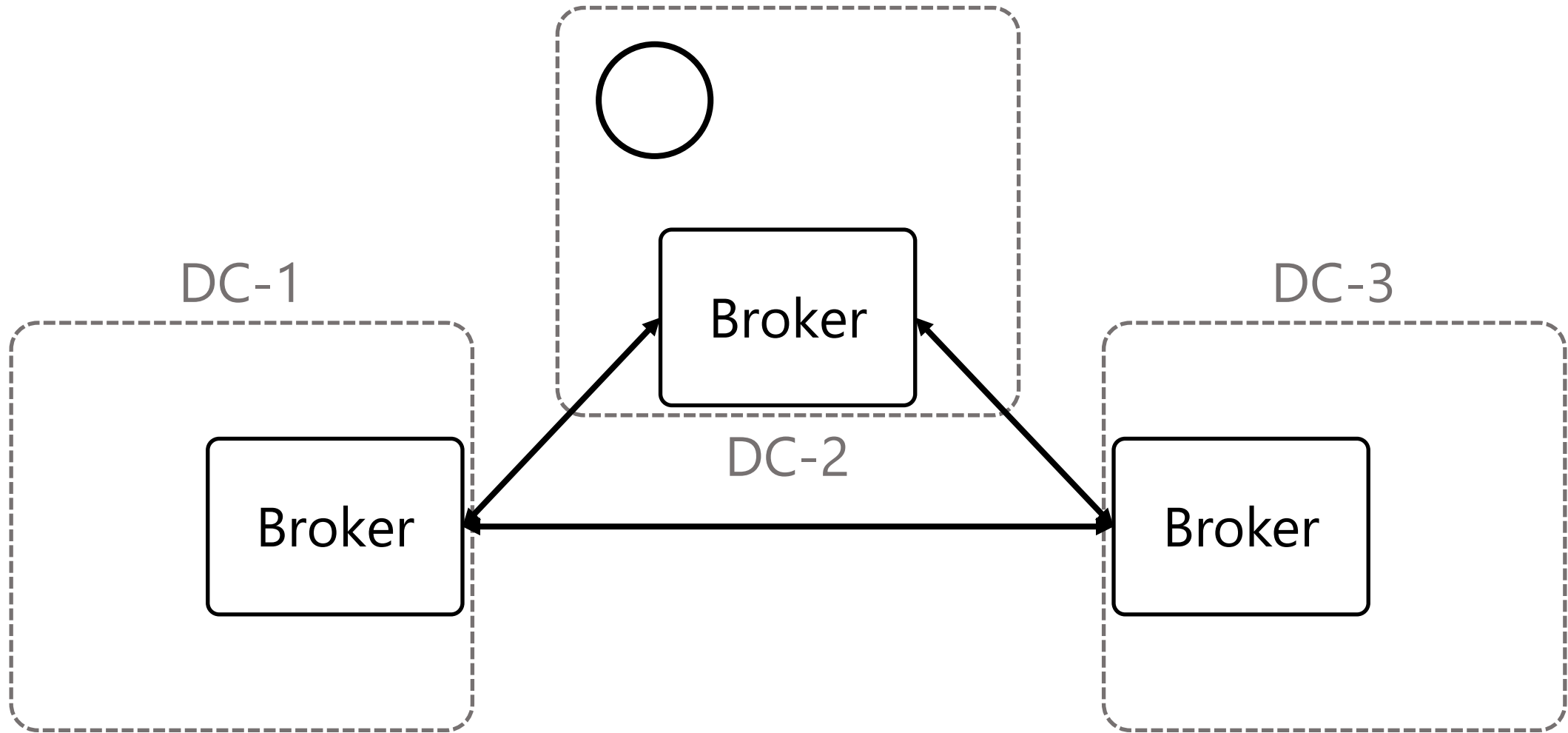
Kafka Consumer



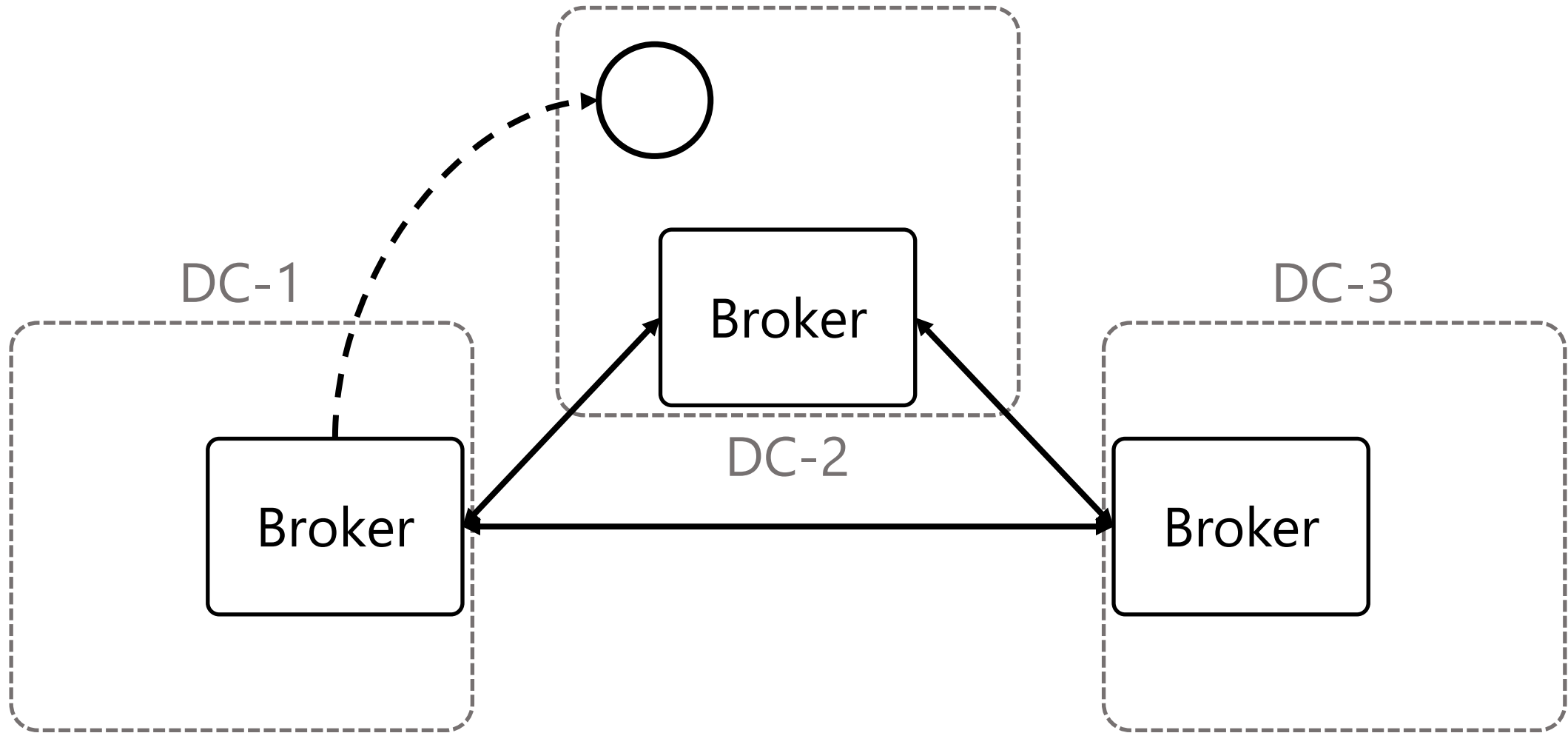
Kafka Consumer



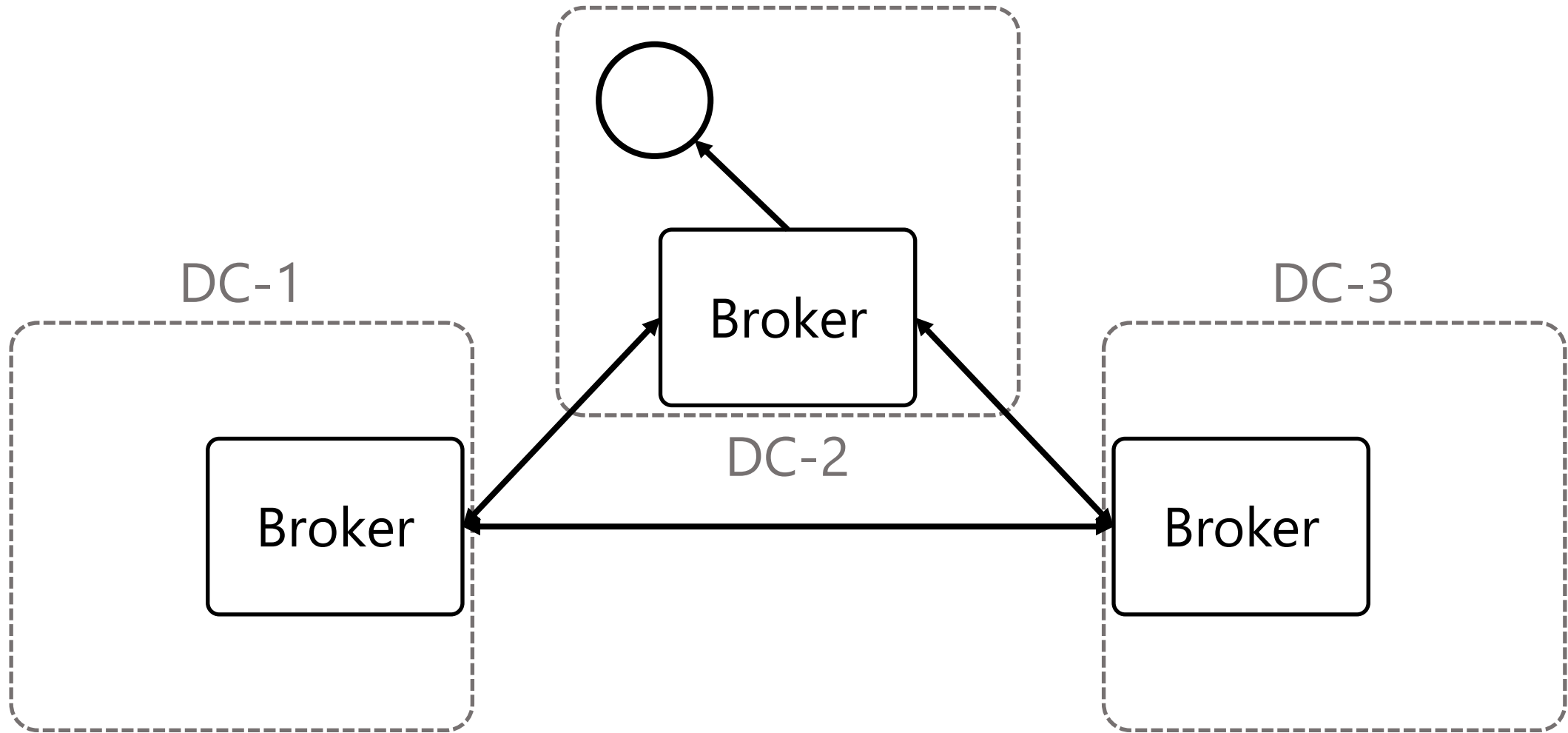
Kafka Consumer



Kafka Consumer

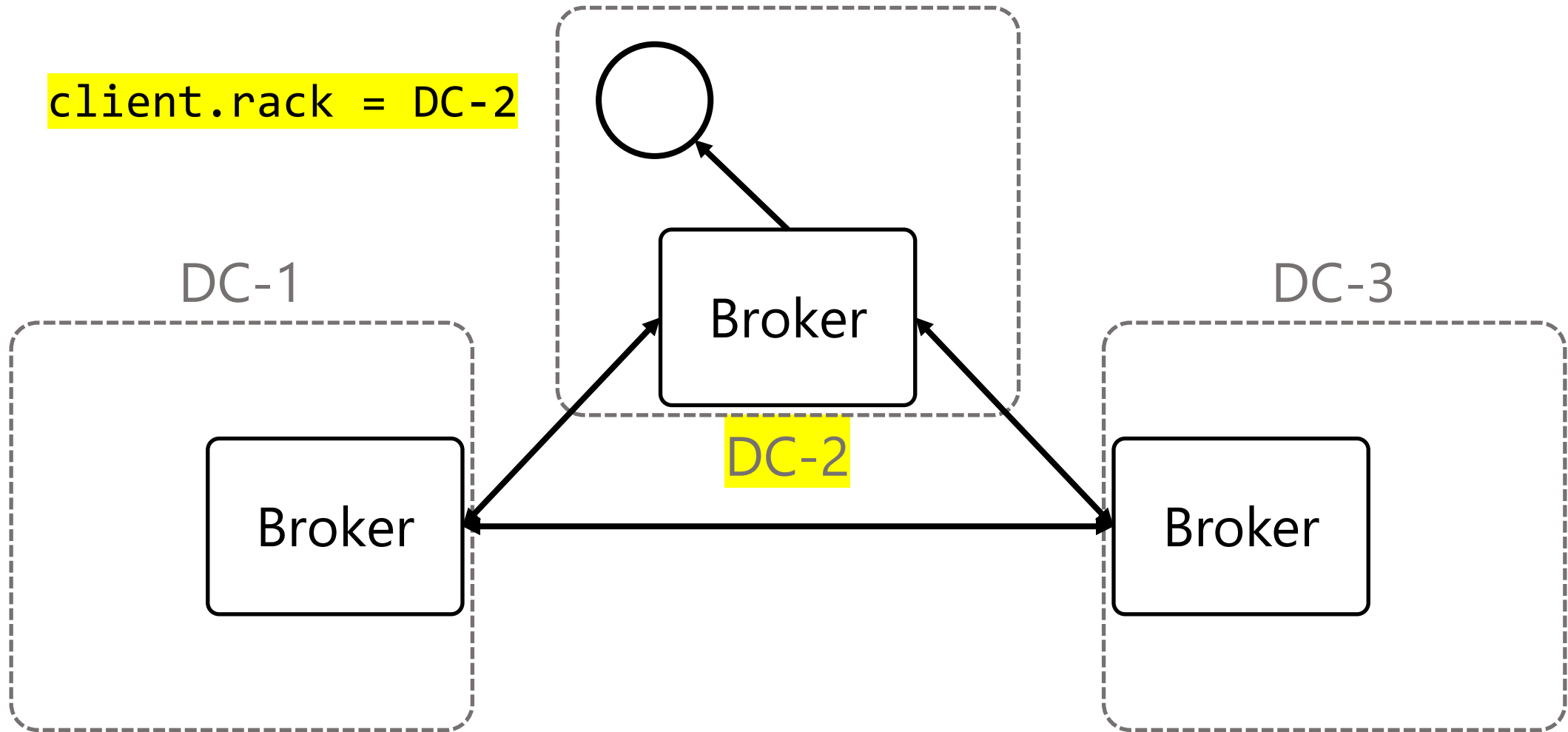


Kafka Consumer



Kafka Consumer

`client.rack = DC-2`



Kafka Consumer

Выводы

- "Smart" Consumer
- Consumer поллит Кафку
- Consumer отвечает за гарантию обработки
- Автоматический фейловер в Consumer-группе
- Независимая обработка разными Consumer-группами

Преимущества Apache Kafka

Преимущества Apache Kafka

— Персистентность данных

Преимущества Apache Kafka

- Персистентность данных
