beetech

DotNext 2024

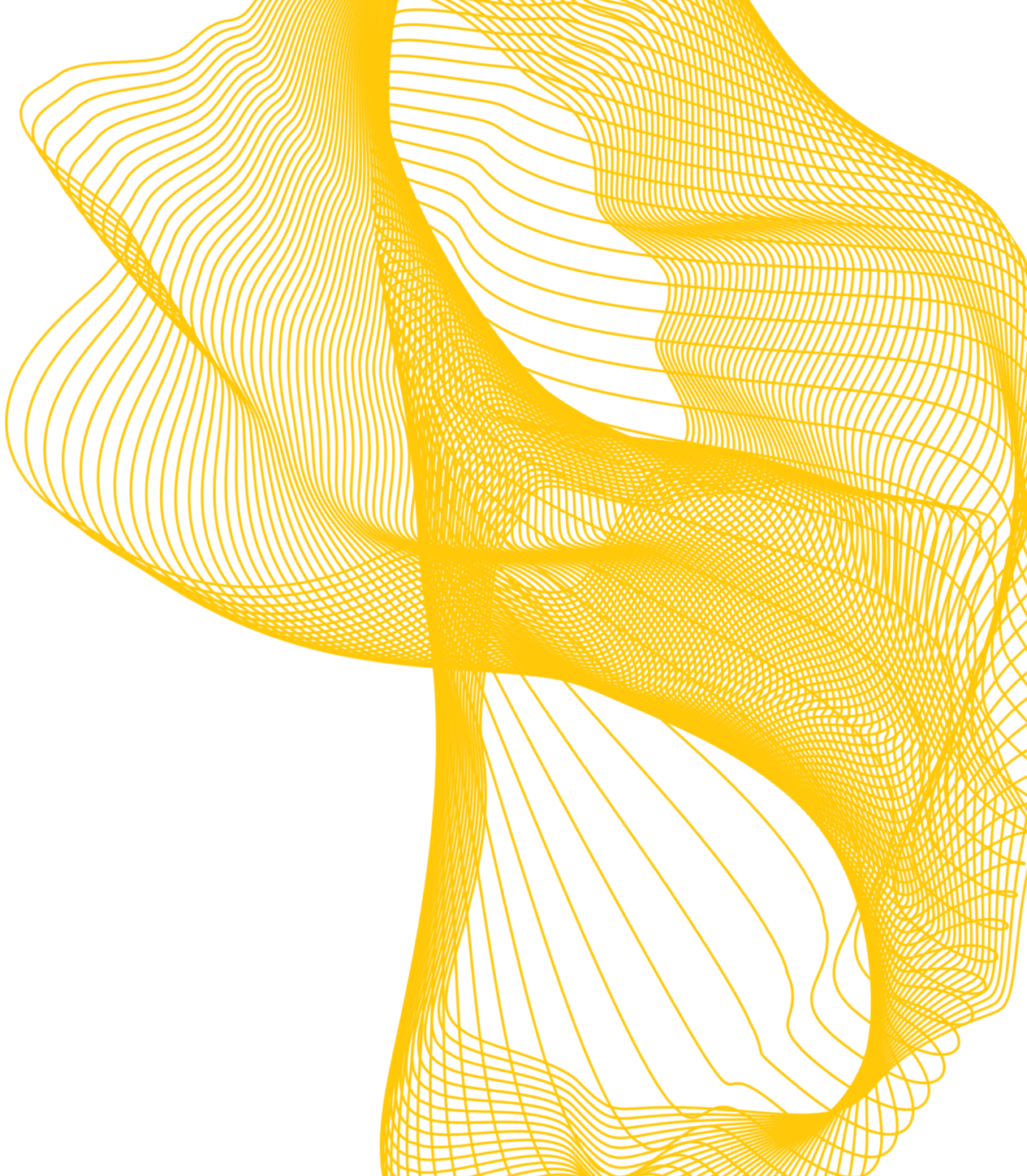
Москва, 10 сентября 2024

beetech

Confluent Kafka:
любовь и ненависть в .net

Алена Мельник,
ведущий разработчик

Москва, 10 сентября 2024



Обо мне

Ведущий разработчик

Занимаюсь развитием референсной архитектуры в компании Билайн

Основные области профессиональных интересов:

- проблемы распределённых систем
- event sourcing
- DDD

Живу в городе Екатеринбург

Алена Мельник

О чем пойдет речь?

1. Топик и партиции. Выбор лидера

О чем пойдет речь?

1. Топик и партиции. Выбор лидера
2. Группы консюмеров

О чем пойдет речь?

1. Топик и партиции. Выбор лидера
2. Группы консюмеров
3. Ребалансировки в кафке и как с этим жить

О чем пойдет речь?

1. Топик и партиции. Выбор лидера
2. Группы консюмеров
3. Ребалансировки в кафке и как с этим жить
4. Нестандартные авторизации

О чем пойдет речь?

1. Топик и партиции. Выбор лидера
2. Группы консюмеров
3. Ребалансировки в кафке и как с этим жить
4. Нестандартные авторизации
5. Немного про баги в confluent и librdkafka

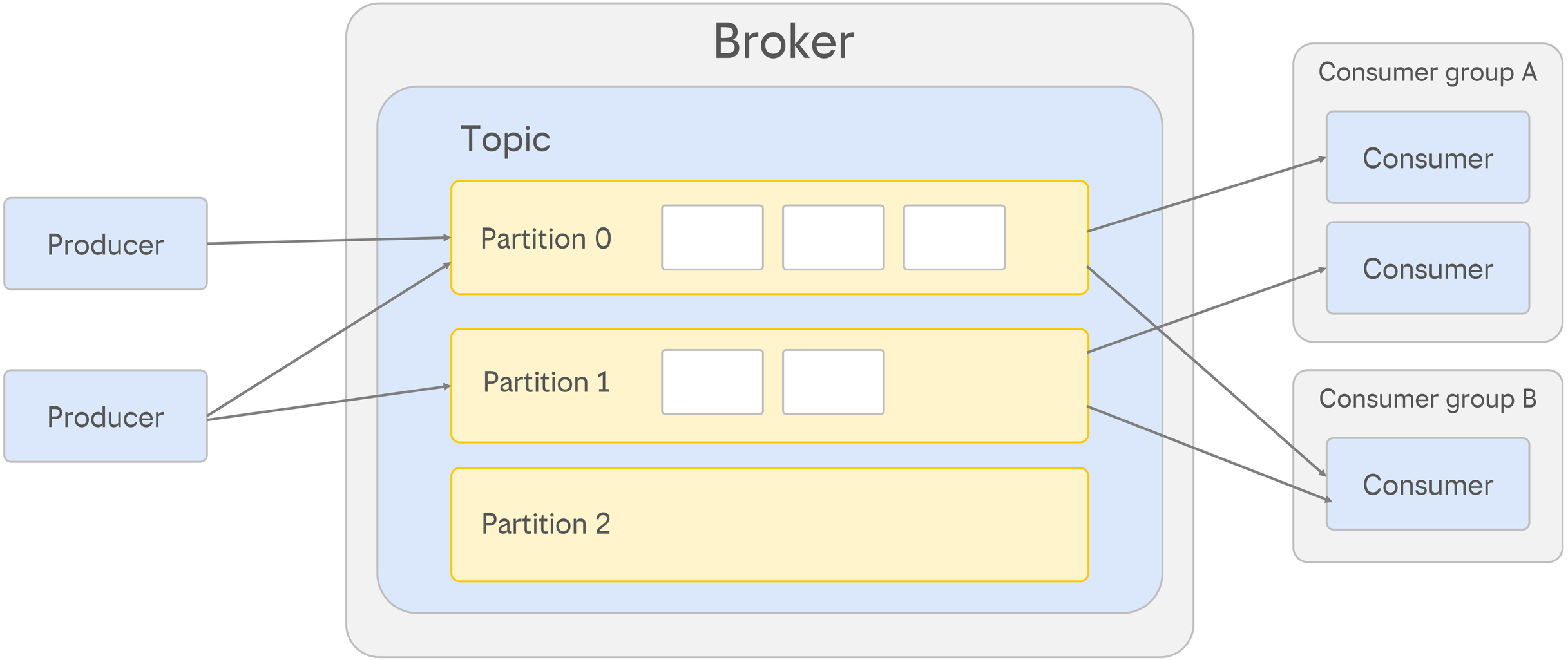
О чем пойдет речь?

1. Топик и партиции. Выбор лидера
2. Группы консюмеров
3. Ребалансировки в кафке и как с этим жить
4. Нестандартные авторизации
5. Немного про баги в confluent и librdkafka
6. Производительность методов продюссеров

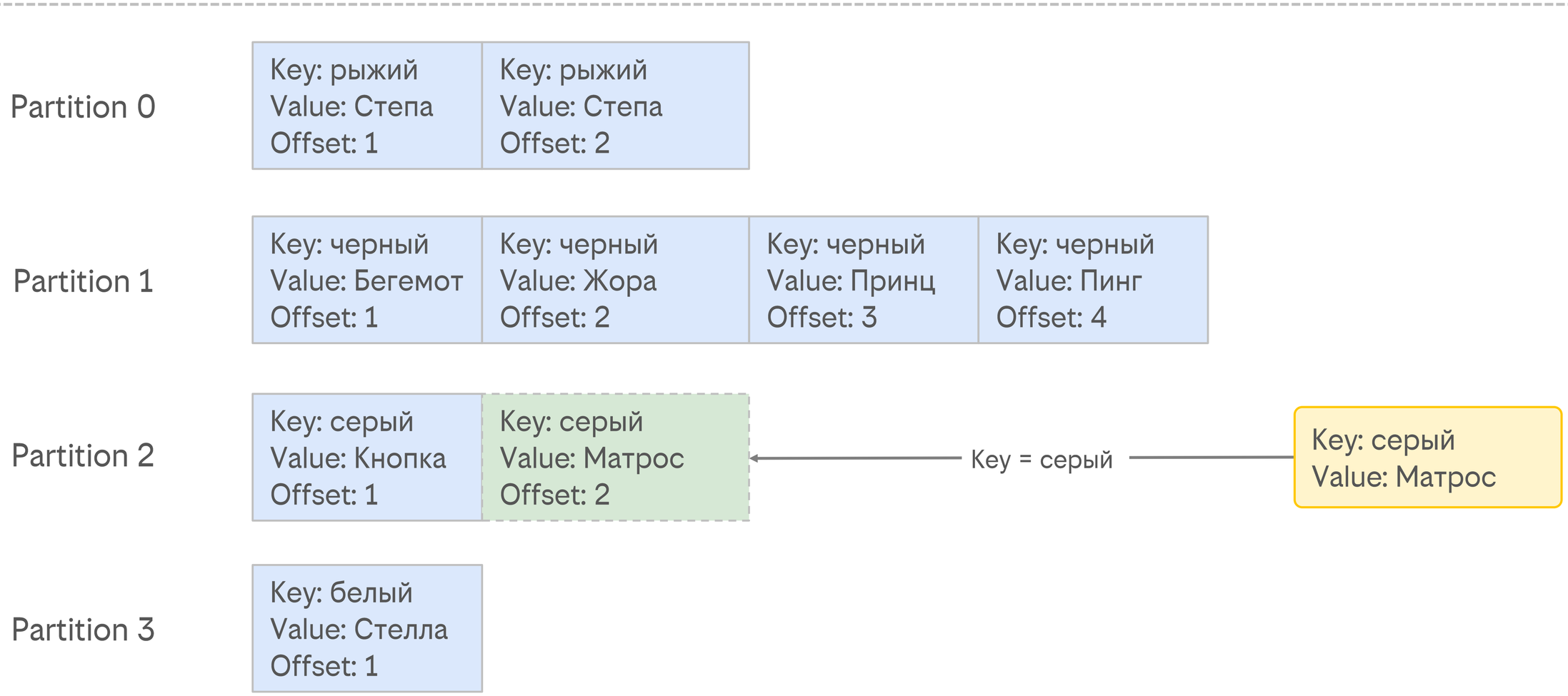
О чем пойдет речь?

1. Топик и партиции. Выбор лидера
2. Группы консюмеров
3. Ребалансировки в кафке и как с этим жить
4. Нестандартные авторизации
5. Немного про баги в confluent и librdkafka
6. Производительность методов продюссеров
7. Многоуровневые хранилища

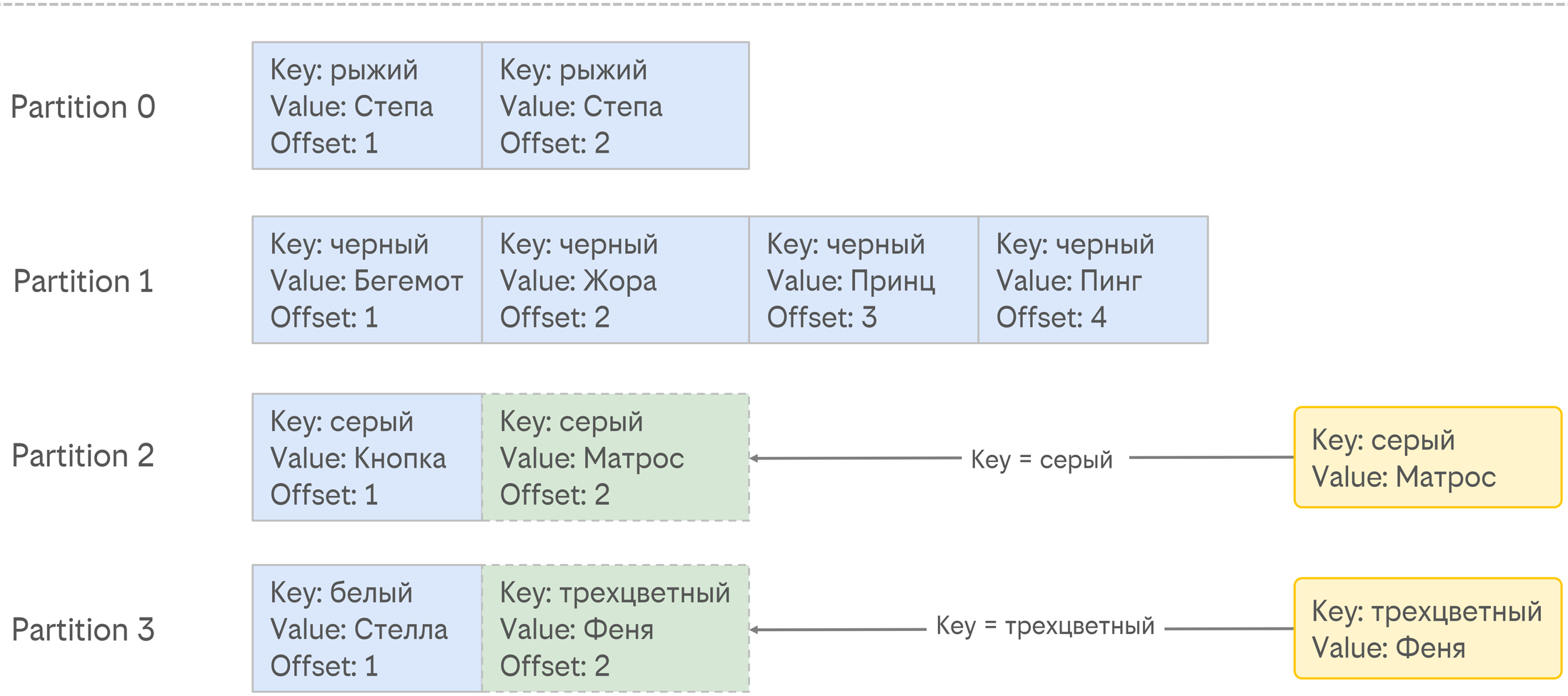
В двух словах о кафке



1 Топик и партиции



2 Топик и партиции



Стратегии распределения по партициям

Strategy	Description
Default partitioner	The key hash is used to map messages to partitions. Null key messages are sent to a partition in a round-robin fashion.
Round-robin partitioner	Messages are sent to partitions in a round-robin fashion.
Uniform sticky partitioner	Messages are sent to a sticky partition (until the batch.size is met or linger.ms time is up) to reduce latency.
Custom partitioner	This approach implements the `Partitioner` interface to override the `partition` method with some custom logic that defines the key-to-partition routing strategy.

Стратегии распределения по партициям

```
producerBuilder.SetPartitioner(topic.Name, (string topicName, int partitionCount, ReadOnlySpan<byte> keyData,
bool keyIsNull) =>
{
    var keyString = System.Text.UTF8Encoding.UTF8.GetString(keyData.ToArray());
    return int.Parse(keyString.Split(" ").Last()) % partitionCount;
});
```

```
producerBuilder.SetDefaultPartitioner((string topicName, int partitionCount, ReadOnlySpan<byte> keyData, bool
keyIsNull) =>
{
    var keyString = System.Text.UTF8Encoding.UTF8.GetString(keyData.ToArray());
    return int.Parse(keyString.Split(" ").Last()) % partitionCount;
});
```

Используется библиотека: Confluent Kafka .Net версии 2.4.0

Стратегии распределения по партициям

```
producerBuilder.SetPartitioner(topic.Name, (string topicName, int partitionCount, ReadOnlySpan<byte> keyData,
bool keyIsNull) =>
{
    var keyString = System.Text.UTF8Encoding.UTF8.GetString(keyData.ToArray());
    return int.Parse(keyString.Split(" ").Last()) % partitionCount;
});
```

```
producerBuilder.SetDefaultPartitioner((string topicName, int partitionCount, ReadOnlySpan<byte> keyData, bool
keyIsNull) =>
{
    var keyString = System.Text.UTF8Encoding.UTF8.GetString(keyData.ToArray());
    return int.Parse(keyString.Split(" ").Last()) % partitionCount;
});
```

Используется библиотека: Confluent Kafka .Net версии 2.4.0

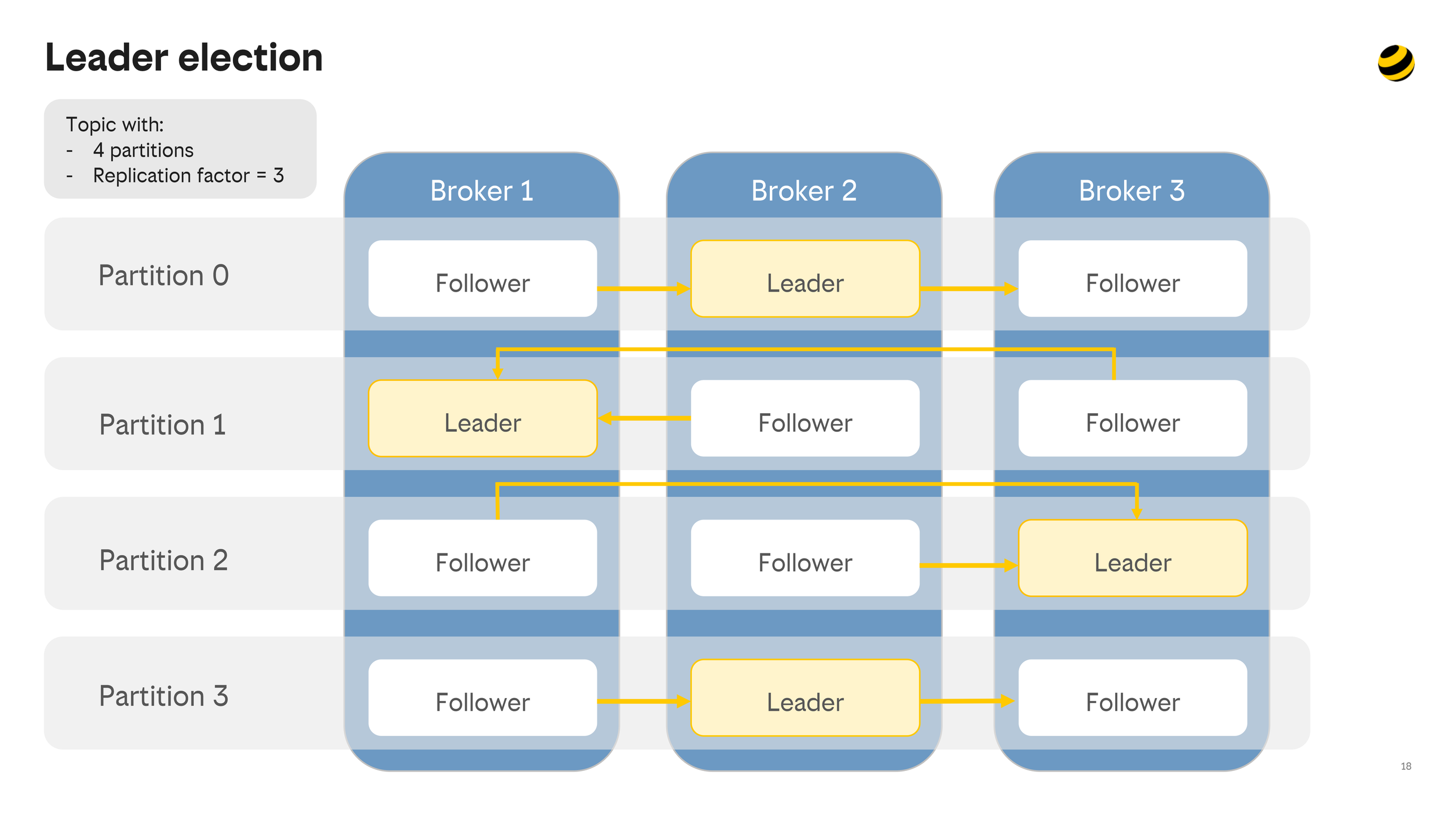
Стратегии распределения по партициям

```
producerBuilder.SetPartitioner(topic.Name, (string topicName, int partitionCount, ReadOnlySpan<byte> keyData,
bool keyIsNull) =>
{
    var keyString = System.Text.UTF8Encoding.UTF8.GetString(keyData.ToArray());
    return int.Parse(keyString.Split(" ").Last()) % partitionCount;
});
```

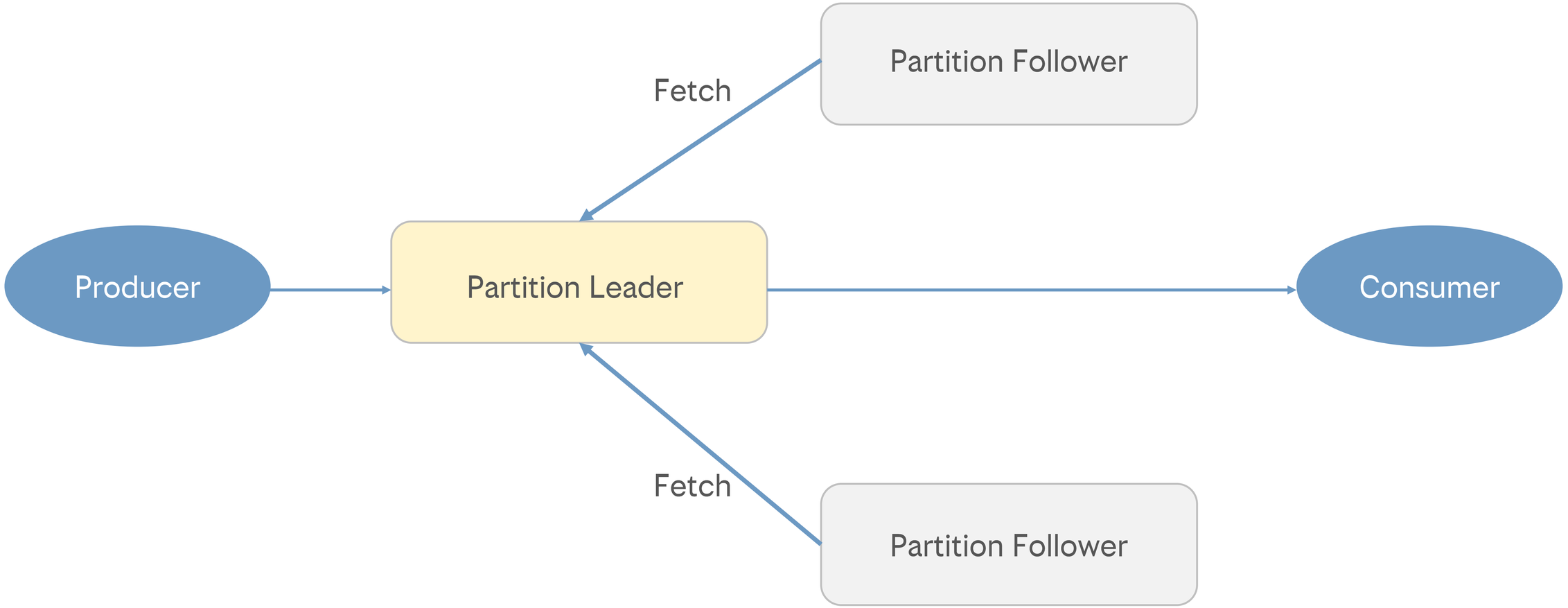
```
producerBuilder.SetDefaultPartitioner((string topicName, int partitionCount, ReadOnlySpan<byte> keyData, bool
keyIsNull) =>
{
    var keyString = System.Text.UTF8Encoding.UTF8.GetString(keyData.ToArray());
    return int.Parse(keyString.Split(" ").Last()) % partitionCount;
});
```

Используется библиотека: Confluent Kafka .Net версии 2.4.0

Leader election

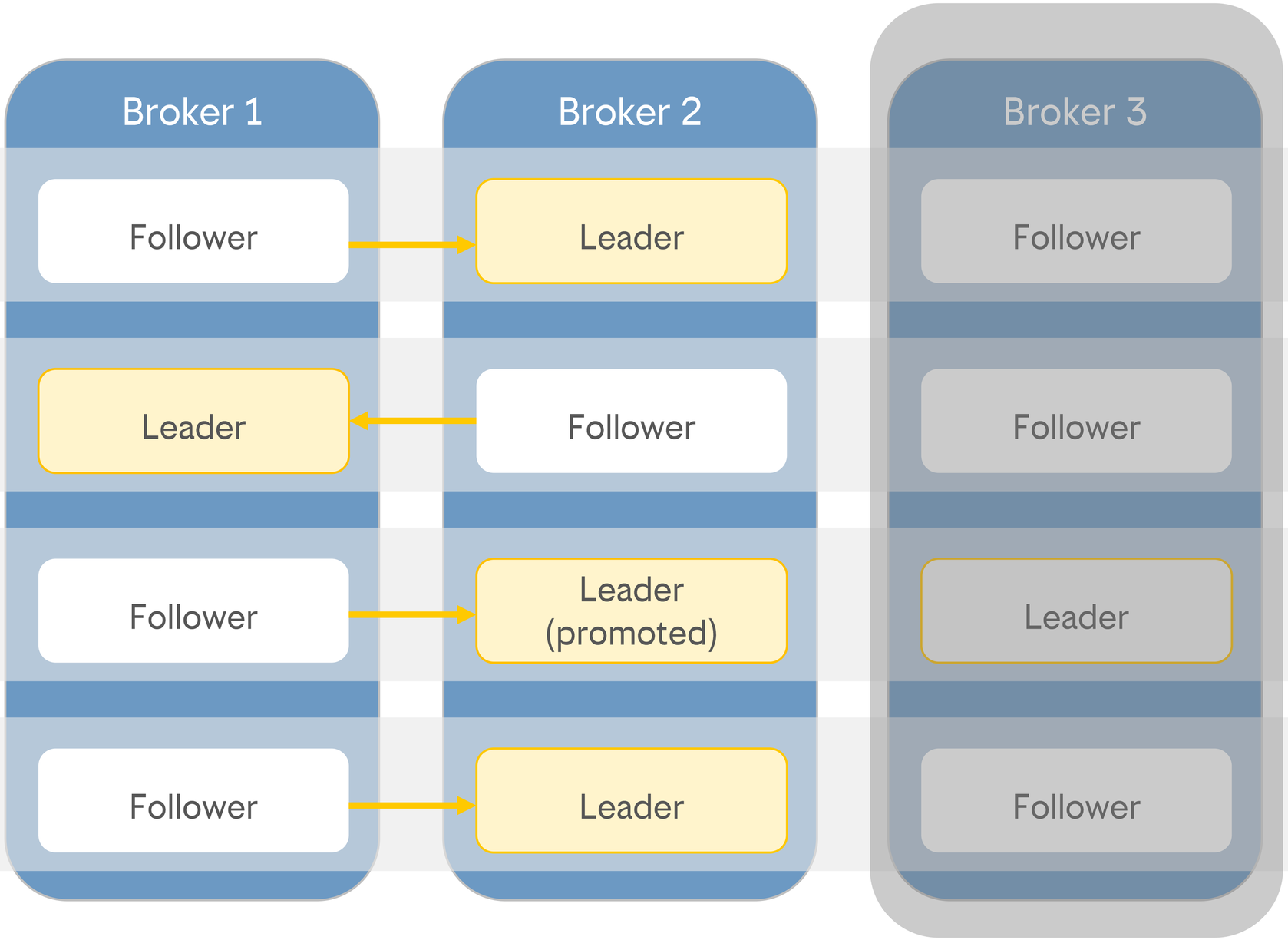


Leader election



Partition fail-over

Topic with:
- 4 partitions
- Replication factor = 3



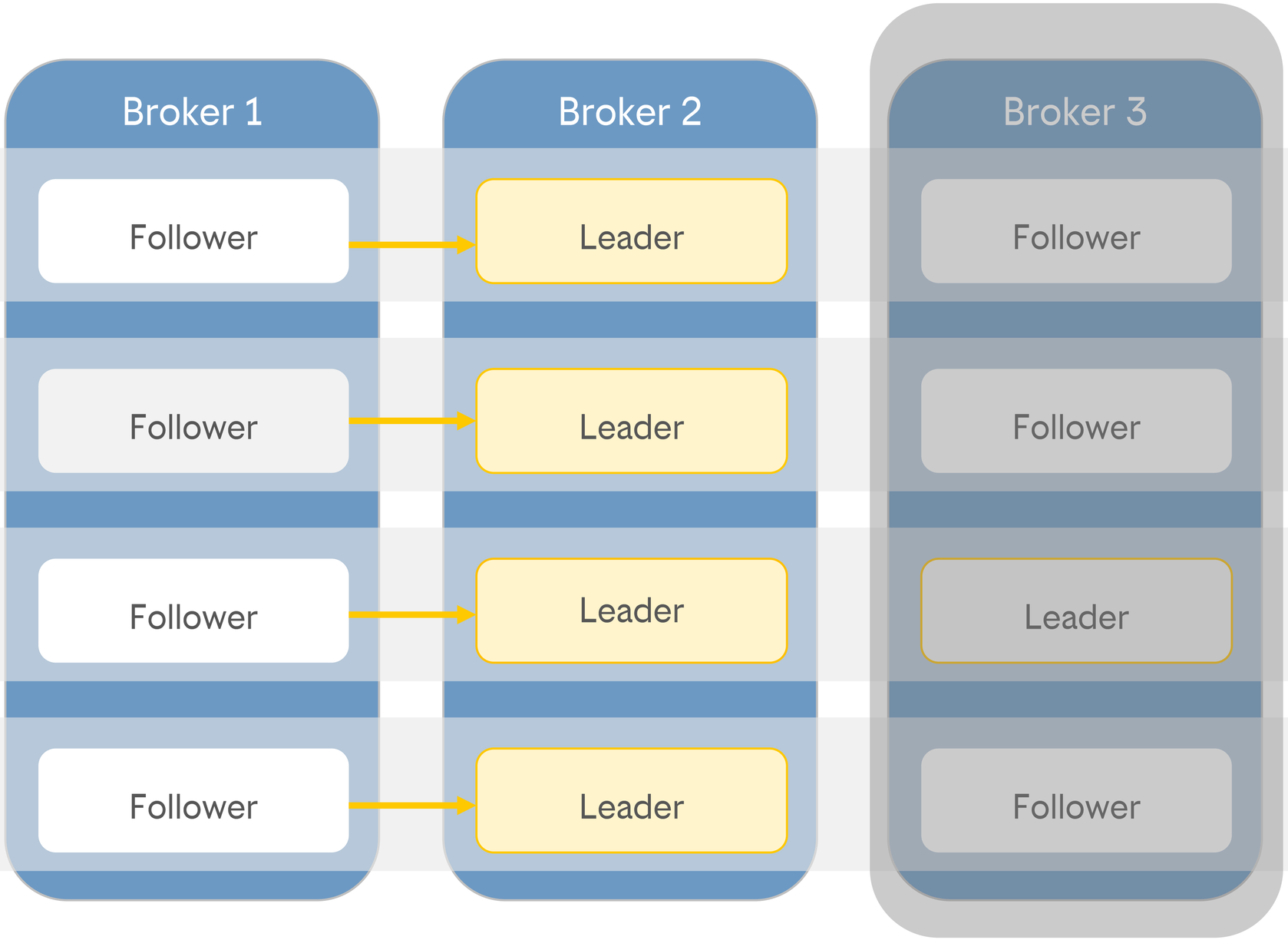
Partition fail-over

Topic with:
- 4 partitions
- Replication factor = 3

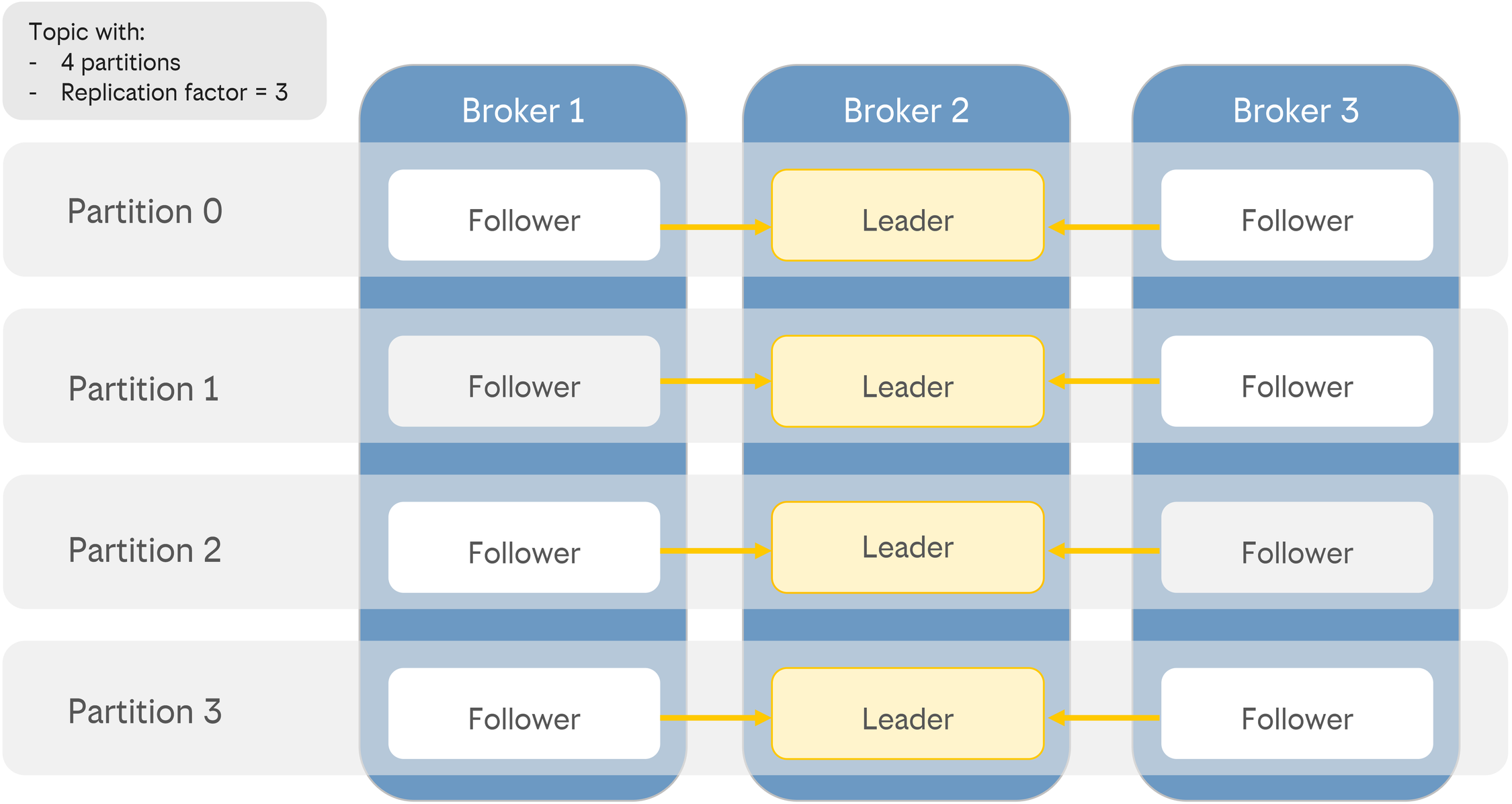
	Broker 1	Broker 2	Broker 3
Partition 0	Follower	Leader	Follower
Partition 1	Leader	Leader (promoted)	Follower
Partition 2	Follower	Leader	Leader
Partition 3	Follower	Leader	Follower

Partition fail-over

Topic with:
- 4 partitions
- Replication factor = 3



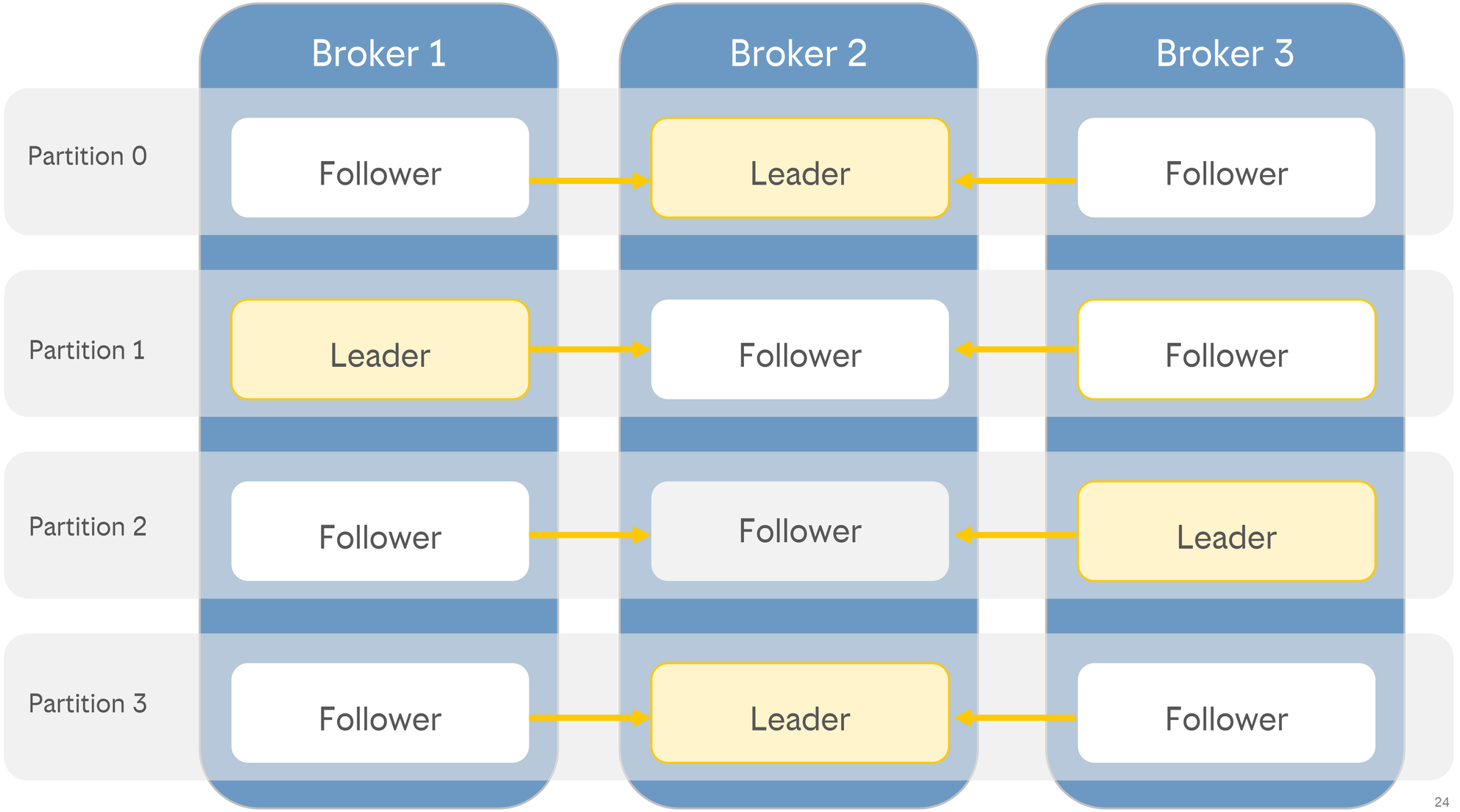
Partition fail-over



Partition fail-over

Partition Rebalancing

auto.leader.rebalance .enable=true



Retention

Old segment will be deleted based on time or space rules

New data is written to the active segment

Retention Policy

Retention

retention.bytes - объем хранимых событий этого топика (в байтах)

retention.ms – длительность (в миллисекундах) хранения событий данного топика

-1 – если не ограничено

значения по умолчанию: bytes – не органичено (-1), ms - 604800000 (7 days)

Retention

retention.bytes - объем хранимых событий этого топика (в байтах)

retention.ms – длительность (в миллисекундах) хранения событий данного топика

-1 – если не ограничено

значения по умолчанию: bytes – не органичено (-1), ms - 604800000 (7 days)

Retention

retention.bytes - объем хранимых событий этого топика (в байтах)

retention.ms – длительность (в миллисекундах) хранения событий данного топика

-1 – если не ограничено

значения по умолчанию: bytes – не органичено (-1), ms - 604800000 (7 days)

Retention

retention.bytes - объем хранимых событий этого топика (в байтах)

retention.ms – длительность (в миллисекундах) хранения событий данного топика

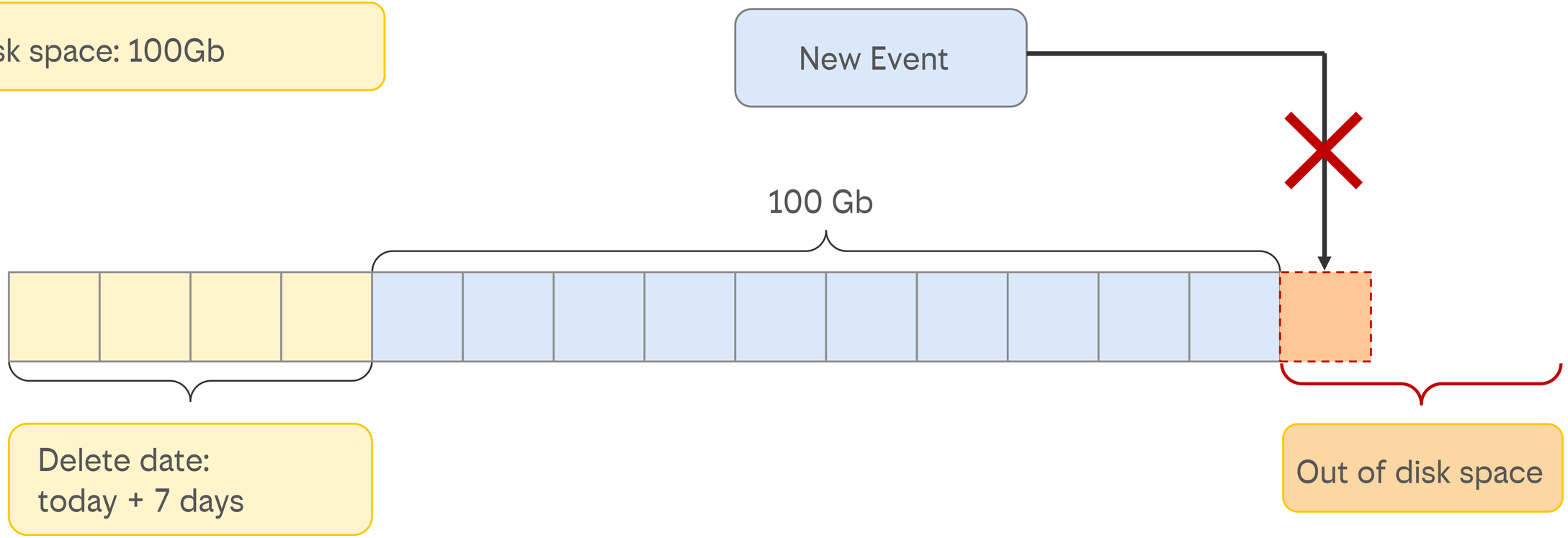
-1 – если не ограничено

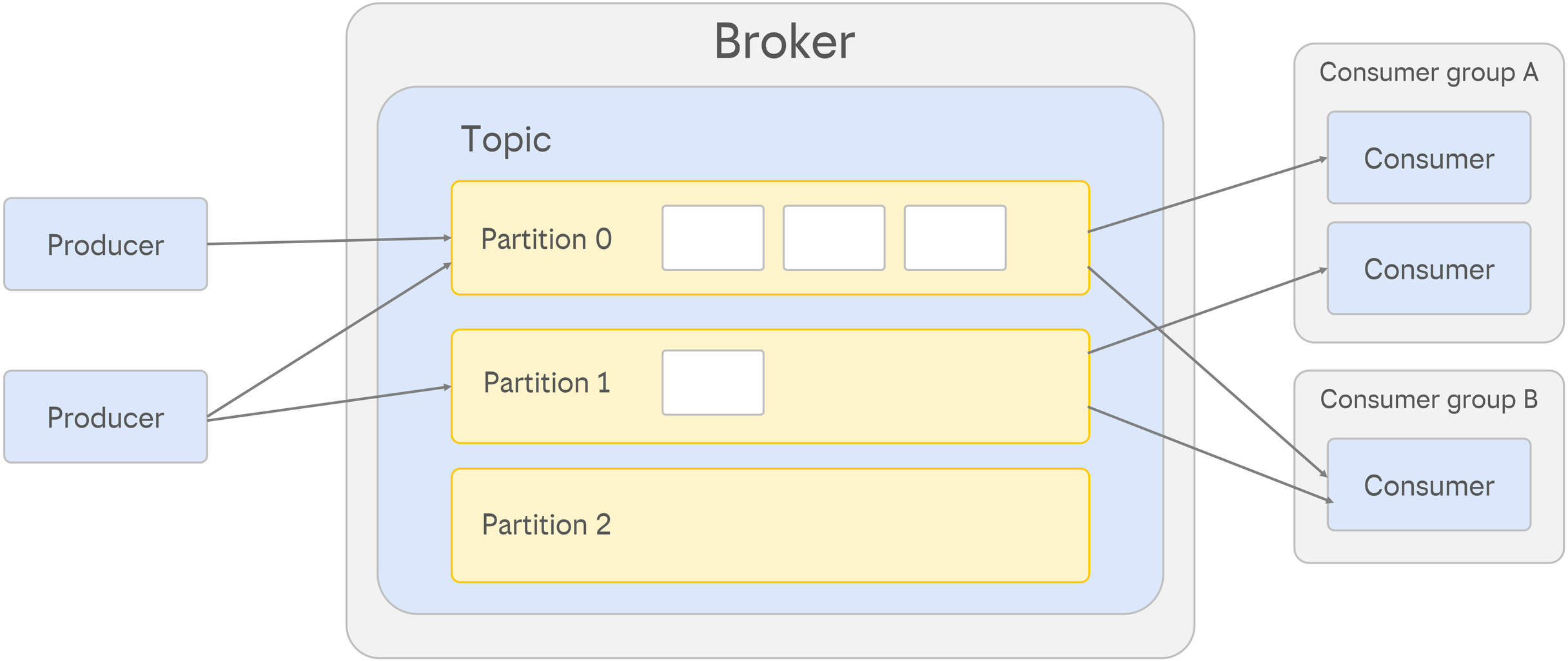
значения по умолчанию: bytes – не органичено (-1), ms - 604800000 (7 days)

Retention

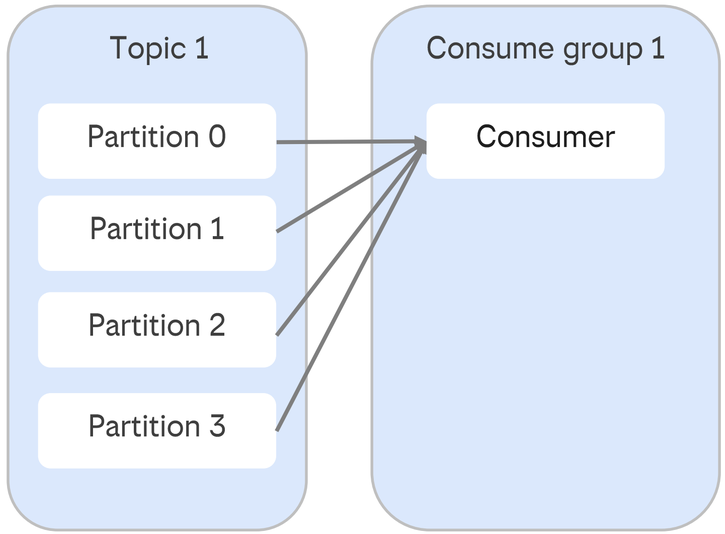
retention.bytes: -1
retention.ms: 604800000

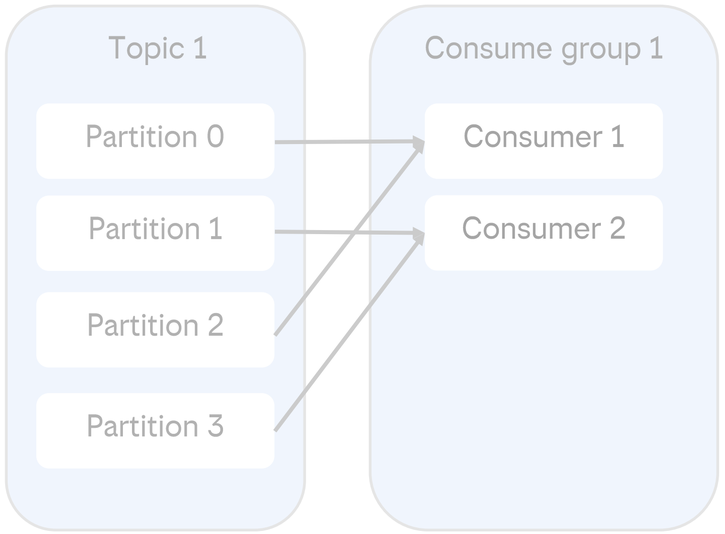
Disk space: 100Gb

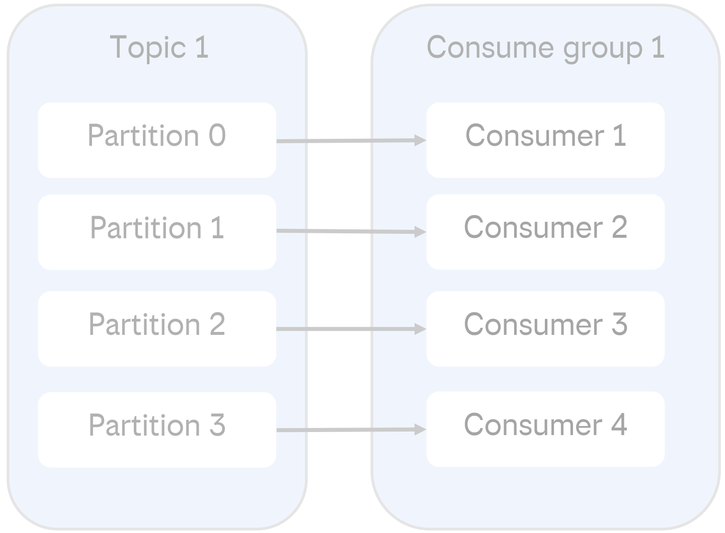


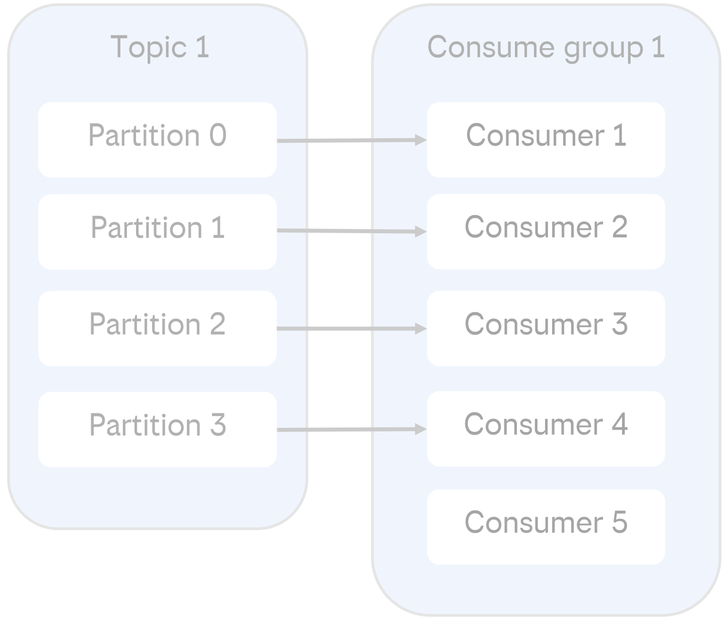

Broker
Topic
Partition 0
Partition 1
Partition 2
Producer
Producer
Consumer group A
Consumer
Consumer
Consumer group B
Consumer