- Высокая производительность

Преимущества Apache Kafka

- Персистентность данных
- Высокая производительность
- Независимость пайплайнов обработки

Преимущества Apache Kafka

- Персистентность данных
- Высокая производительность
- Независимость пайплайнов обработки
- Возможность «проиграть» историю заново

Преимущества Apache Kafka

- Персистентность данных
- Высокая производительность
- Независимость пайплайнов обработки
- Возможность «проиграть» историю заново
- Гибкость в использовании (благодаря простоте)

Когда Apache Kafka 🔥

- Персистентность данных
- Высокая производительность
- Независимость пайплайнов обработки
- Возможность «проиграть» историю заново
- Гибкость в использовании (благодаря простоте)

Когда Apache Kafka 🔥

— **λ** -архитектура и **K**-архитектура

Когда Apache Kafka 🔥

- **λ**-архитектура и **K**-архитектура
- Стриминг данных

Когда Apache Kafka 🔥

- **λ**-архитектура и **K**-архитектура
- Стриминг БОЛЬШИХ данных

Когда Apache Kafka 🔥

- **λ**-архитектура и **K**-архитектура
- Стриминг БОЛЬШИХ данных
- Много клиентов (Producer и Consumer)

Когда Apache Kafka 🔥

- **λ**-архитектура и **K**-архитектура
- Стриминг БОЛЬШИХ данных
- Много клиентов (Producer и Consumer)
- Требуется кратное масштабирование

Когда Apache Kafka 🔥

- **λ**-архитектура и **K**-архитектура
- Стриминг БОЛЬШИХ данных
- Много клиентов (Producer и Consumer)
- Требуется кратное масштабирование
- Велосипедостроение

Чего нет в Kafka из коробки

Чего нет в Kafka из коробки



Kafka — это не брокер сообщений!

Чего нет в Kafka из коробки

- Отложенные сообщения
- DLQ
- AMQP / MQTT
- TTL на сообщение
- Очереди с приоритетами

Q/A

Другие доклады и материалы:

https://tg.me/chnl_GregoryKoshelev



Какие есть альтернативы?

- RabbitMQ Streams <https://www.rabbitmq.com/streams.html>
- Apache Pulsar <https://pulsar.apache.org>
- Apache RocketMQ <https://rocketmq.apache.org>