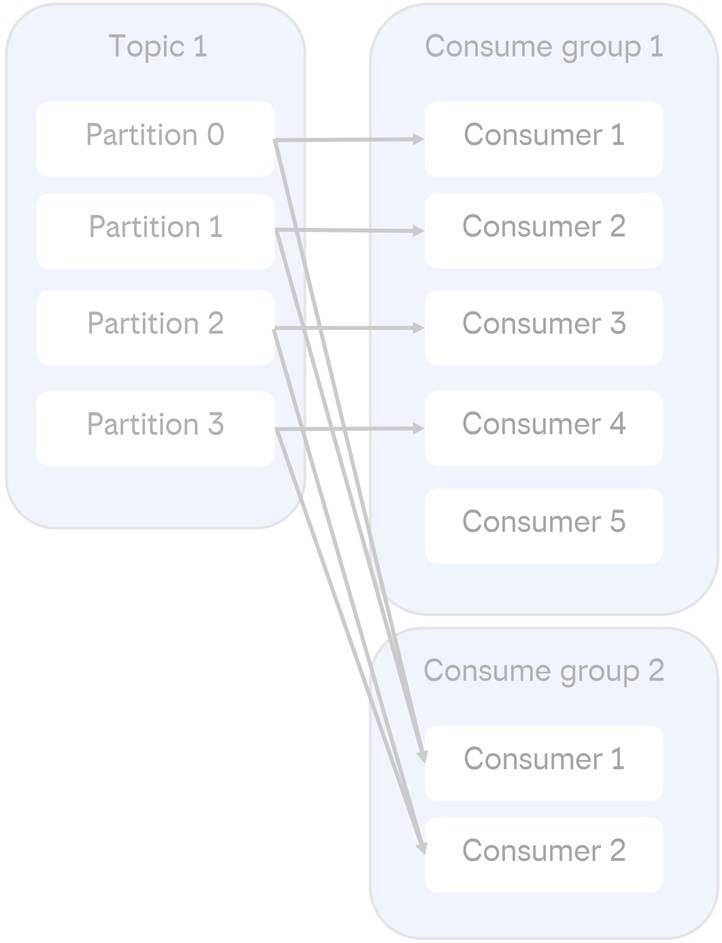
Группы потребителей



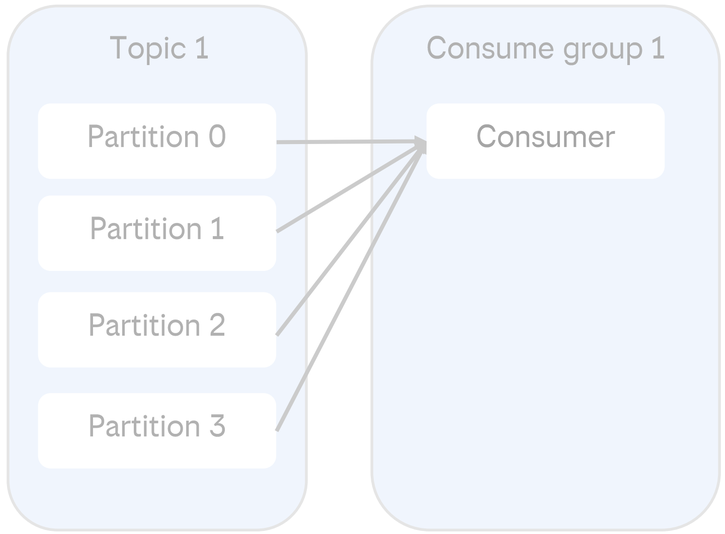


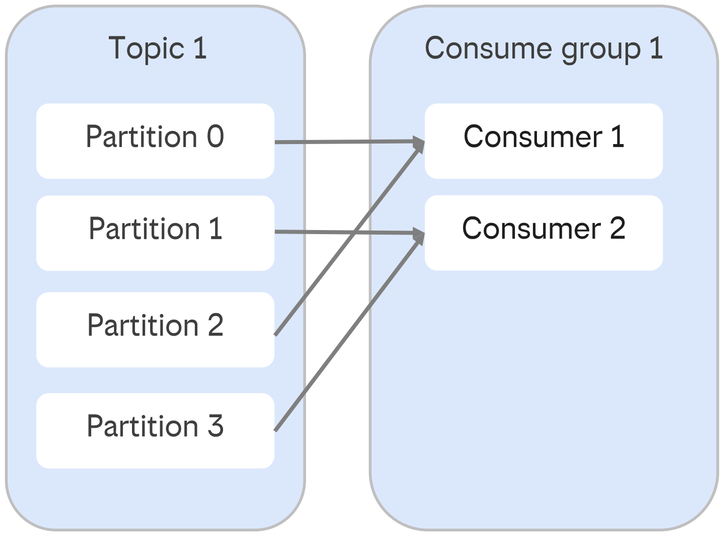


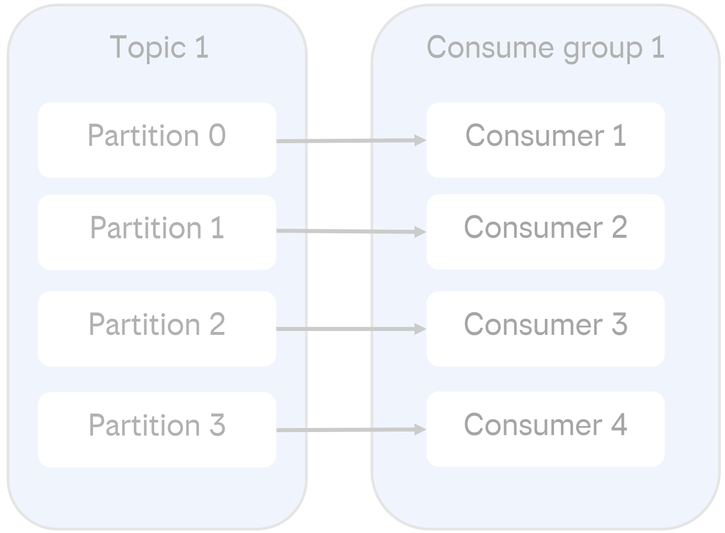


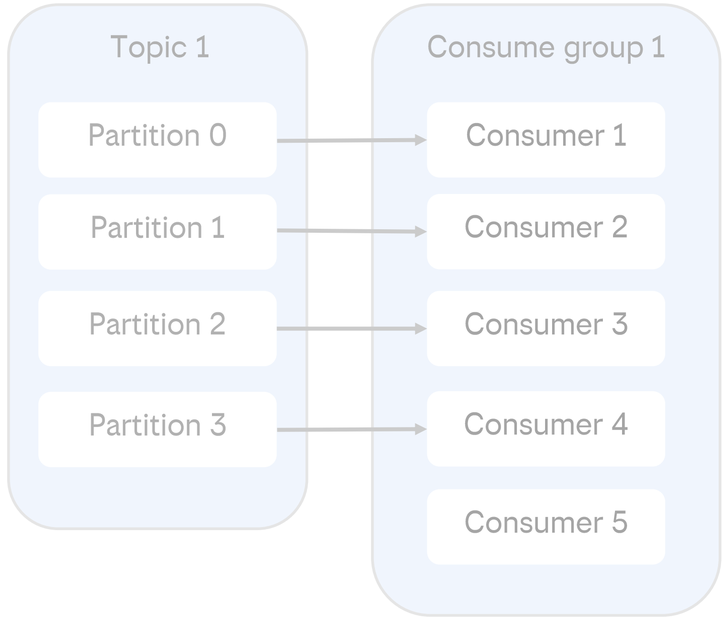


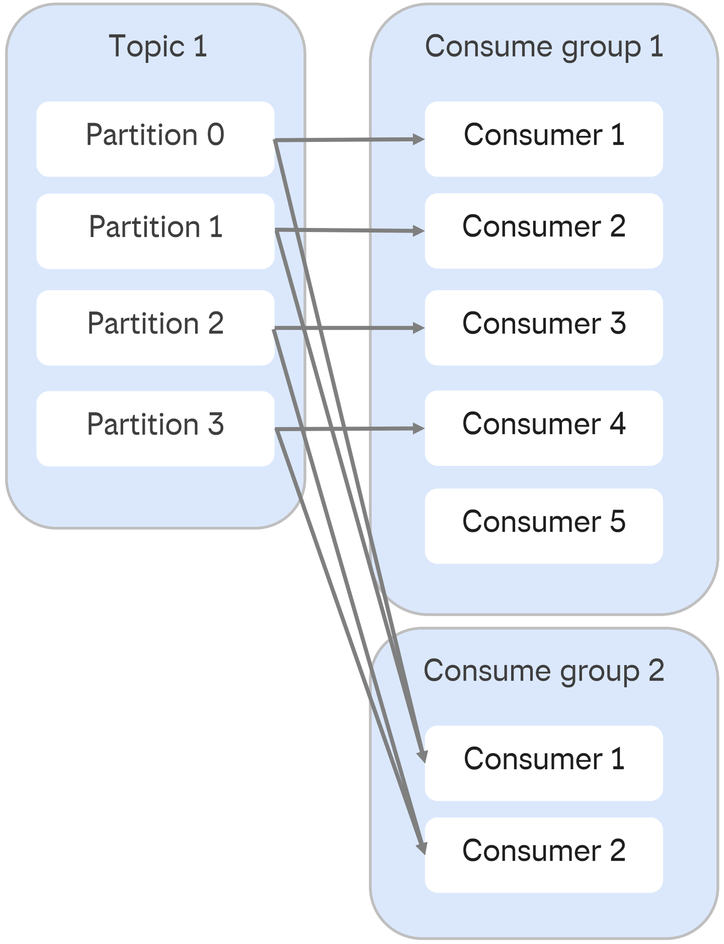
Группы потребителей



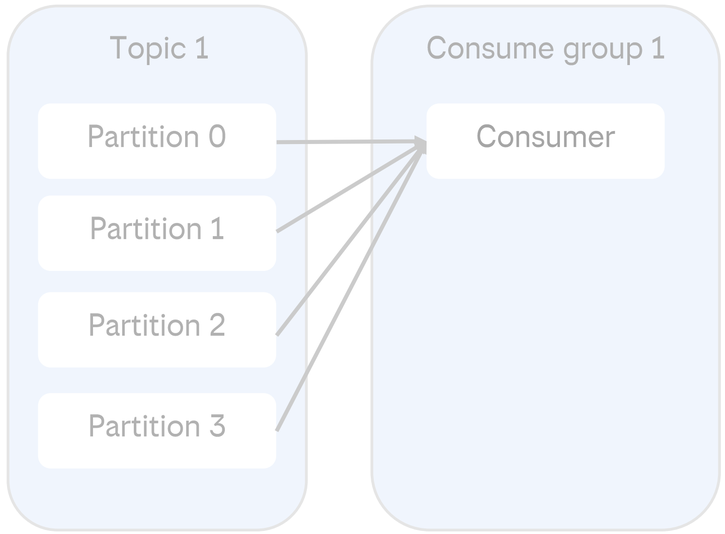


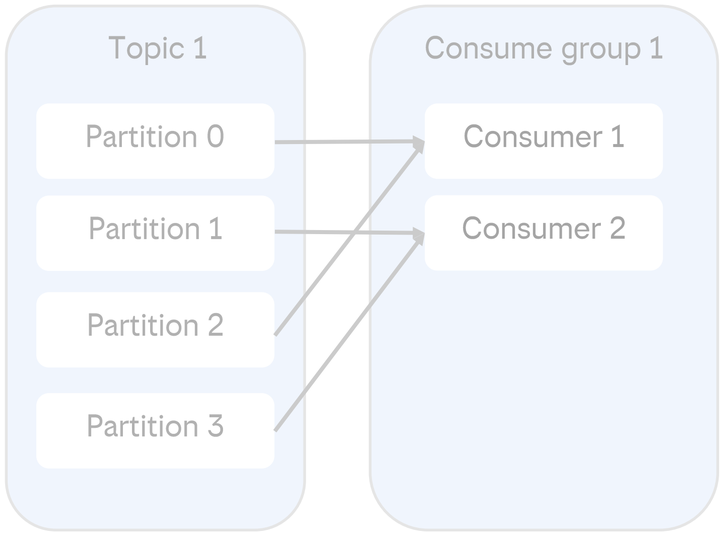


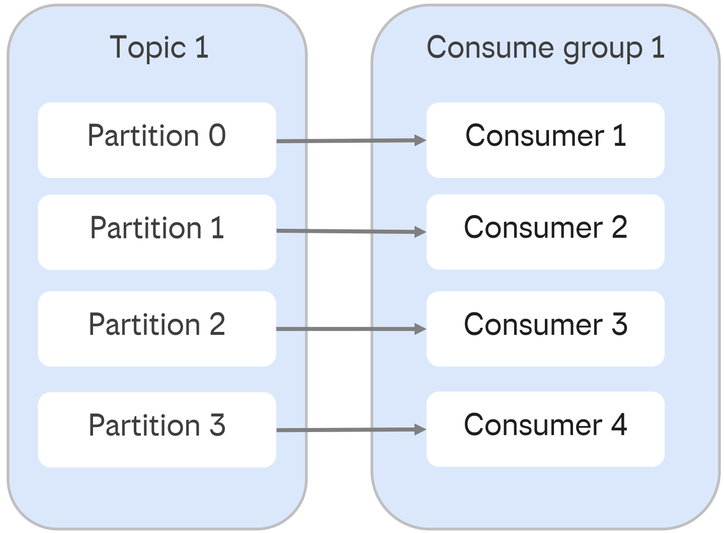


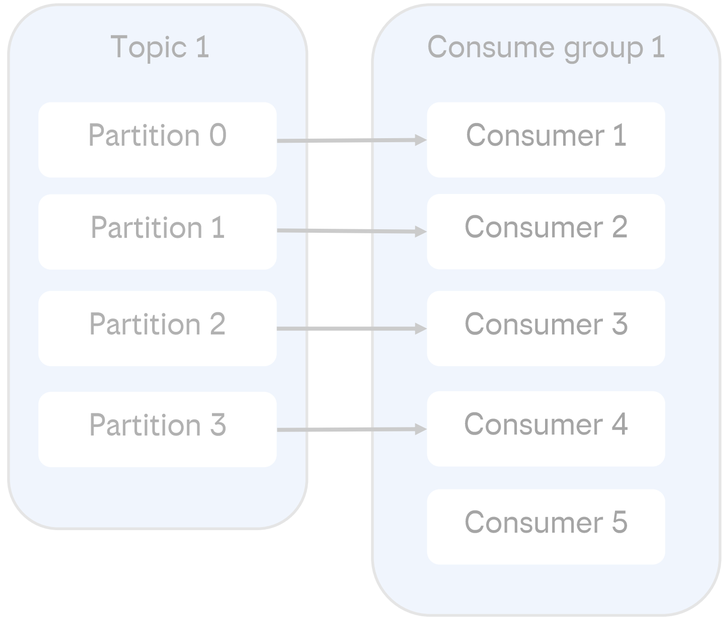


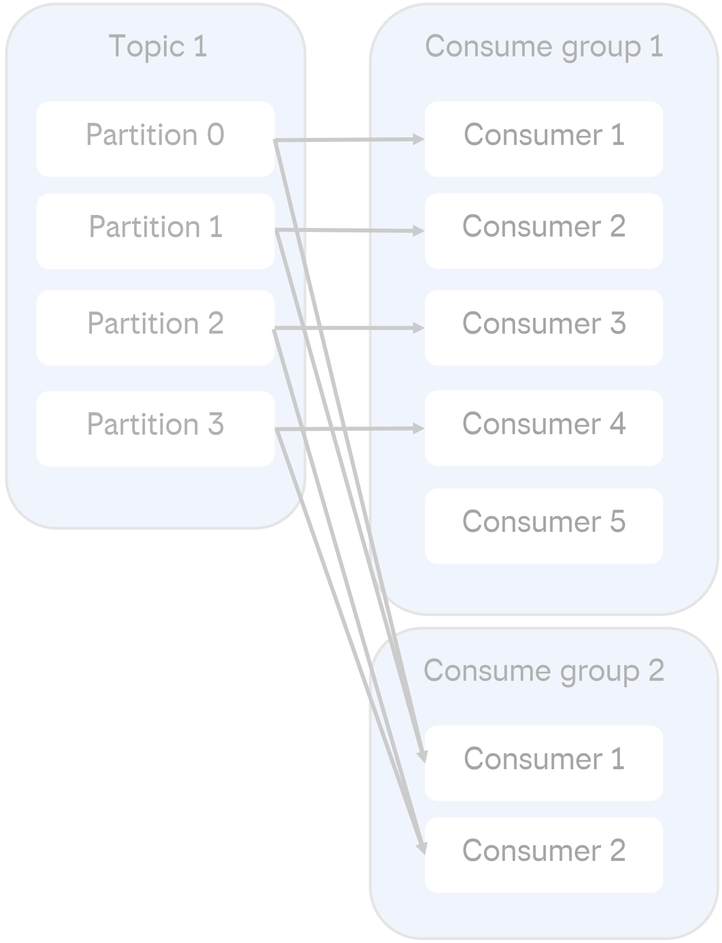
Группы потребителей



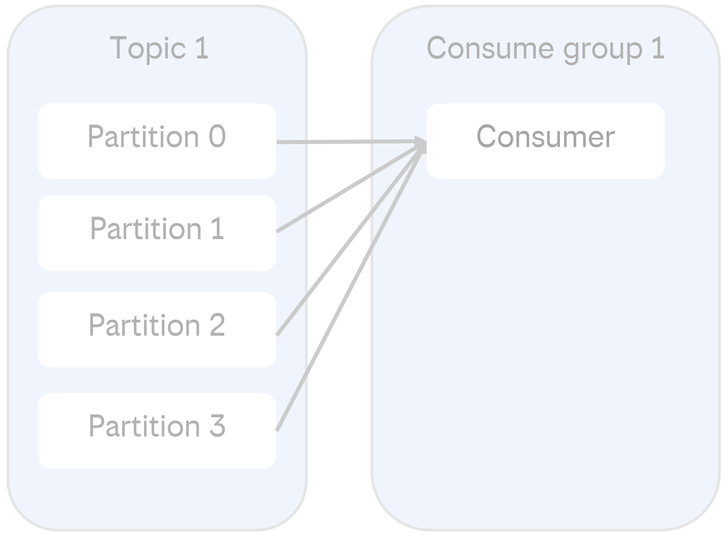


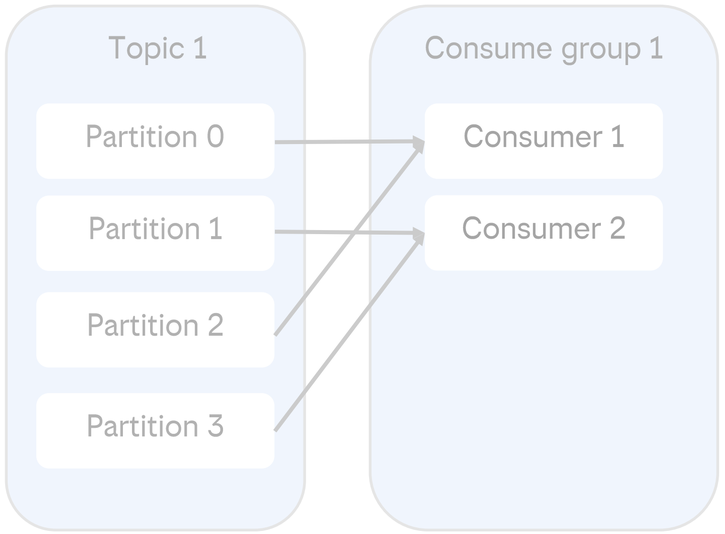


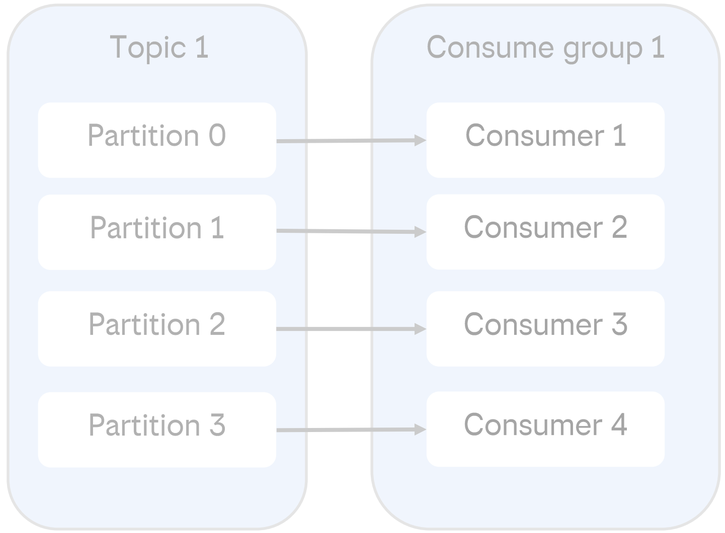


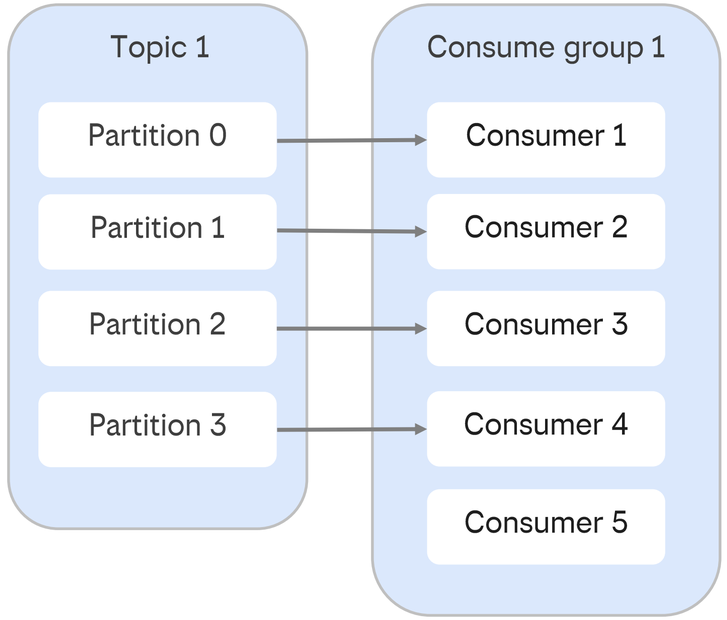


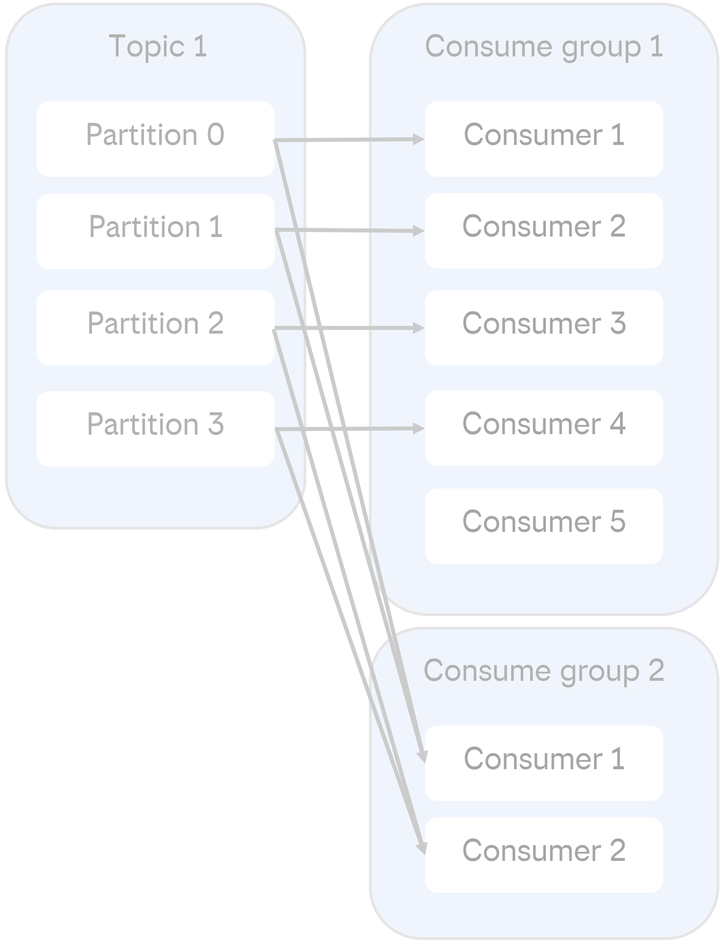
Группы потребителей



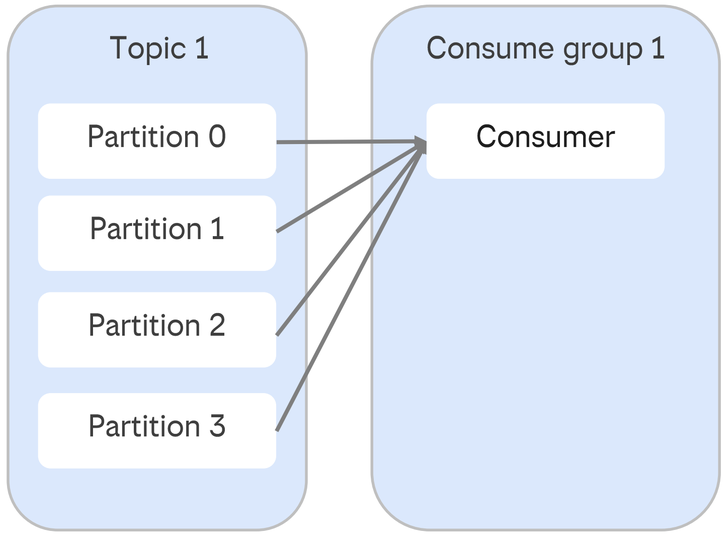


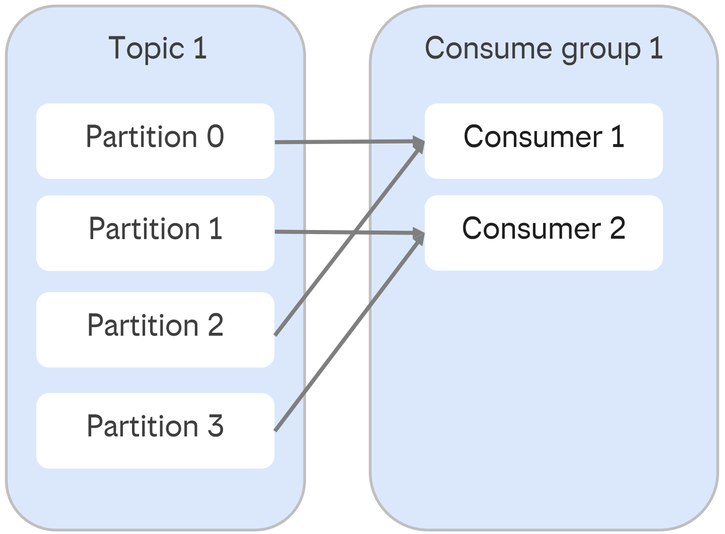


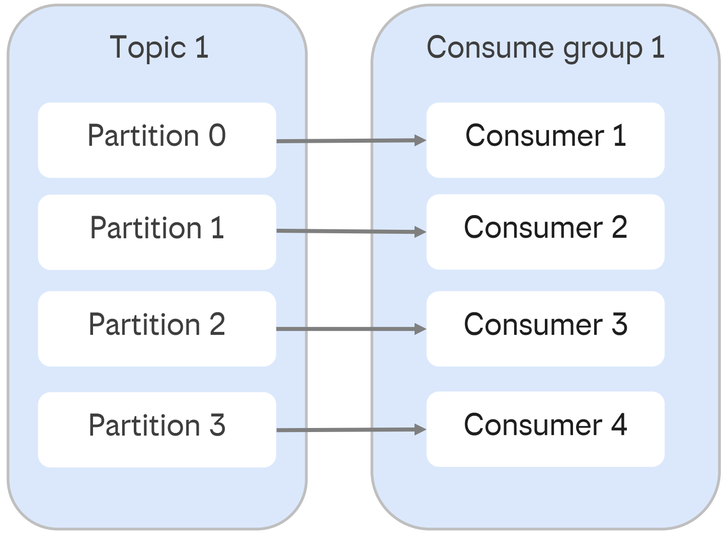


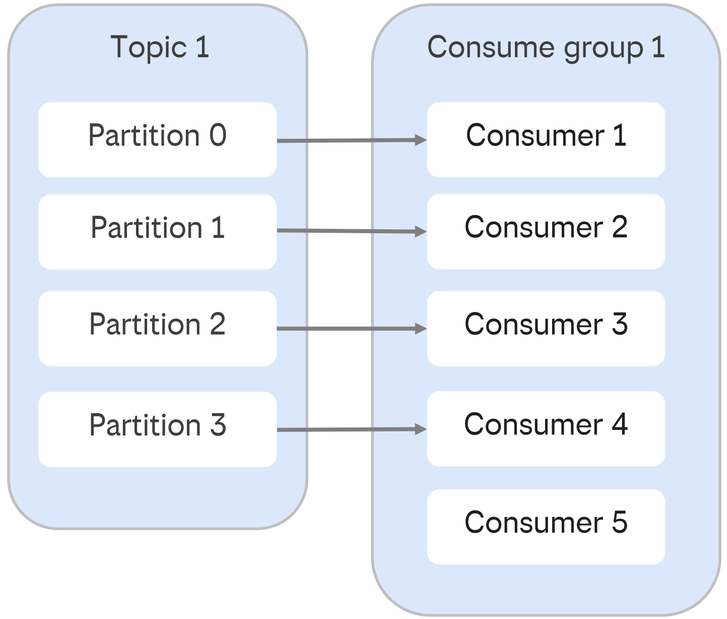


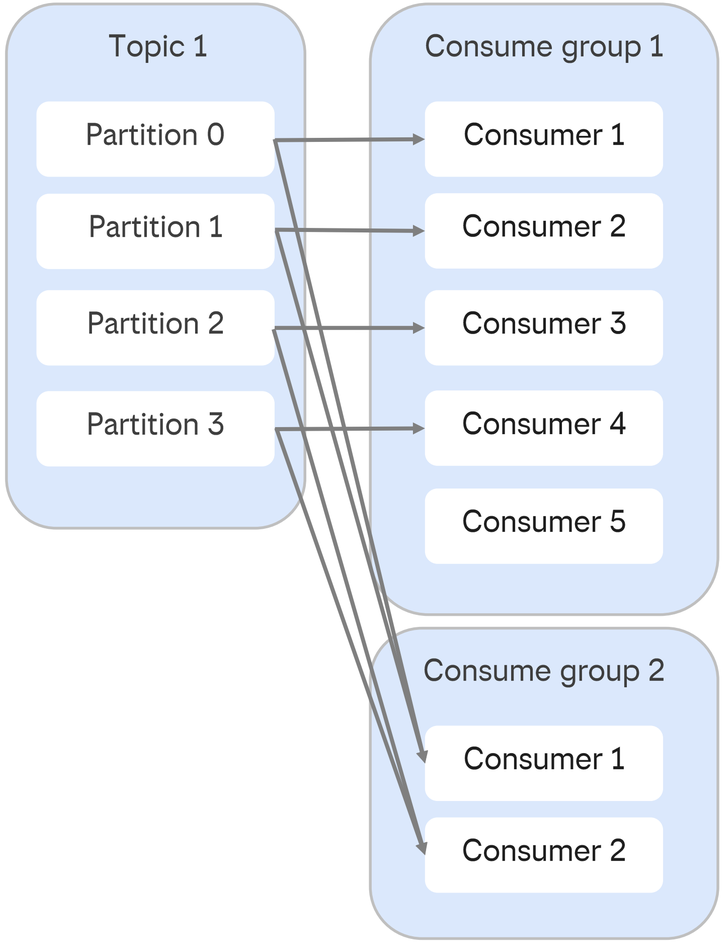
Группы потребителей

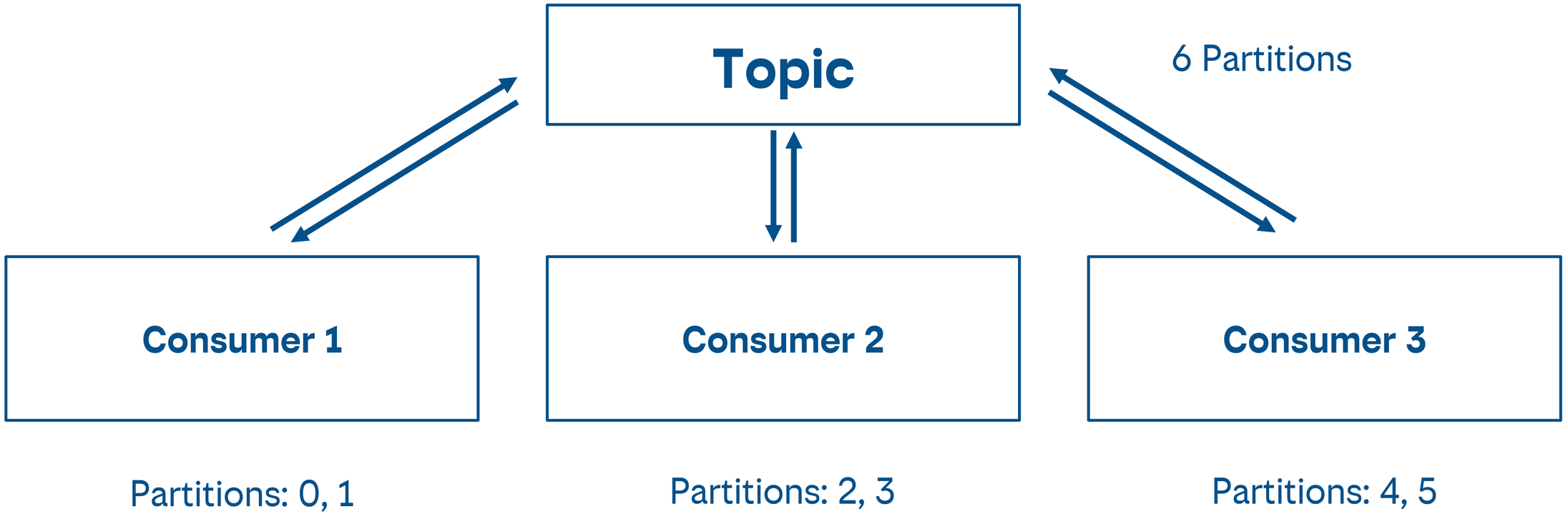


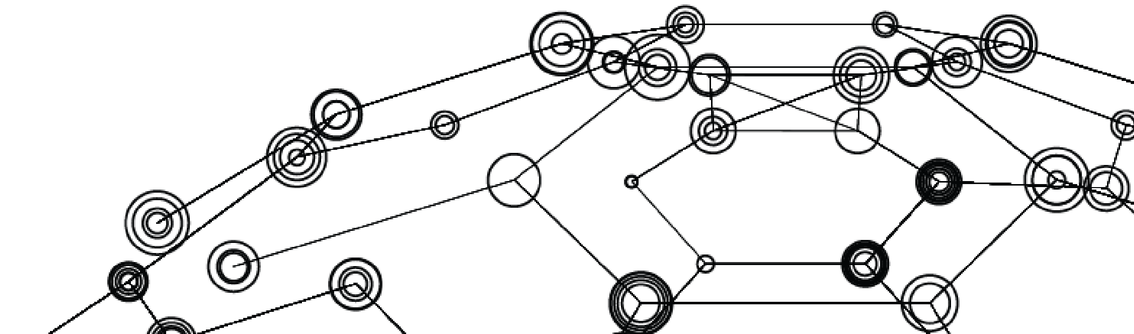


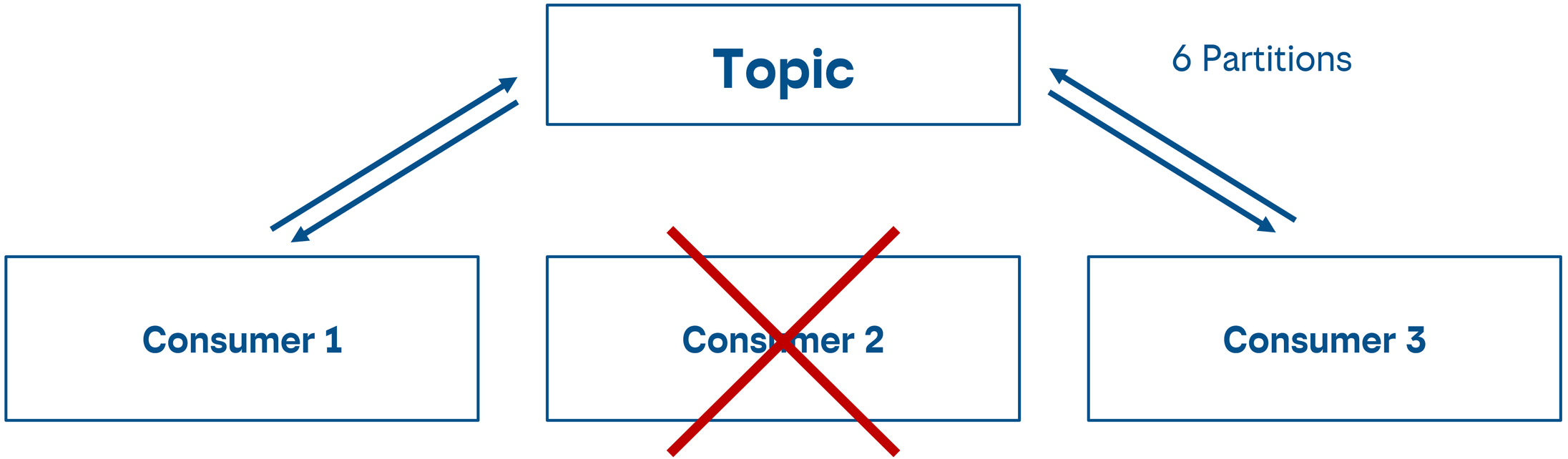


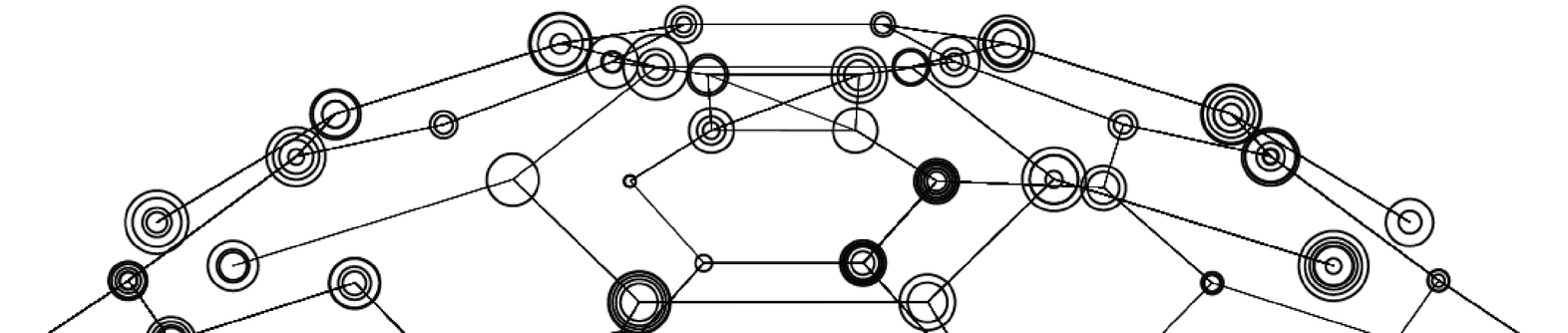




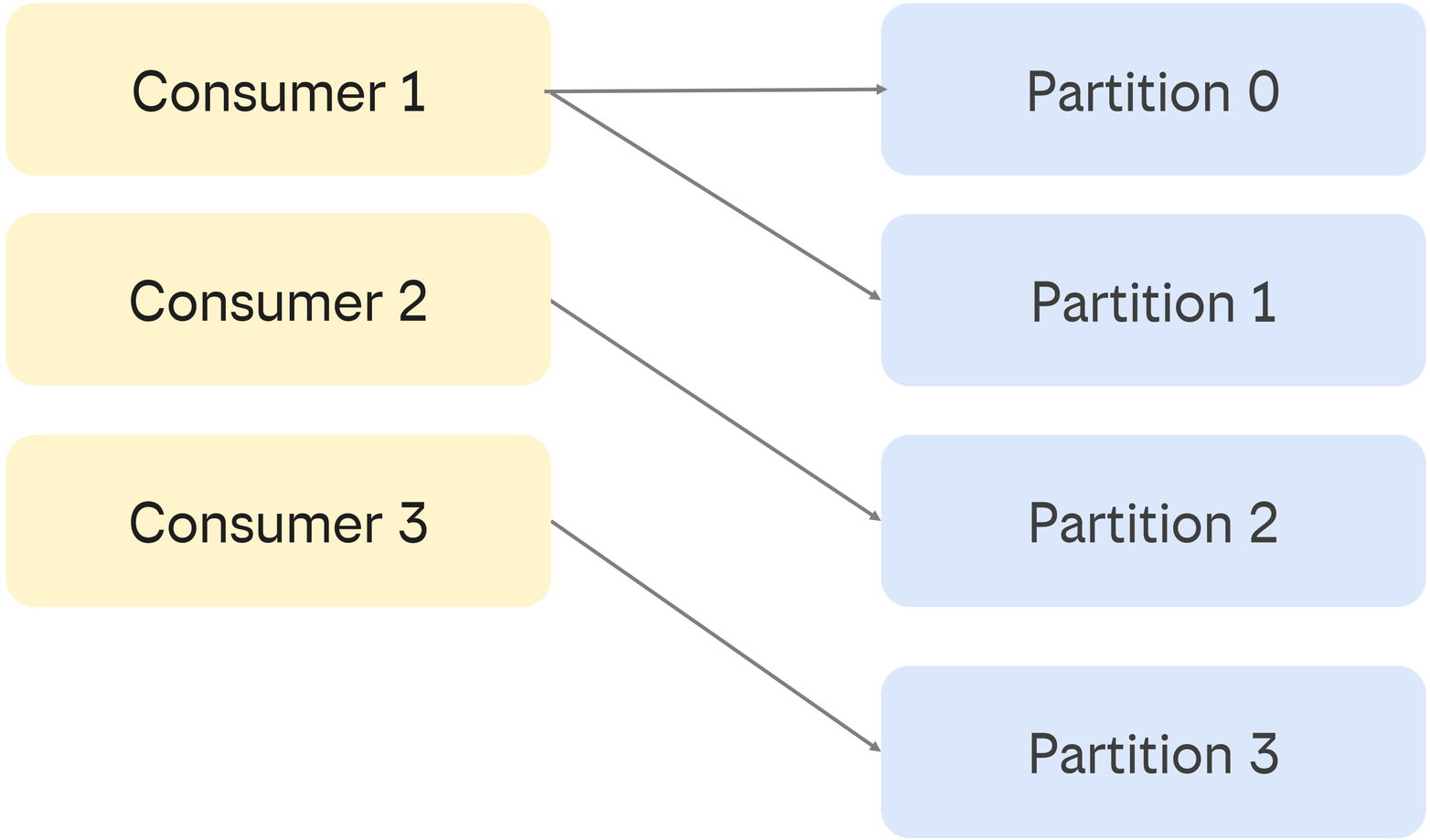

Topic
6 Partitions
Consumer 1
Consumer 2
Consumer 3
Partitions: 0, 1
Partitions: 2, 3
Partitions: 4, 5




Topic
6 Partitions
Consumer 1
Consumer 2
Consumer 3
Partitions: 0, 1, 3
Partitions: 2, 4, 5

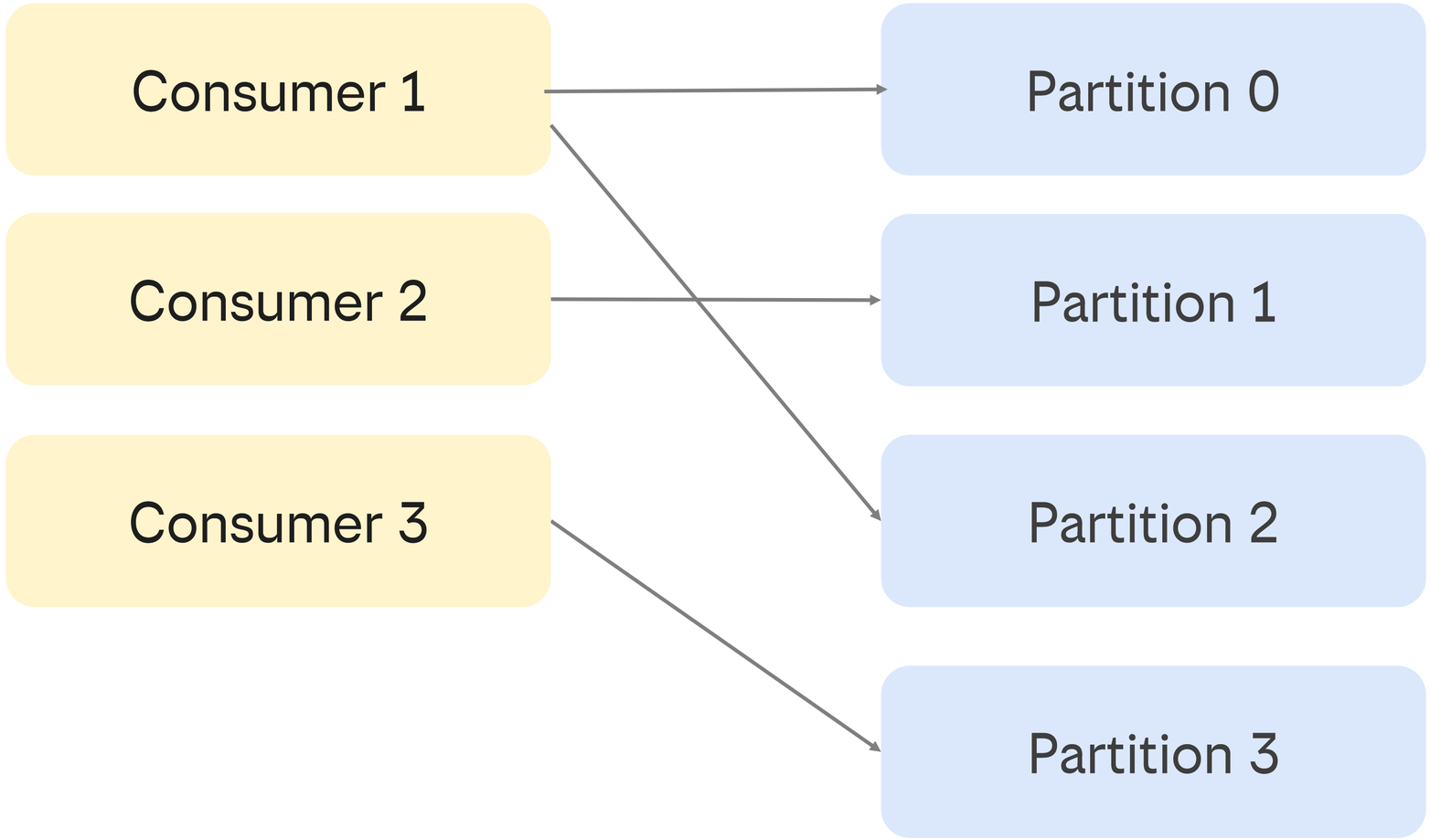


Range assignor



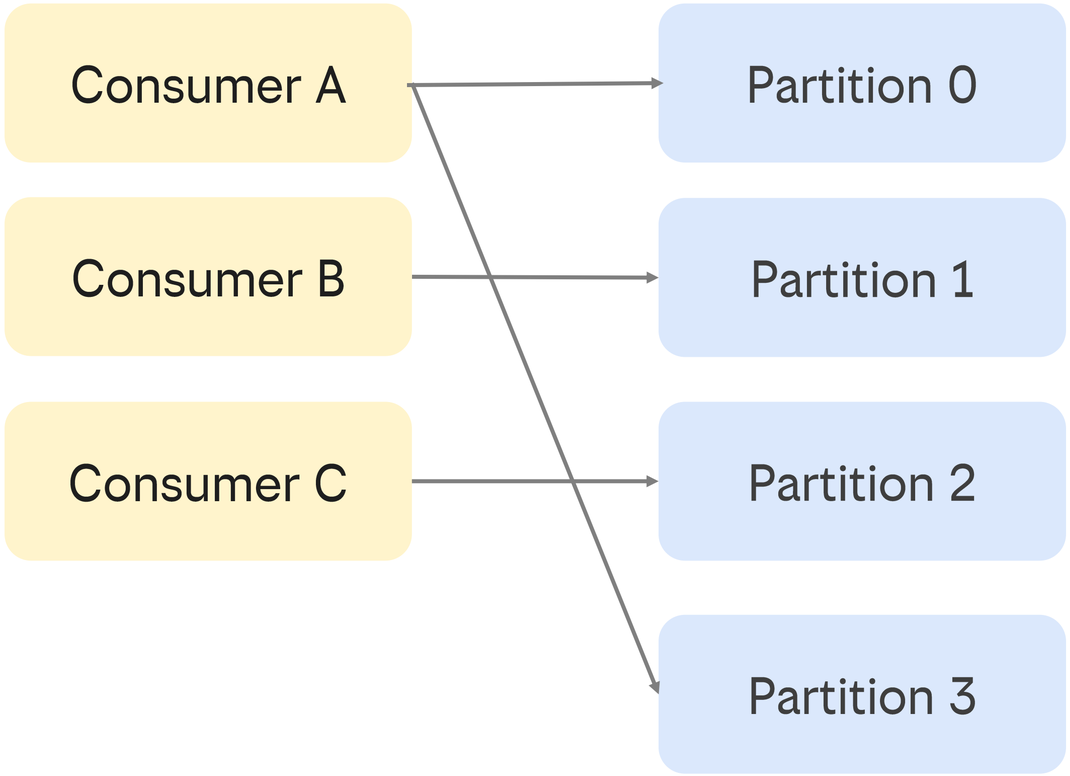
partition.assignment.strategy = range

Round Robin Assignor

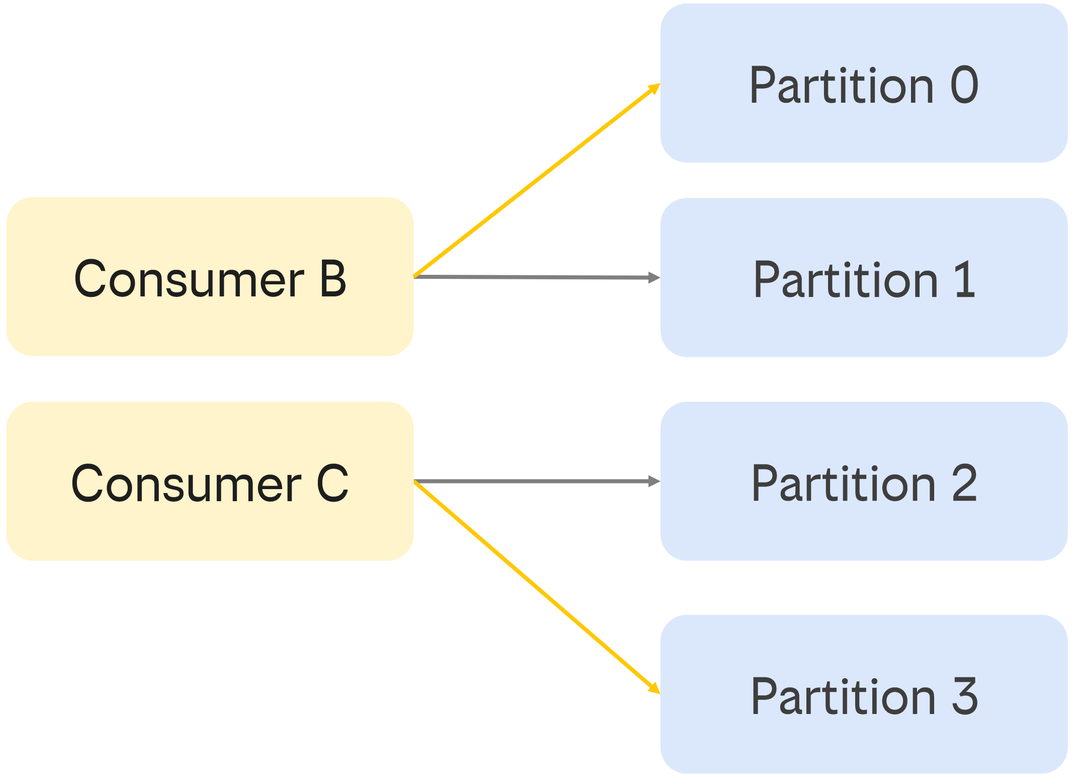


partition.assignment.strategy = roundrobin

Sticky Assignor & Cooperative Sticky Assignor



до ребалансировки

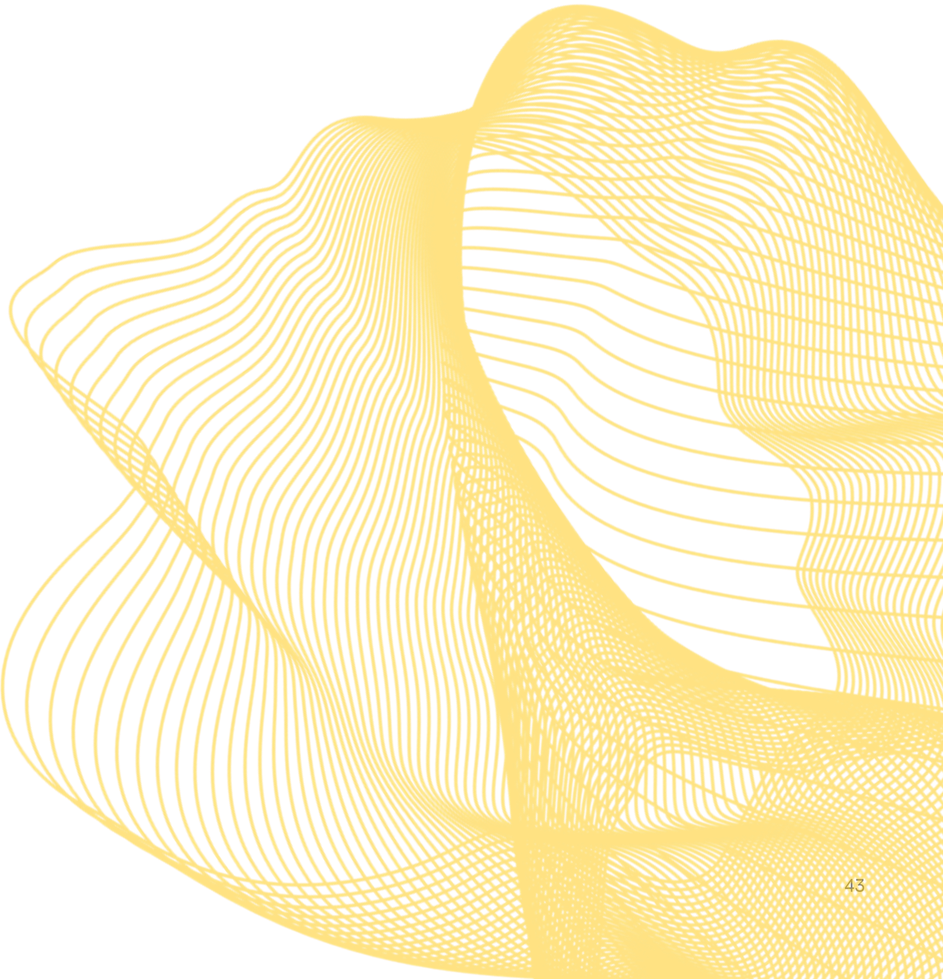


после ребалансировки

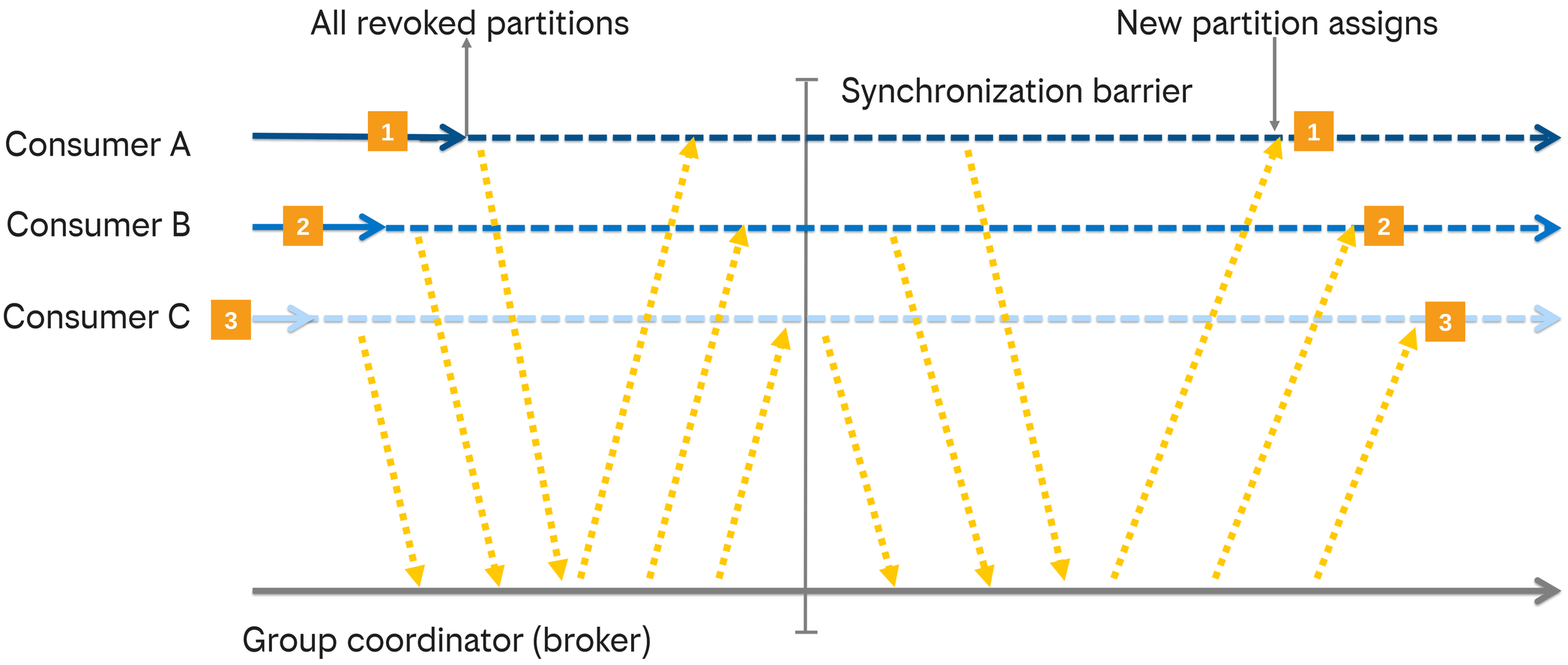
partition.assignment.strategy = cooperative-sticky

Ребалансировка

- **Безотлагательная**
- **Совместная**



Безотлагательная ребалансировка



Совместная ребалансировка

Совместная ребалансировка приостанавливает потребление лишь для подмножества партиций, которые будут переназначены

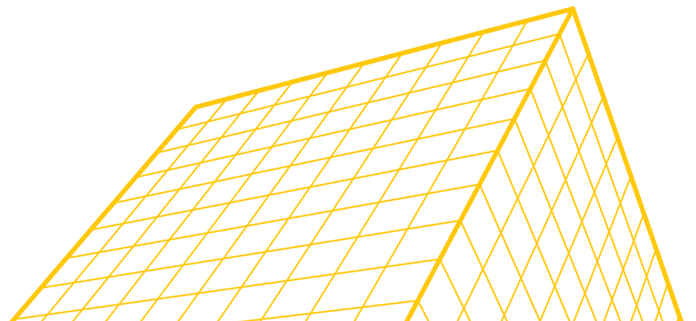
Параметры консюмеров

heartbeat.interval.ms - интервал отправки контрольных сигнаглов консюмера (по умолчанию - 3 сек)

session.timeout.ms - окно на отправку контрольных сигналов (по умолчанию - 10 сек)

session.timeout.ms > heartbeat.interval.ms, - обычно соотношение 1 к 3

max.poll.interval.ms – максимальное время на обработку события перед получением следующего (по умолчанию - 5 мин)



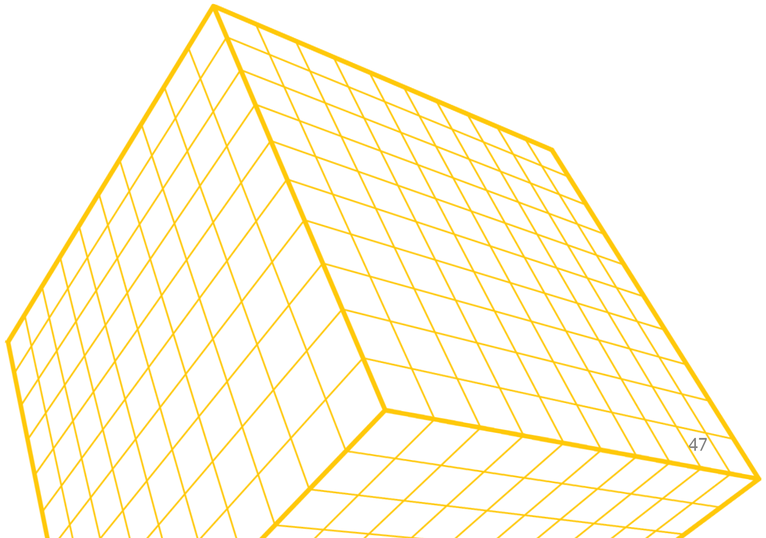
Параметры консюмеров

heartbeat.interval.ms - интервал отправки контрольных сигнаглов консюмера (по умолчанию - 3 сек)

session.timeout.ms - окно на отправку контрольных сигналов (по умолчанию - 10 сек)

session.timeout.ms > heartbeat.interval.ms, - обычно соотношение 1 к 3

max.poll.interval.ms – максимальное время на обработку события перед получением следующего (по умолчанию - 5 мин)



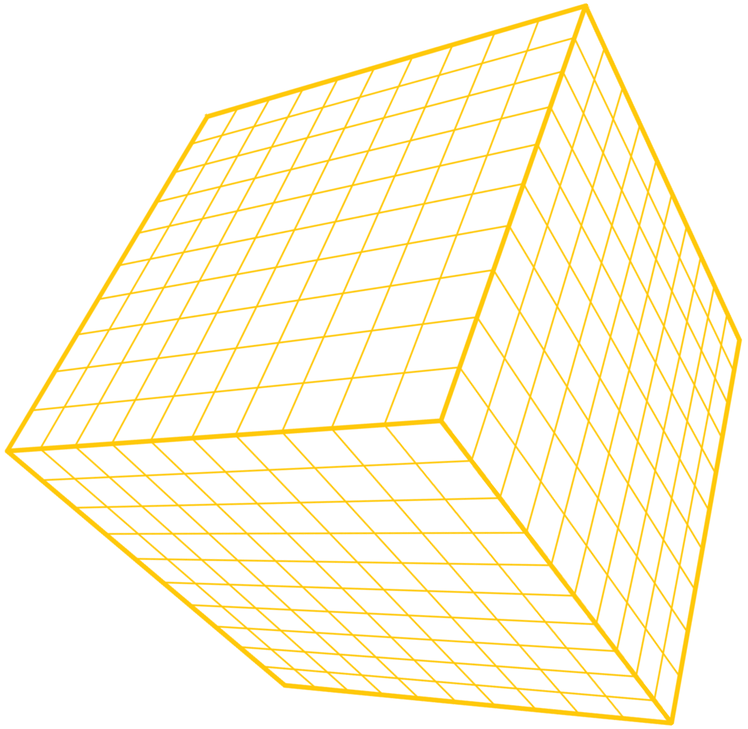
Параметры консюмеров

heartbeat.interval.ms - интервал отправки контрольных сигнаглов консюмера (по умолчанию - 3 сек)

session.timeout.ms - окно на отправку контрольных сигналов (по умолчанию - 10 сек)

session.timeout.ms > heartbeat.interval.ms, - обычно соотношение 1 к 3

max.poll.interval.ms – максимальное время на обработку события перед получением следующего (по умолчанию - 5 мин)



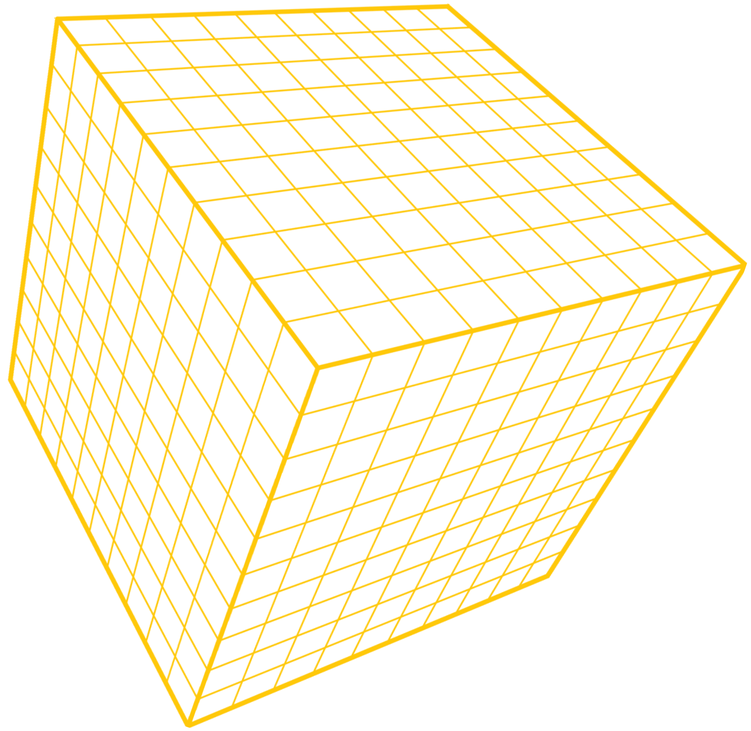
Параметры консюмеров

heartbeat.interval.ms - интервал отправки контрольных сигнаглов консюмера (по умолчанию - 3 сек)

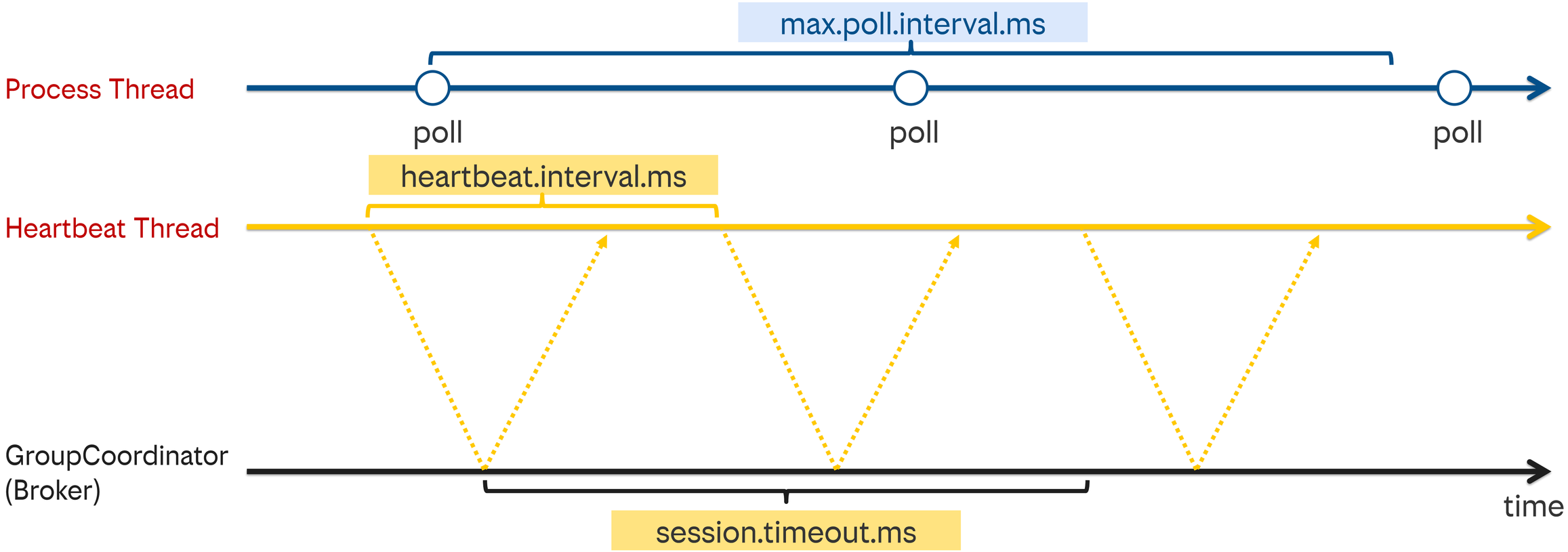
session.timeout.ms - окно на отправку контрольных сигналов (по умолчанию - 10 сек)

session.timeout.ms > heartbeat.interval.ms, - обычно соотношение 1 к 3

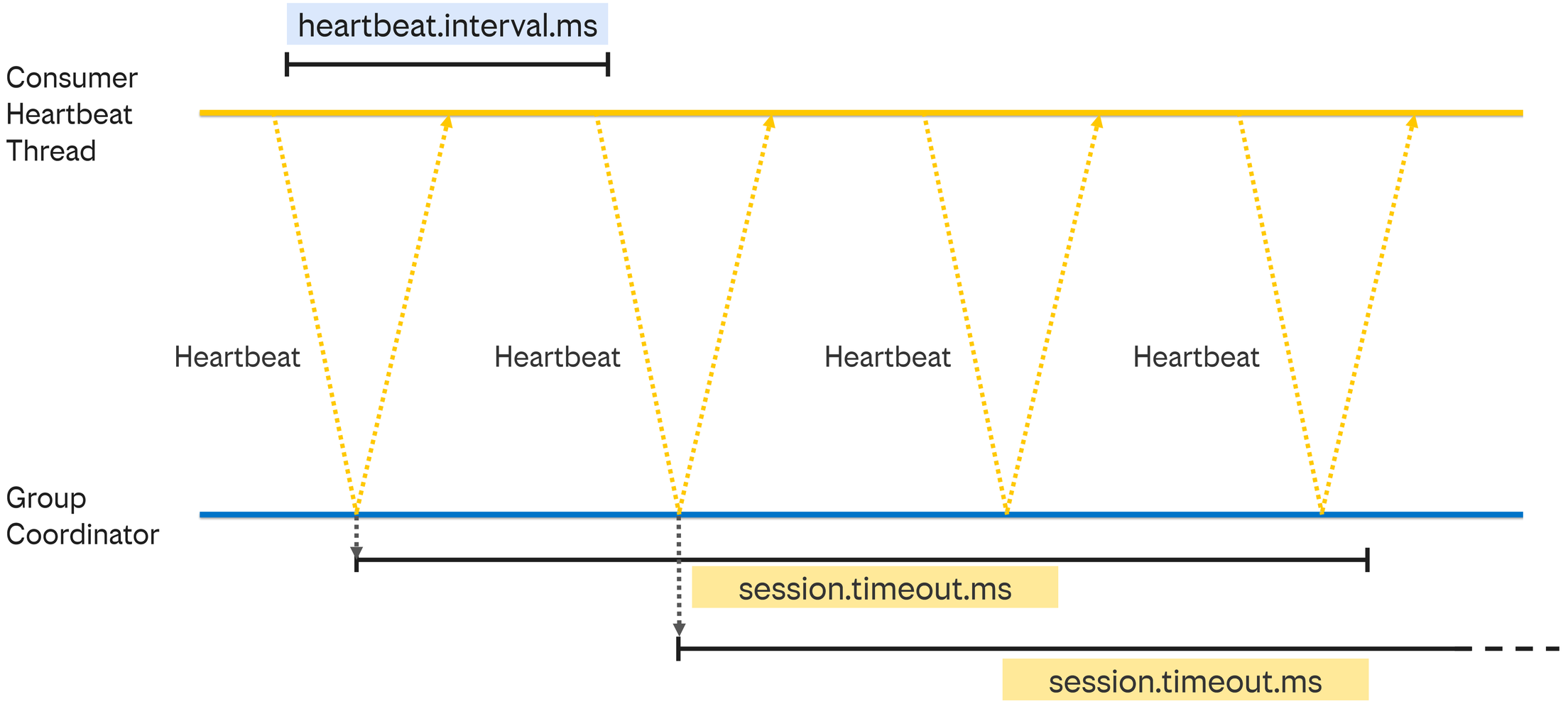
max.poll.interval.ms – максимальное время на обработку события перед получением следующего (по умолчанию - 5 мин)



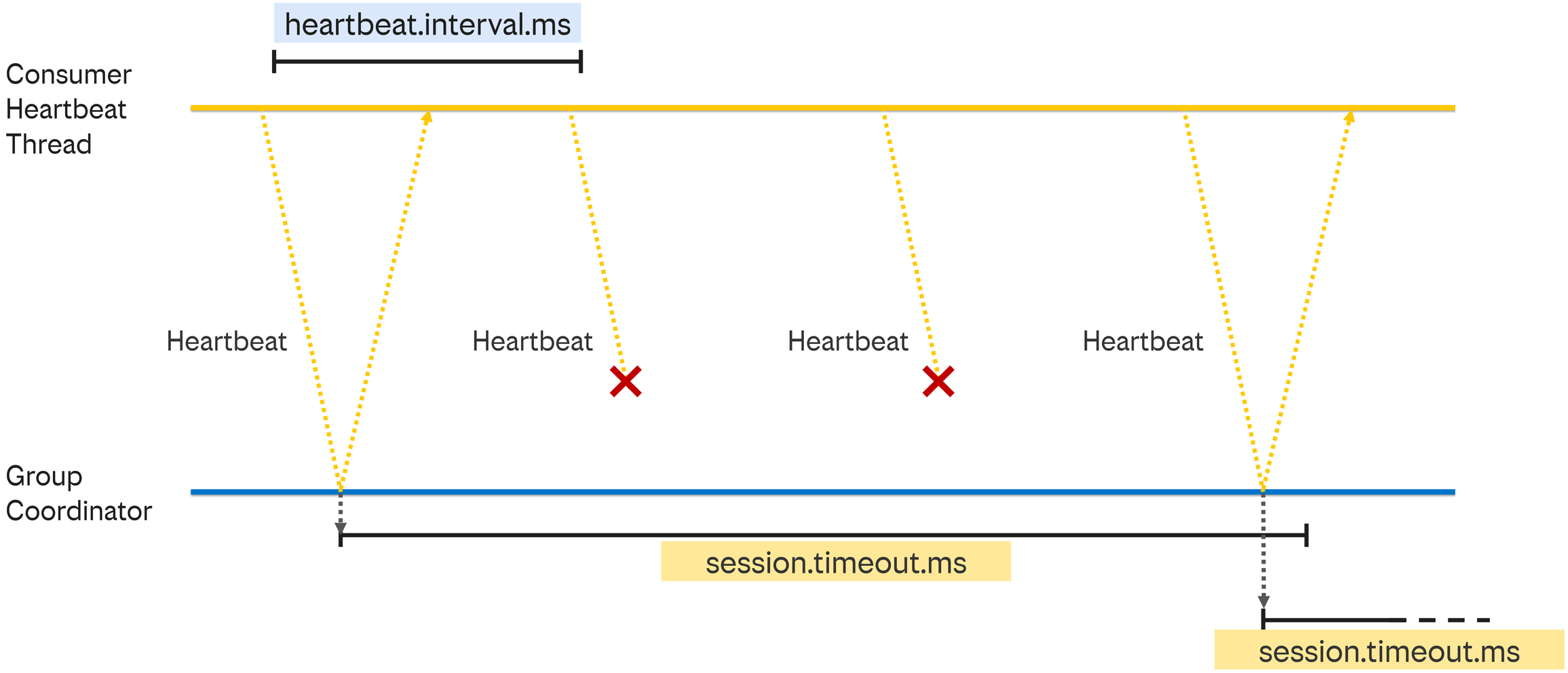
Параметры консюмеров



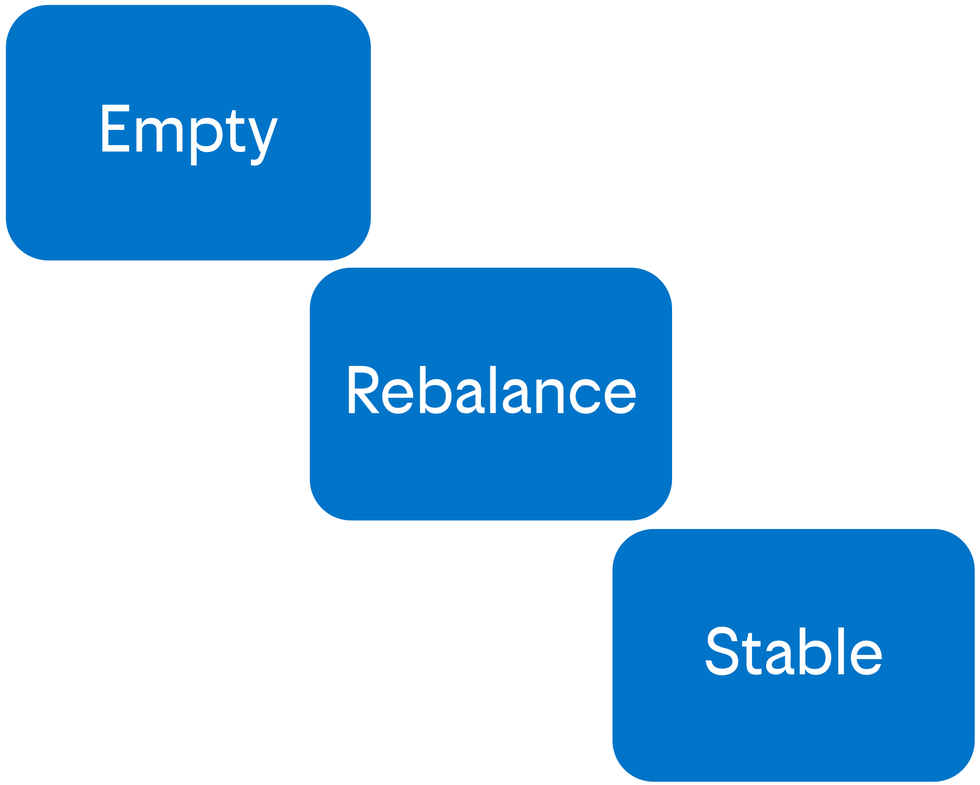
Параметры консюмеров



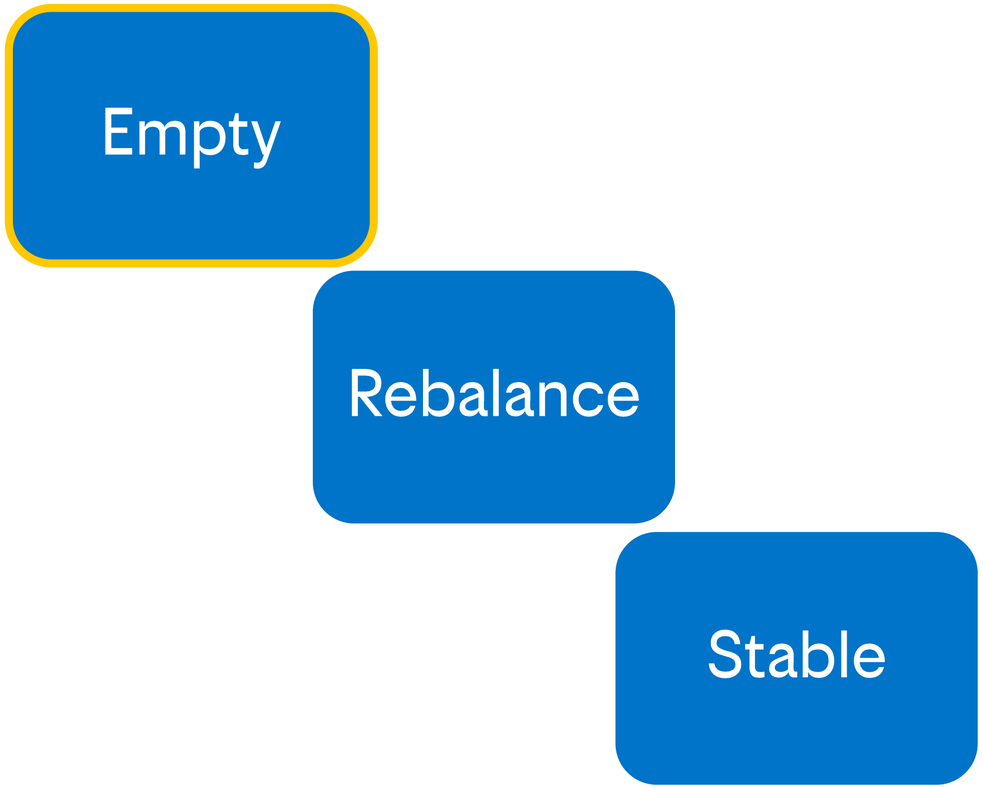
Параметры консюмеров



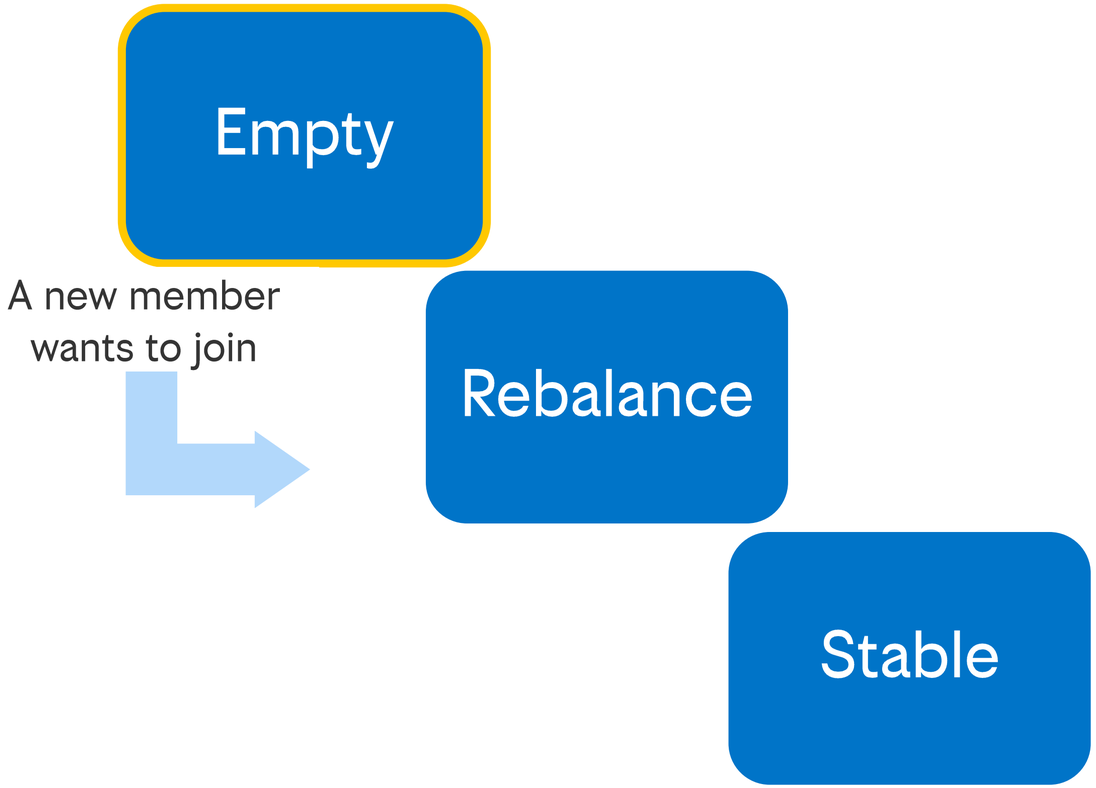
Причины ребалансировки



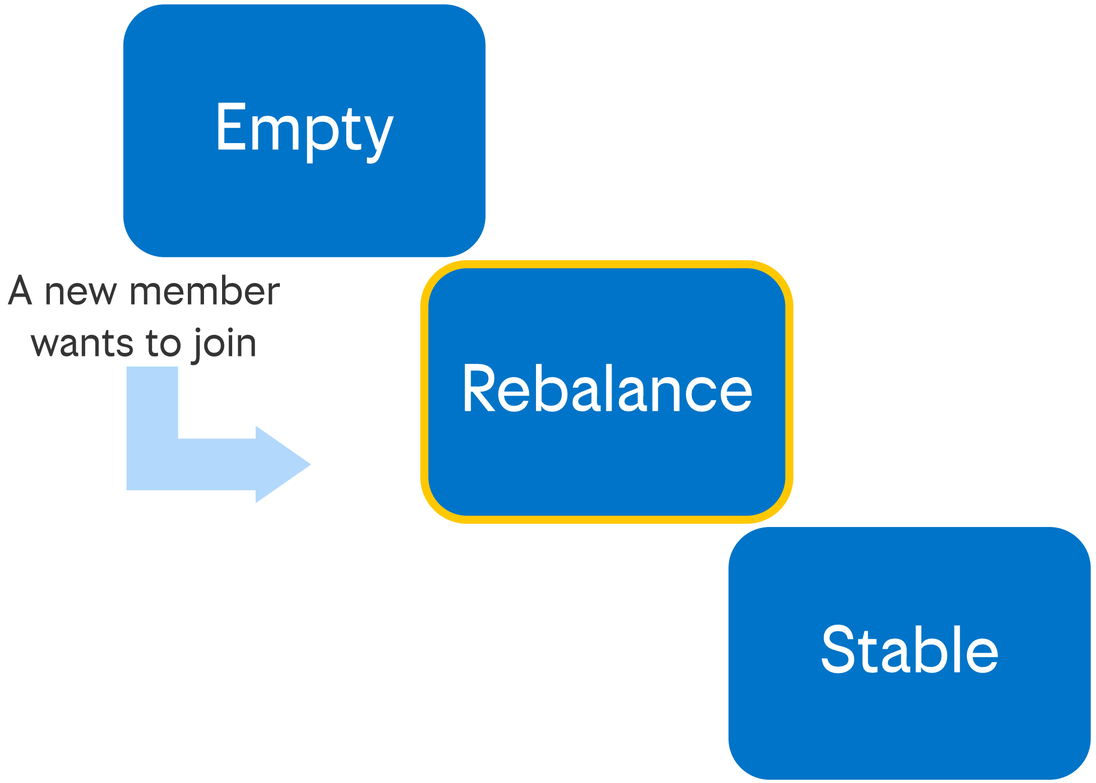
Причины ребалансировки



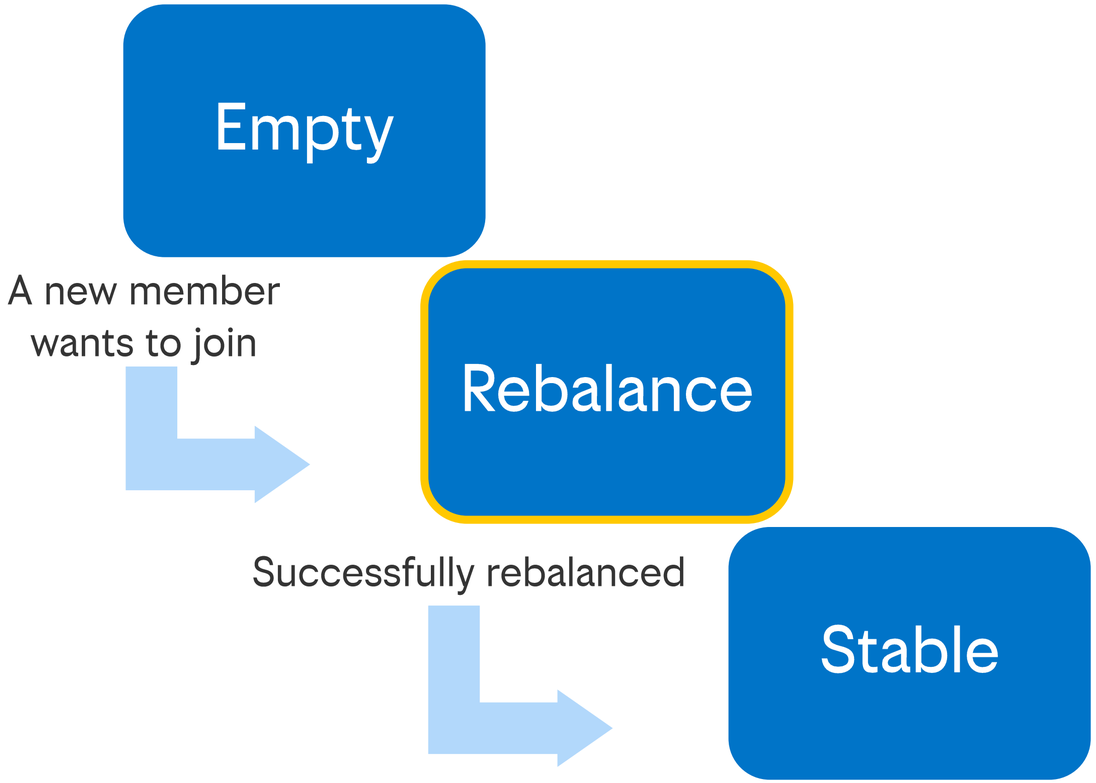
Причины ребалансировки



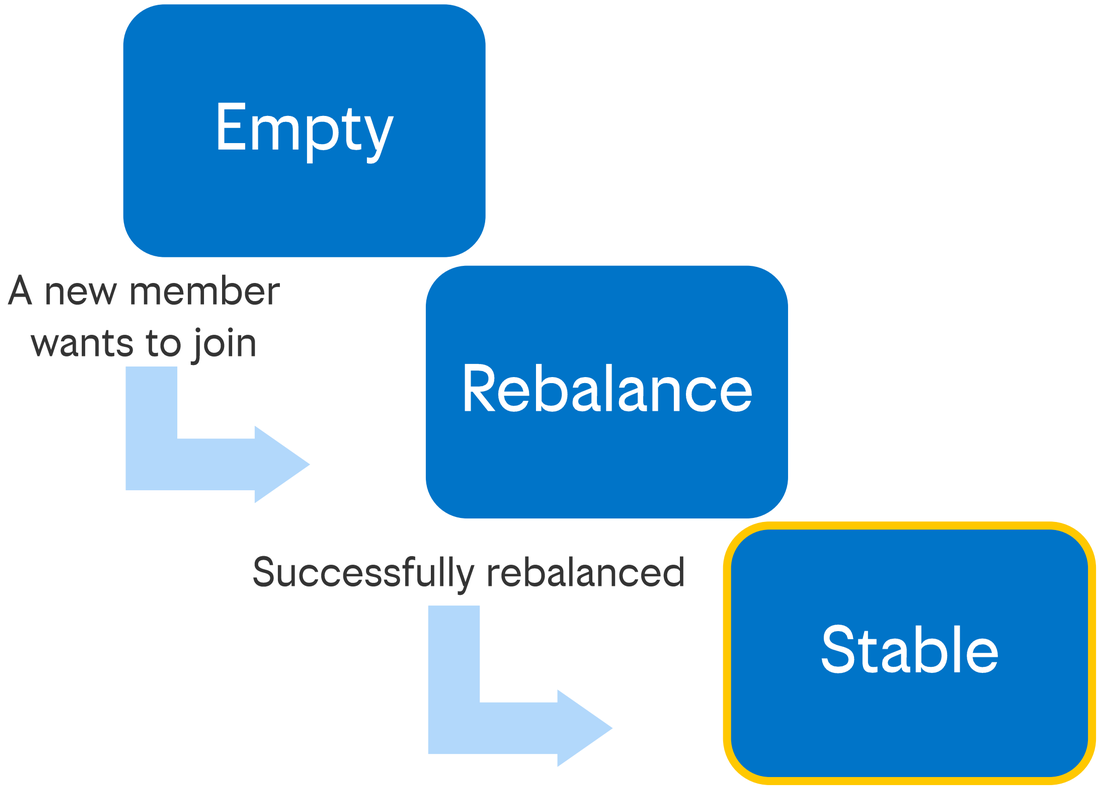
Причины ребалансировки



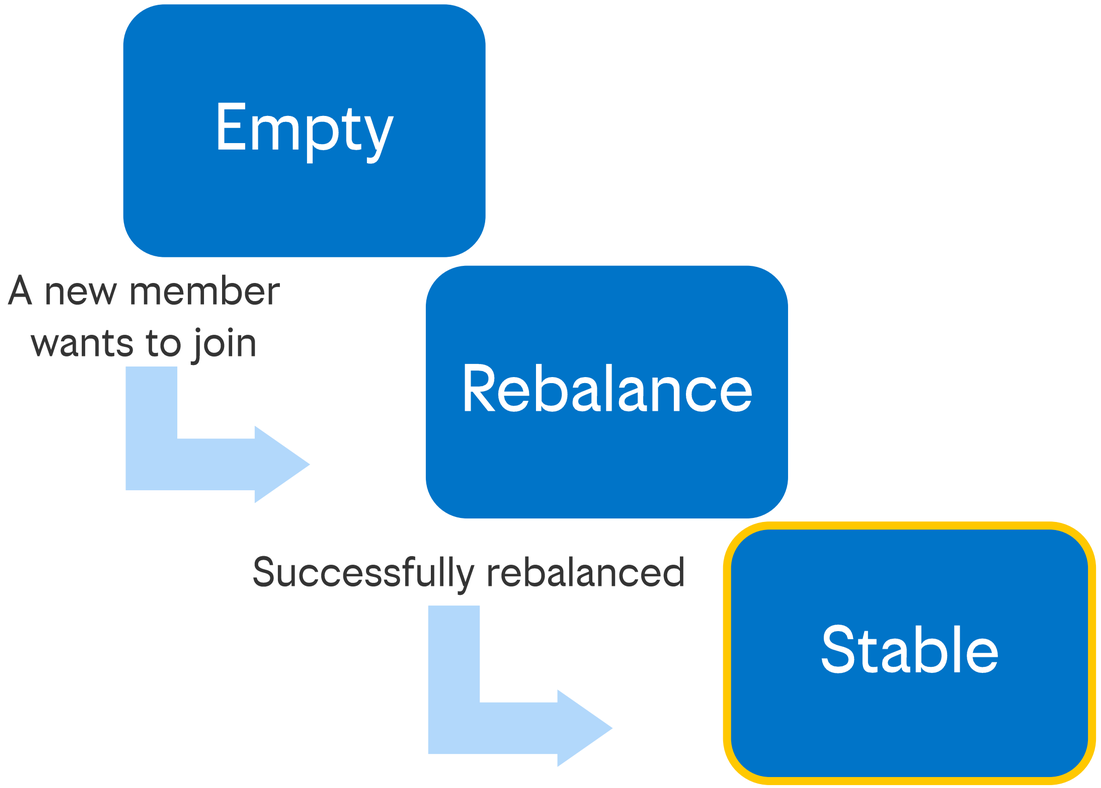
Причины ребалансировки



Причины ребалансировки

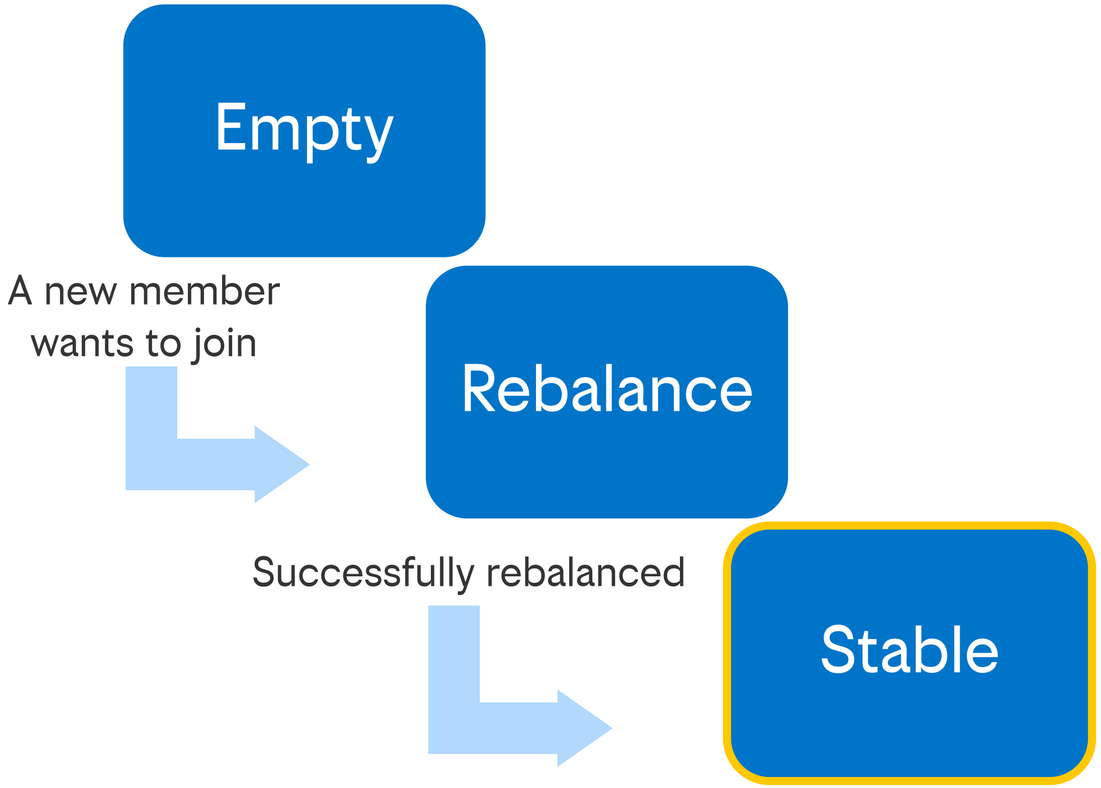


Причины ребалансировки



One of the following:

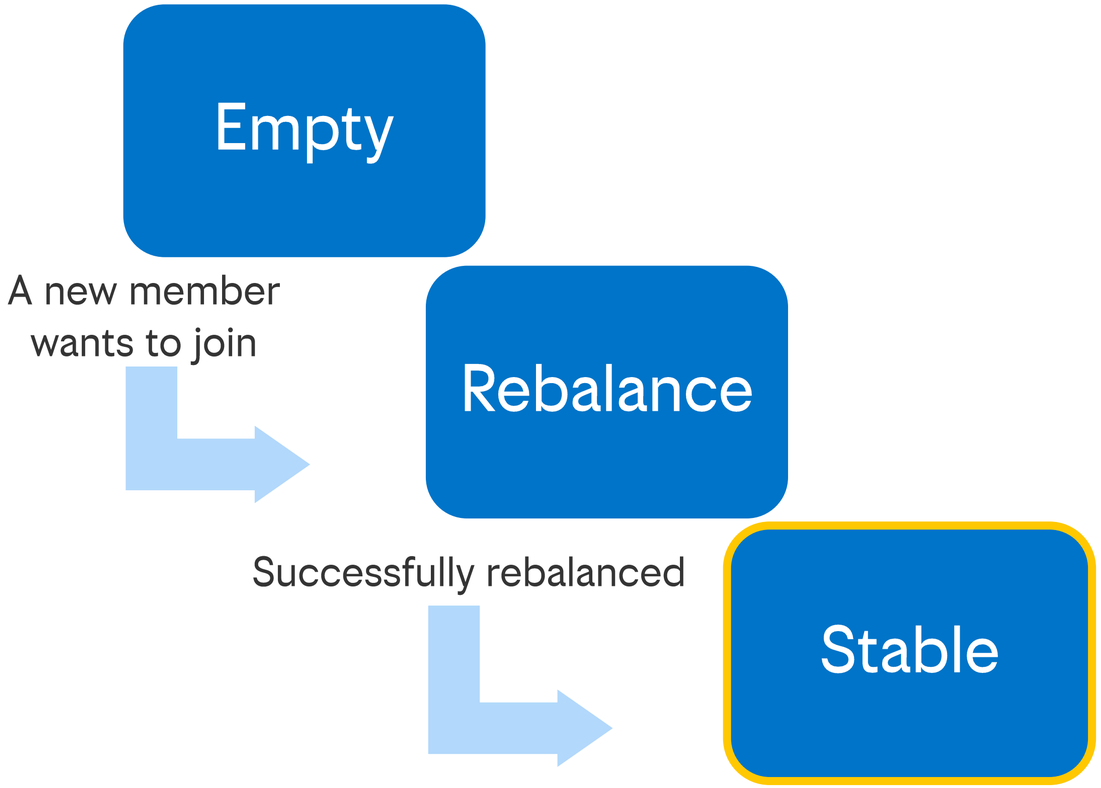
Причины ребалансировки



- Не сработал heartbeat (интервал оповещения кафки косюмером о том, что он всё ещё жив) от одного из consumer

One of the following:
- Heartbeat failure detected

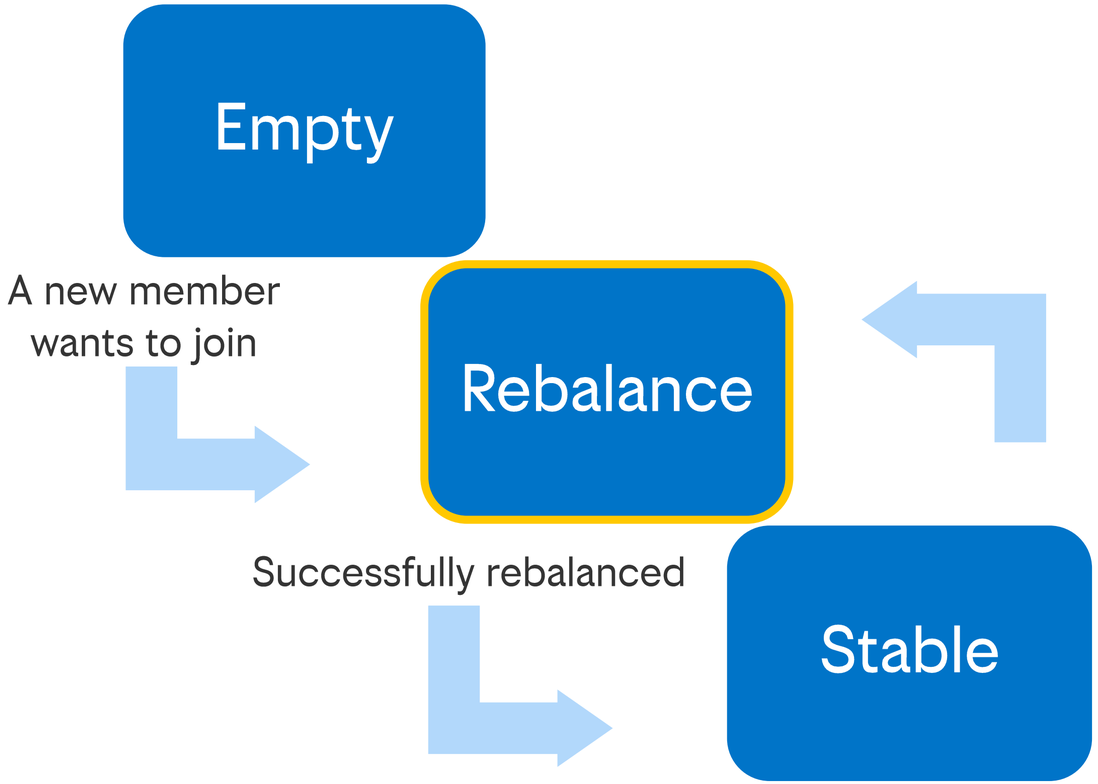
Причины ребалансировки



- Не сработал heartbeat (интервал оповещения кафки косюмером о том, что он всё ещё жив) от одного из consumer
- Превышено установленное ожидания получения следующего события (по умолчанию 5 минут).

One of the following:
- Heartbeat failure detected
- Members has left

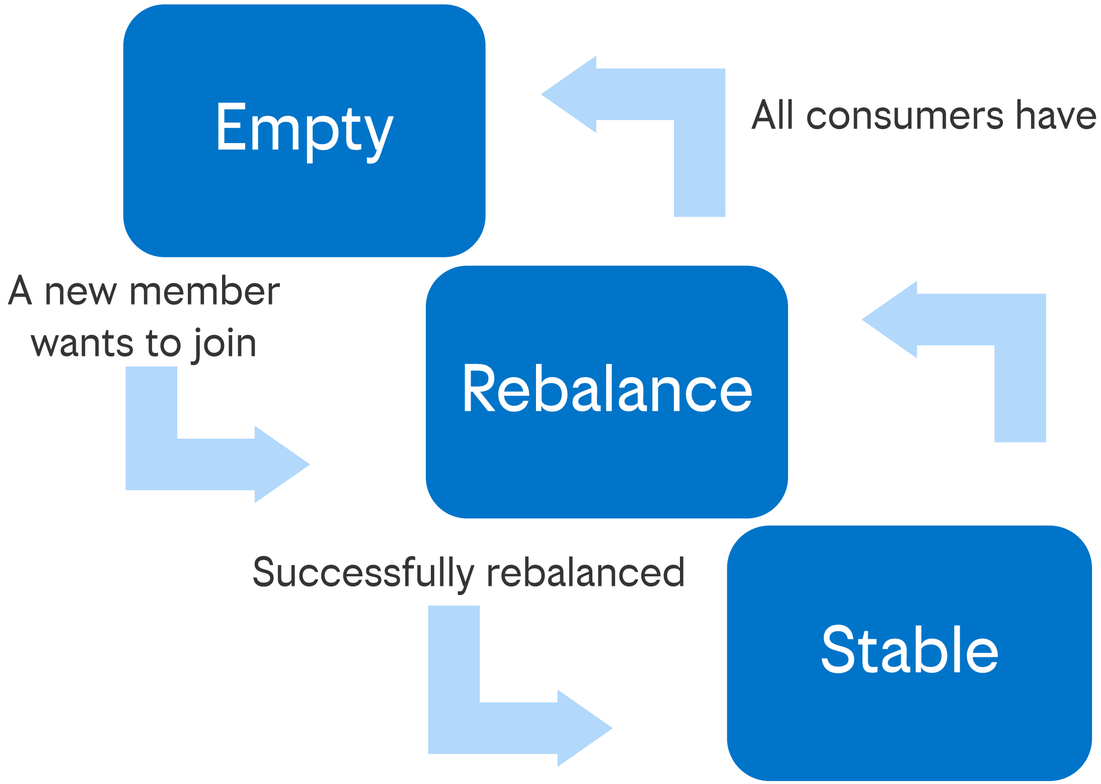
Причины ребалансировки



- Не сработал heartbeat (интервал оповещения кафки косюмером о том, что он всё ещё жив) от одного из consumer
- Превышено установленное ожидания получения следующего события (по умолчанию 5 минут).
- Подключение нового consumer.

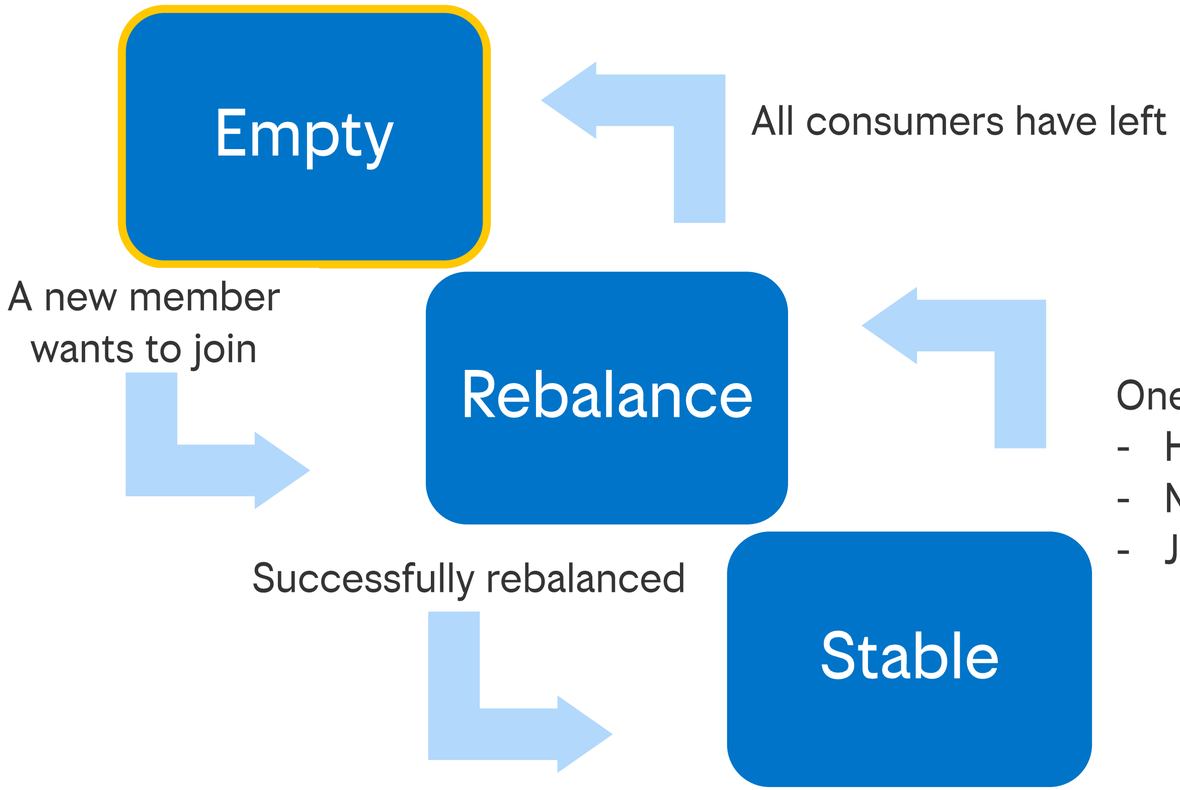
One of the following:
- Heartbeat failure detected
- Members has left
- JoinGroup received

Причины ребалансировки



- Не сработал heartbeat (интервал оповещения кафки косюмером о том, что он всё ещё жив) от одного из consumer
- Превышено установленное ожидания получения следующего события (по умолчанию 5 минут).
- Подключение нового consumer.

Причины ребалансировки

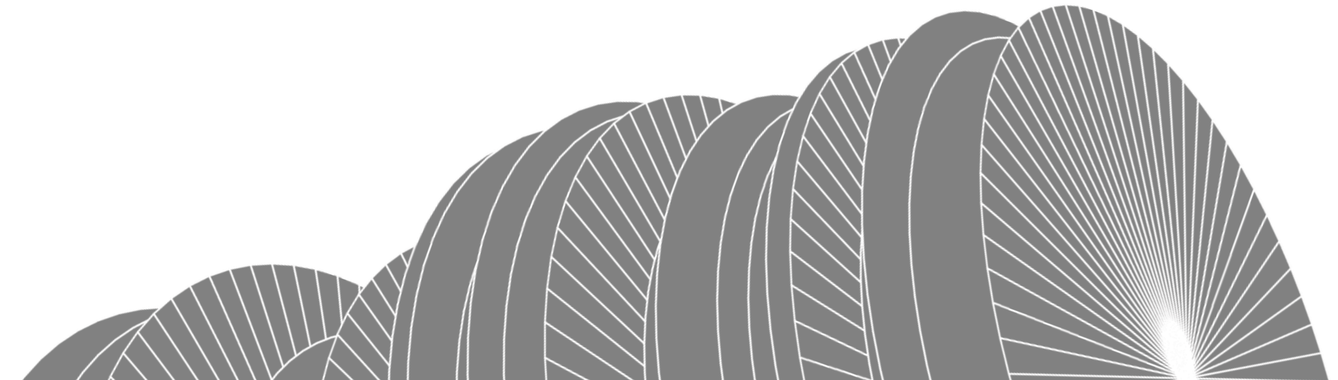


- Не сработал heartbeat (интервал оповещения кафки косюмером о том, что он всё ещё жив) от одного из consumer
- Превышено установленное ожидания получения следующего события (по умолчанию 5 минут).
- Подключение нового consumer.

Решение проблем с консьюмингом

1. **Проверить логику консюмеров** (долгая обработка).

Сработал таймаут. Консюмер вышел из группы. Но само приложение работает и думает, что всё ок.

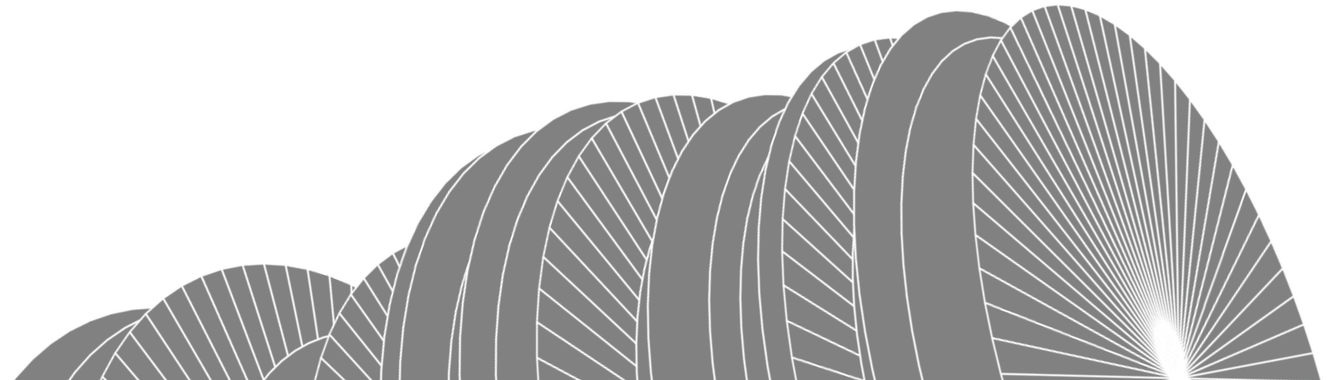


Решение проблем с консьюмингом

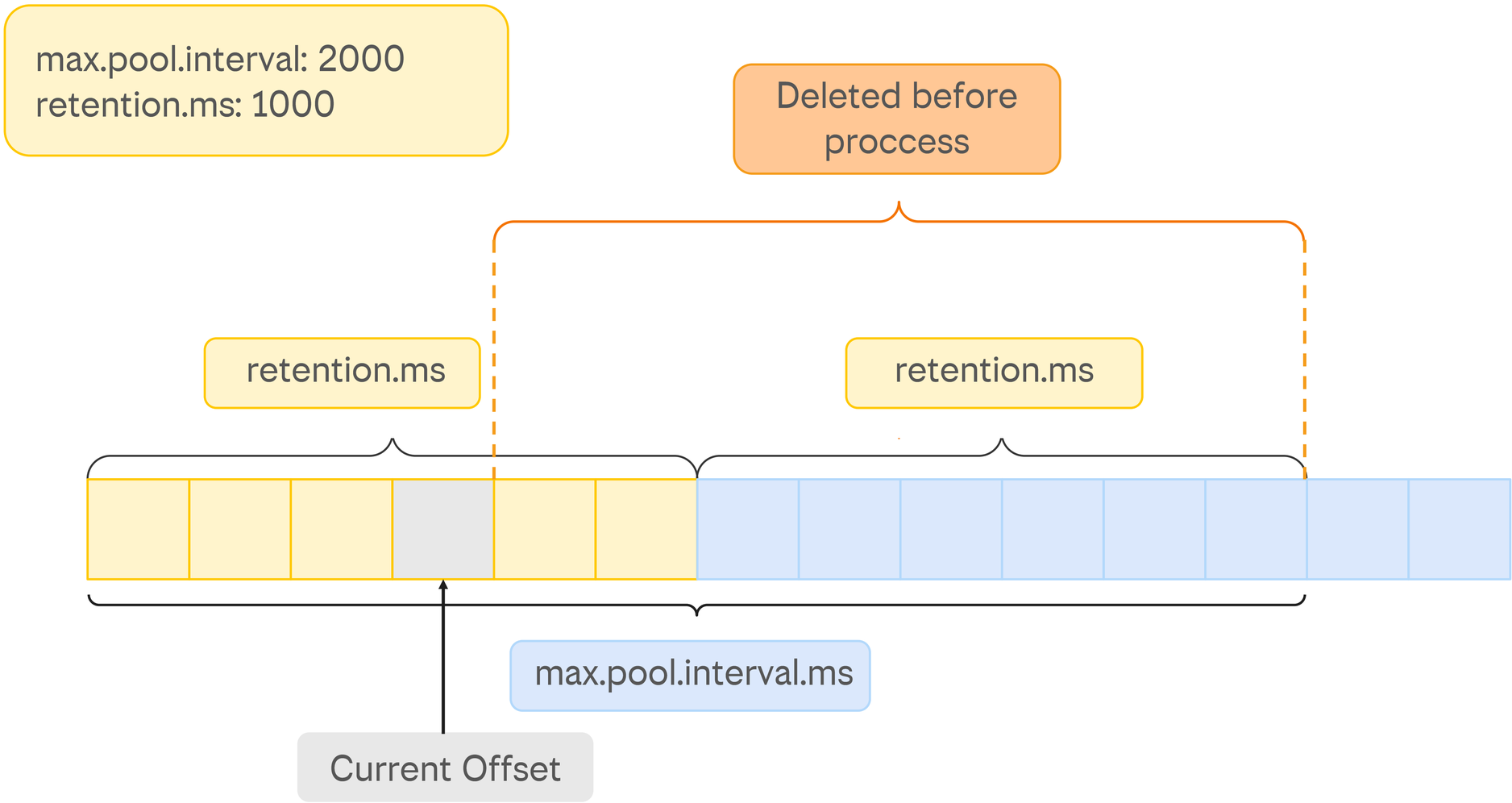
1 **Проверить логику консюмеров** (долгая обработка).

Сработал таймаут. Консюмер вышел из группы. Но само приложение работает и думает, что всё ок.

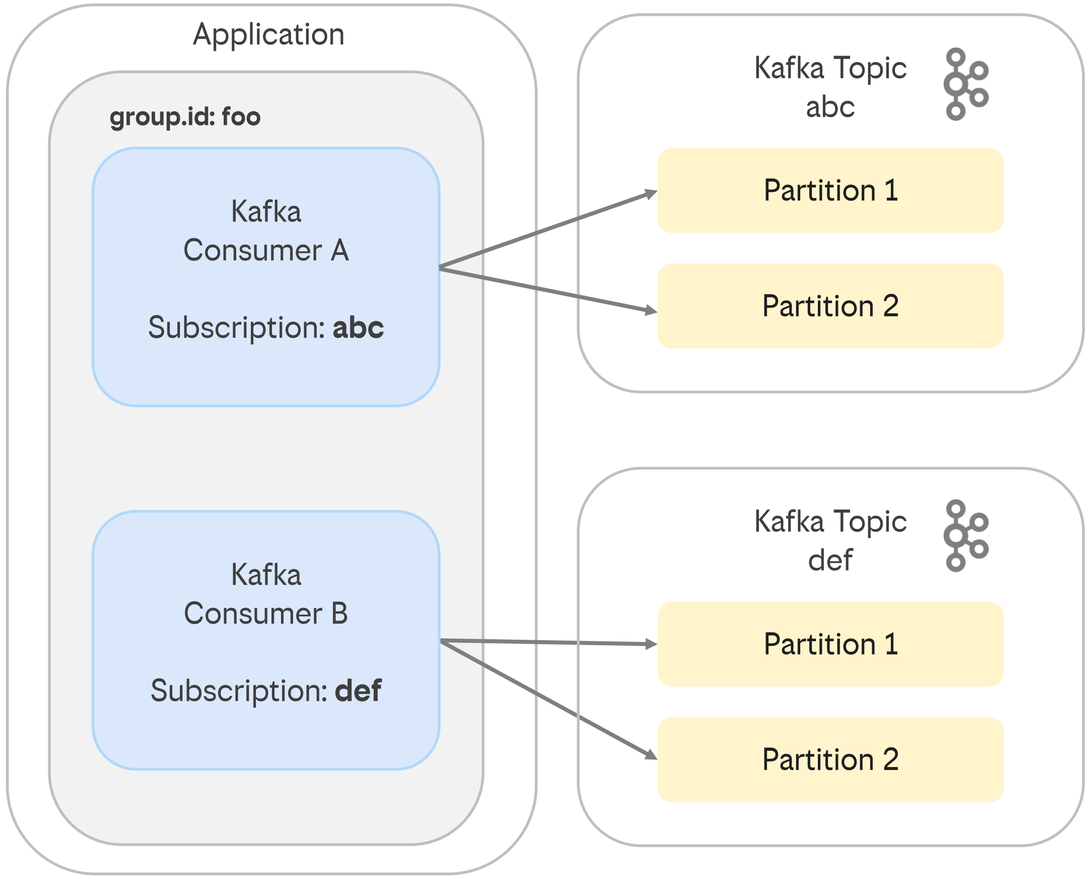
2 **Увеличить настройку max.poll.interval.ms –** время на обработку события (по умолчанию - 5 мин).

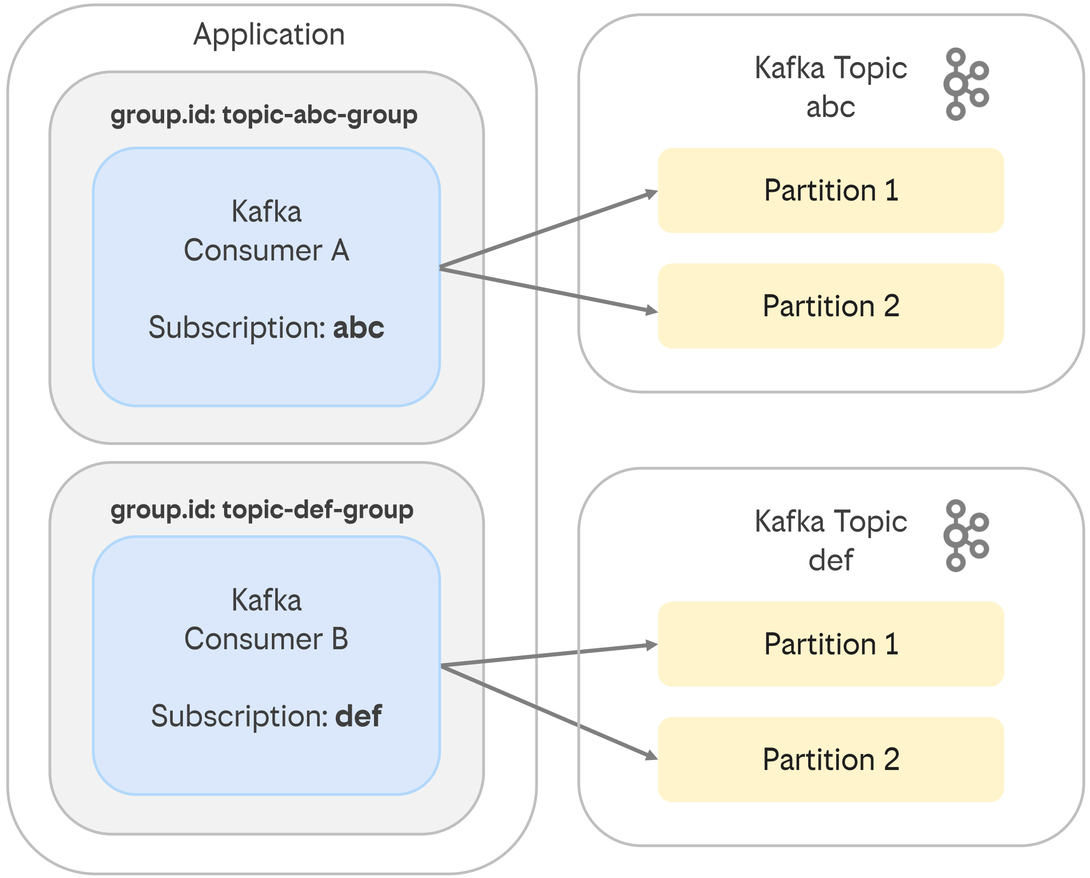


retention.ms < max.pool.interval.ms



Ребалансировка нескольких топиков



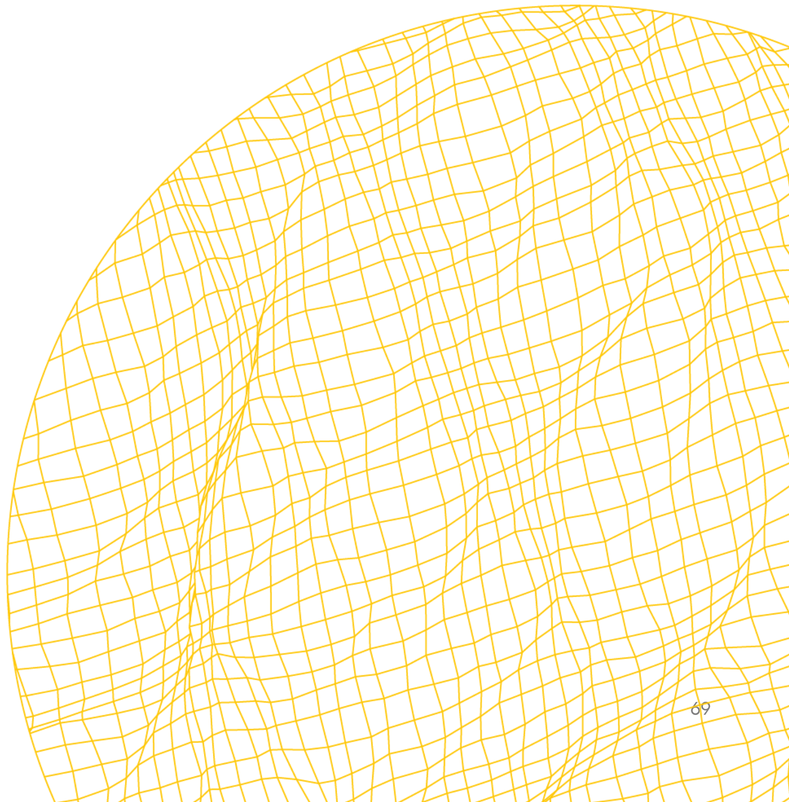


Колбэки при ребалансировке

SetPartitionsAssignedHandler(…) – назначены новые партиции

SetPartitionsRevokedHandler(…) – потеря владения патрициями

SetPartitionsLostHandler(…) – партиции назначены другому консюмеру (совместная ребалансировка)

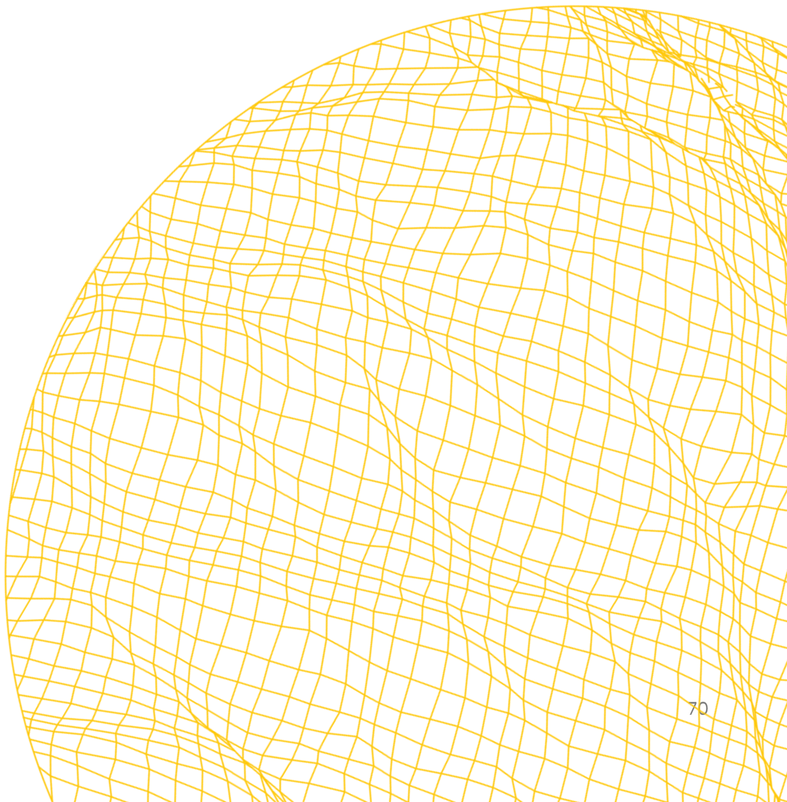


Колбэки при ребалансировке

SetPartitionsAssignedHandler(…) – назначены новые партиции

SetPartitionsRevokedHandler(…) – потеря владения патрициями

SetPartitionsLostHandler(…) – партиции назначены другому консюмеру (совместная ребалансировка)

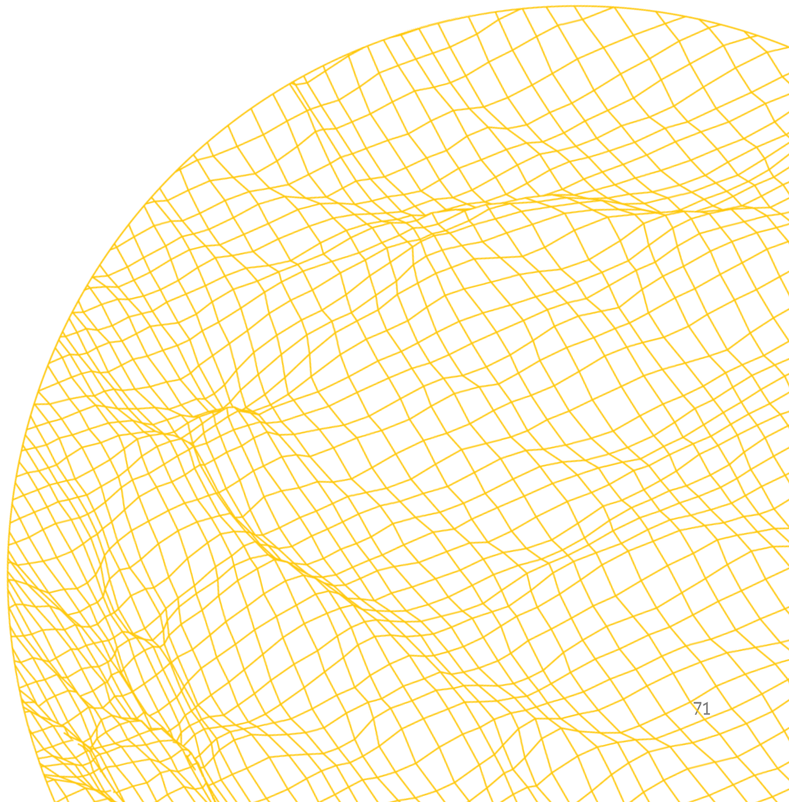


Колбэки при ребалансировке

SetPartitionsAssignedHandler(…) – назначены новые партиции

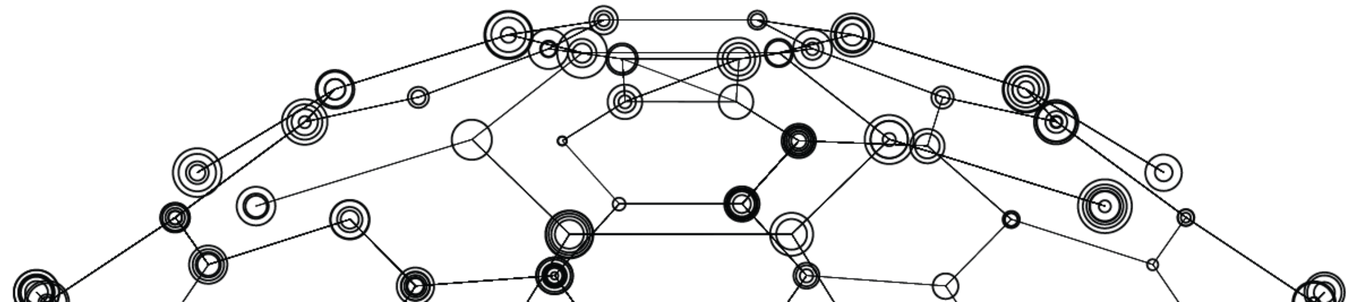
SetPartitionsRevokedHandler(…) – потеря владения патрициями

SetPartitionsLostHandler(…) – партиции назначены другому консюмеру (совместная ребалансировка)



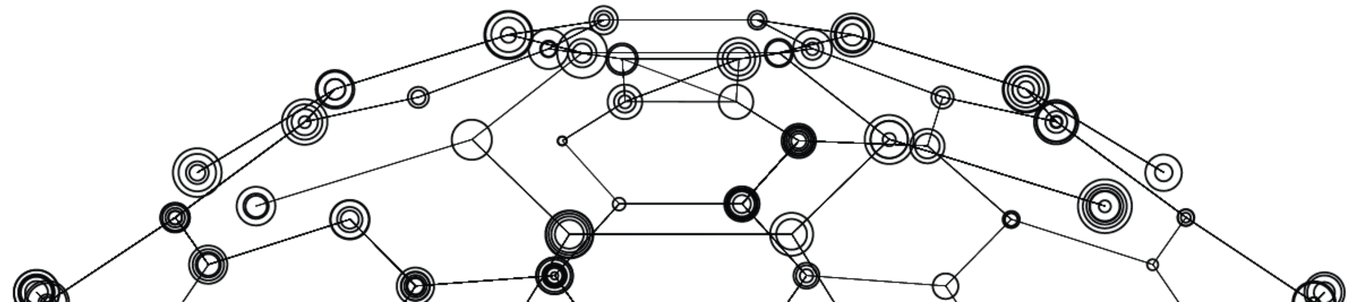
Callback при получении партиции

```
return new ConsumerBuilder<string, byte[]>(consumerConfig)
    .SetPartitionsAssignedHandler((consumer, partitions) =>
    {
        _logger.LogInformation($"Partitions assignment: [{string.Join(", ", partitions)}]");

        DoSmth(partitions);
        var offsets =_repository.GetOffsets(partitions);
        DoSmthWithOffsets(offsets);
    })
    .Build();
```



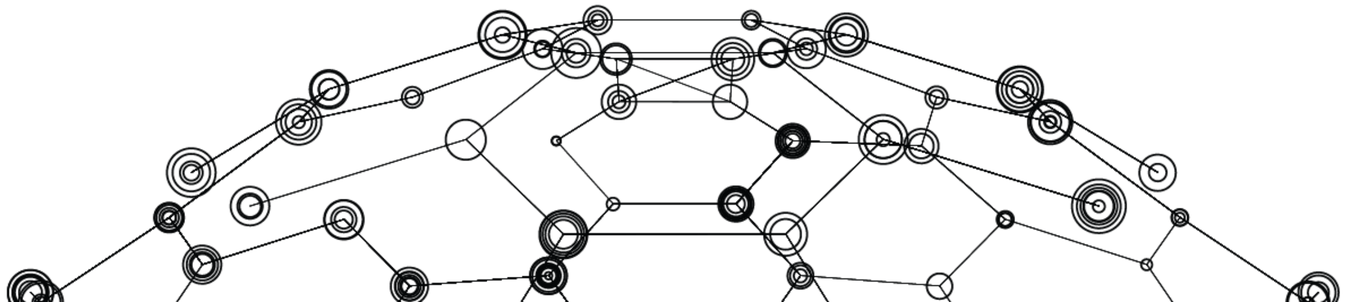
Callback при переназначении партиции

```
return new ConsumerBuilder<string, byte[]>(consumerConfig)
    .SetPartitionsRevokedHandler((consumer, partitions) =>
    {
        _logger.LogInformation($"Revoking assignment: [{string.Join(", ", partitions)}]");

        CommitOffsets(partitions);

        _repository.SaveOffsets(partitions);
    })
    .Build();
```



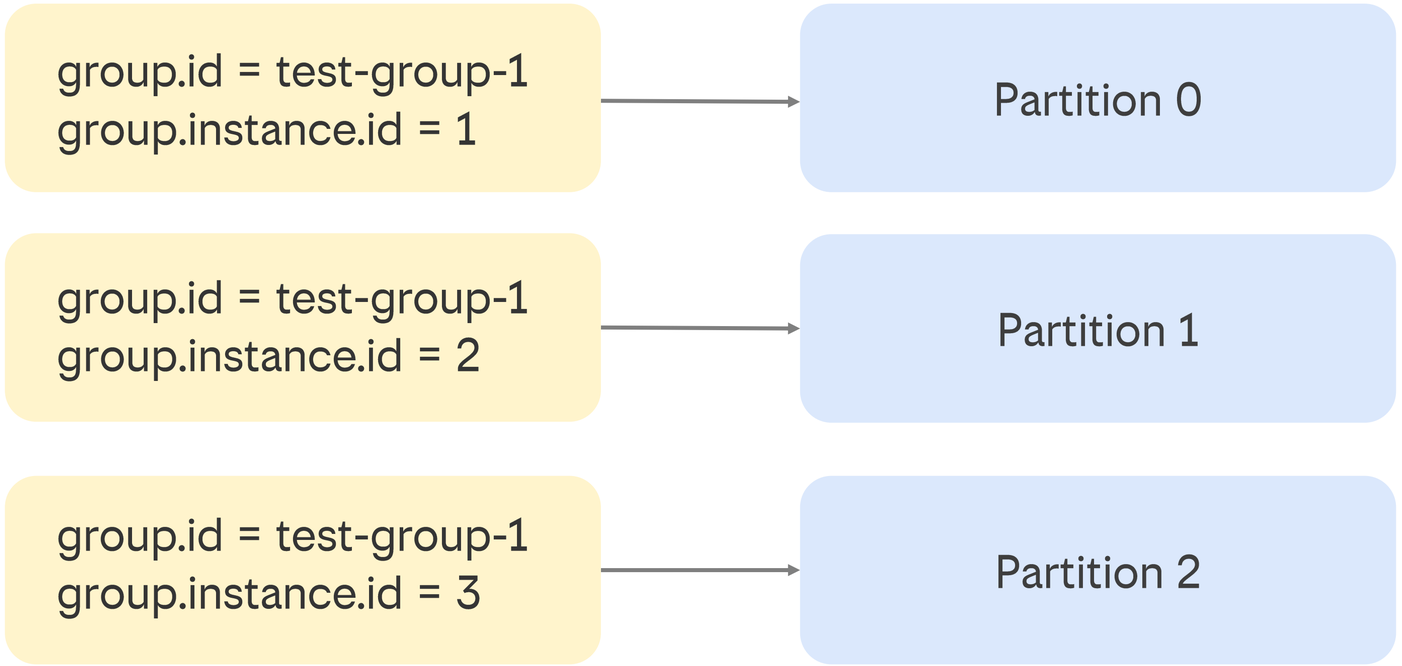
Callback при потере партиции

```
return new ConsumerBuilder<string, byte[]>(consumerConfig)
    .SetPartitionsLostHandler((consumer, partitions) =>
    {
        _logger.LogInformation($"Lost ownership: [{string.Join(", ", partitions)}]");

        DoSmthWithOffsets(partitions);

        _repository.SaveOffsetsMetadata(partitions);
    })
    .Build();
```



Статические члены группы

group.instance.id – параметр для закрепления партиции за консюмером



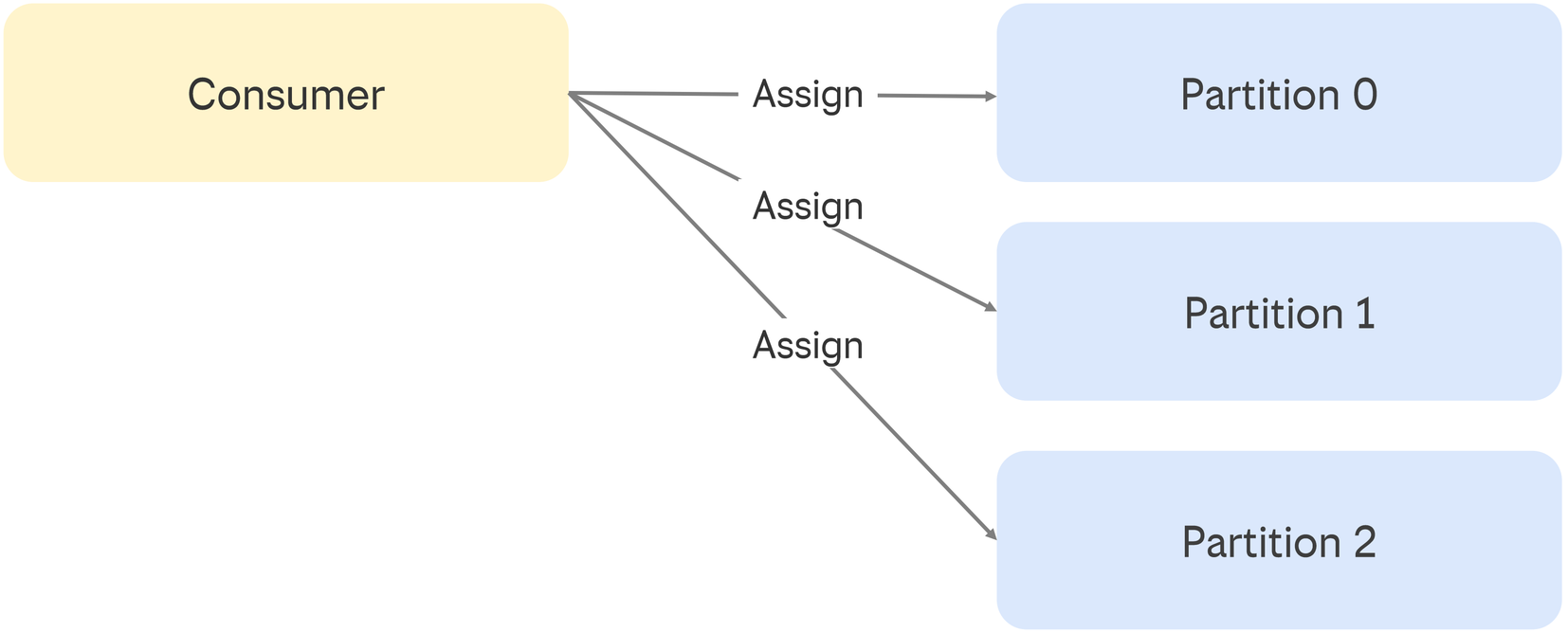
Зависит от **session.timeout.ms**

Автономный консюмер

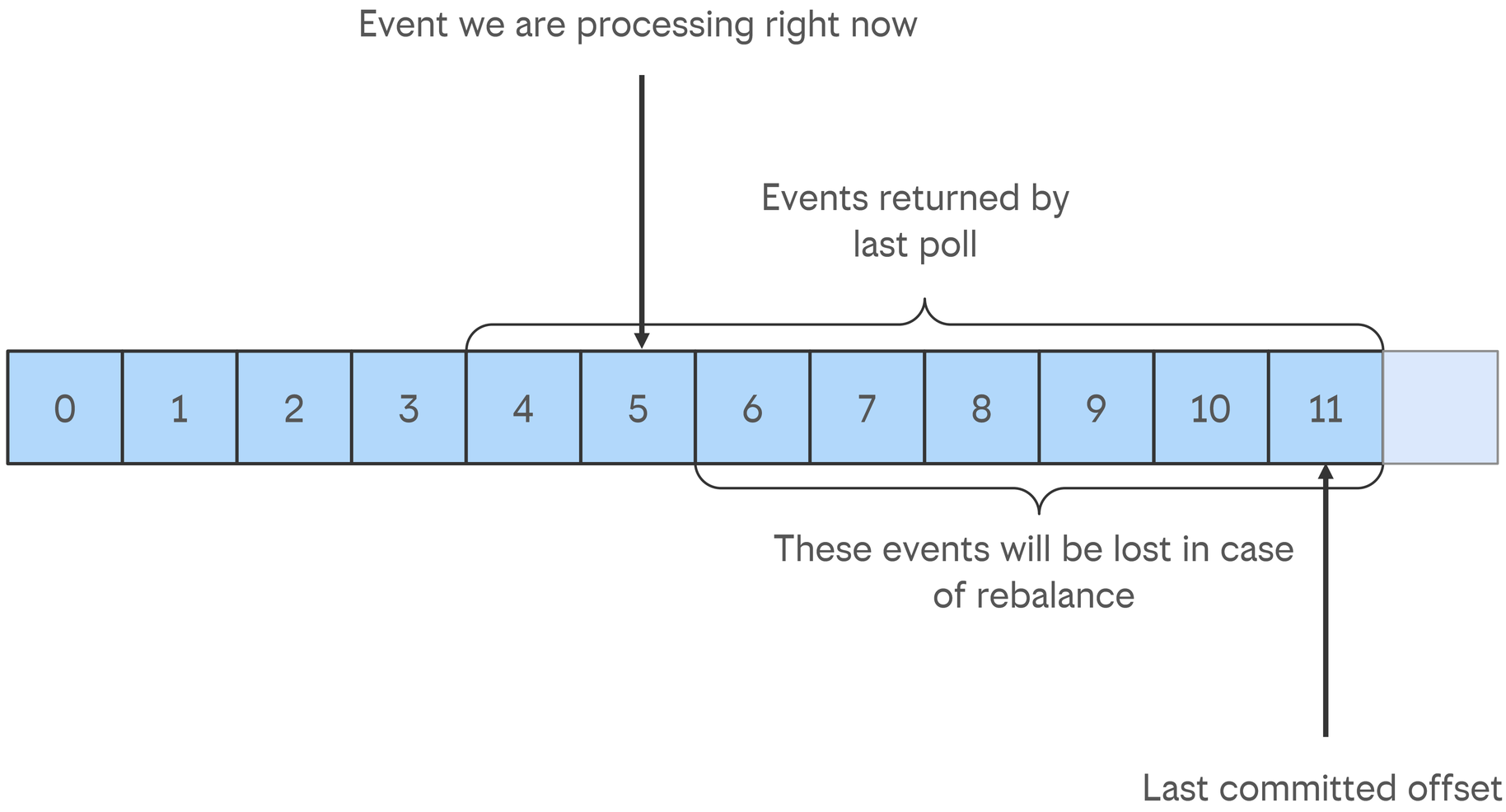
```
consumer.Assign(new TopicPartitionOffset(messagesTopicPartition, new
Offset(offsetMore.Value)));
```

Partition: 0

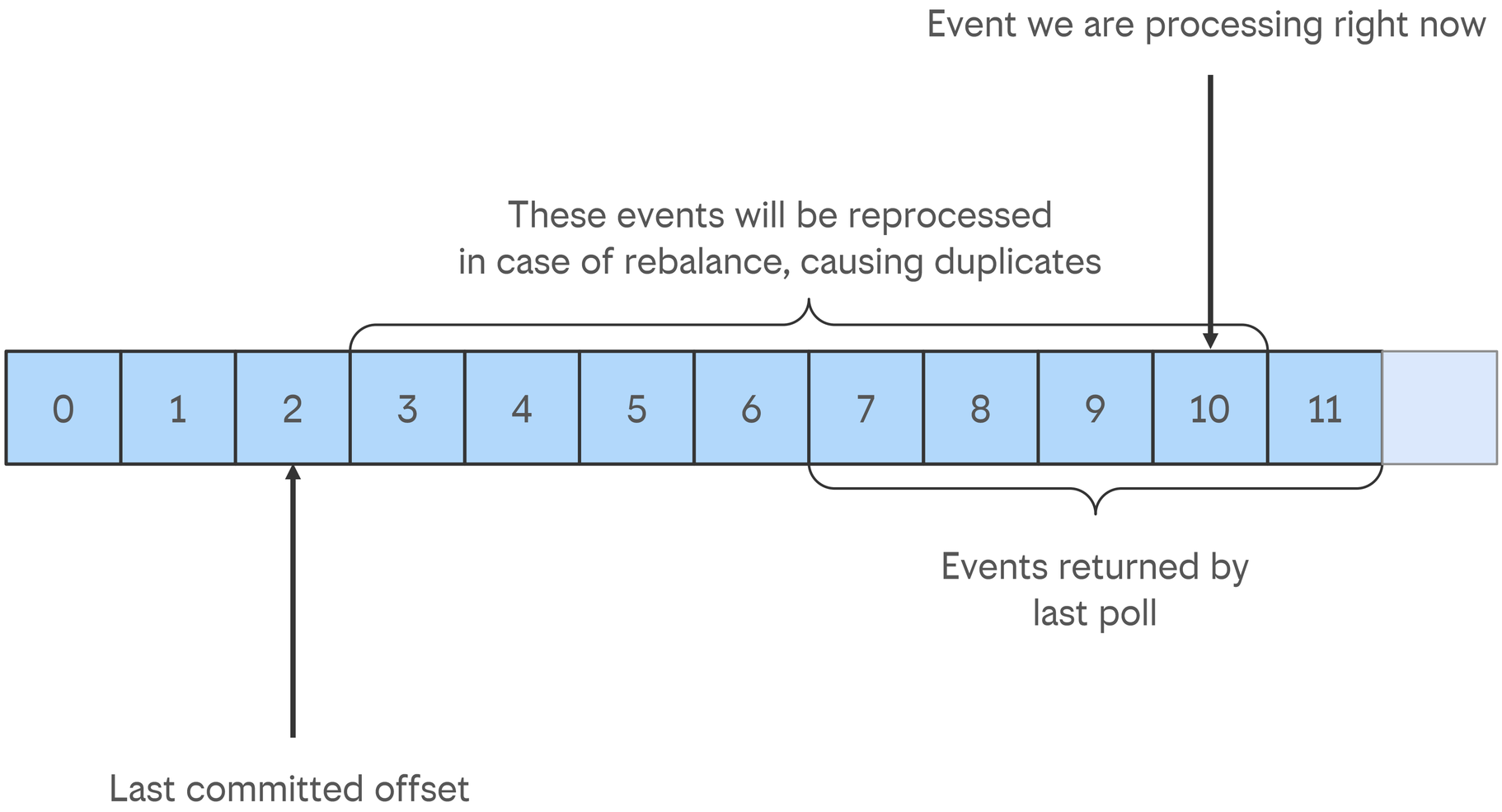
Offset: 0



```
"AutoOffsetReset": "earliest",
"EnableAutoOffsetStore": true
```



```
"AutoOffsetReset": "earliest",
"EnableAutoOffsetStore": false
```



Ручной Commit

```
"AutoOffsetReset": "earliest",
"EnableAutoOffsetStore": false,
"EnableAutoCommit": false
```

```
var result =
consumer.Consume(TimeSpan.FromMilliseconds(10));

if (result == null || result.IsPartitionEOF)
{
    continue;
}
```

```
consumer.StoreOffset(result.TopicPartitionOffset);
consumer.Commit(new List<TopicPartitionOffset>
{
    result.TopicPartitionOffset
});
```

Мой консюмер не читает события

1 Версия брокера кафка в кластере сильно разошлась с текущей версией библиотеки Confluent's Kafka .NET.

Решение: Необходимо обновить версию библиотеки до последней актуальной, относительно используемого брокера.

Мой консюмер не читает события

1

Версия брокера кафка в кластере сильно разошлась с текущей версией библиотеки Confluent's Kafka .NET.

Решение: Необходимо обновить версию библиотеки до последней актуальной, относительно используемого брокера.

2

В одной группе находится несколько консюмеров и событие прочитал другой.

Решение: Проверить правильность выбора консюм группы.

Мой консюмер не читает события

1. Версия брокера кафка в кластере сильно разошлась с текущей версией библиотеки Confluent's Kafka .NET.

Решение: Необходимо обновить версию библиотеки до последней актуальной, относительно используемого брокера.

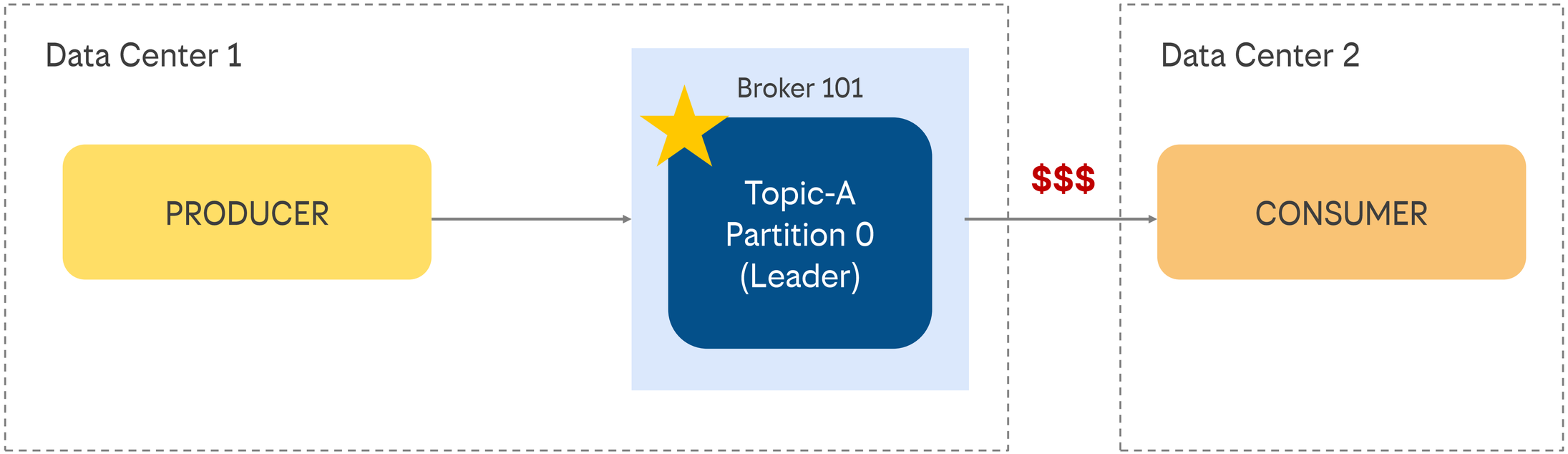
2. В одной группе находится несколько консюмеров и событие прочитал другой.

Решение: Проверить правильность выбора консюм группы.

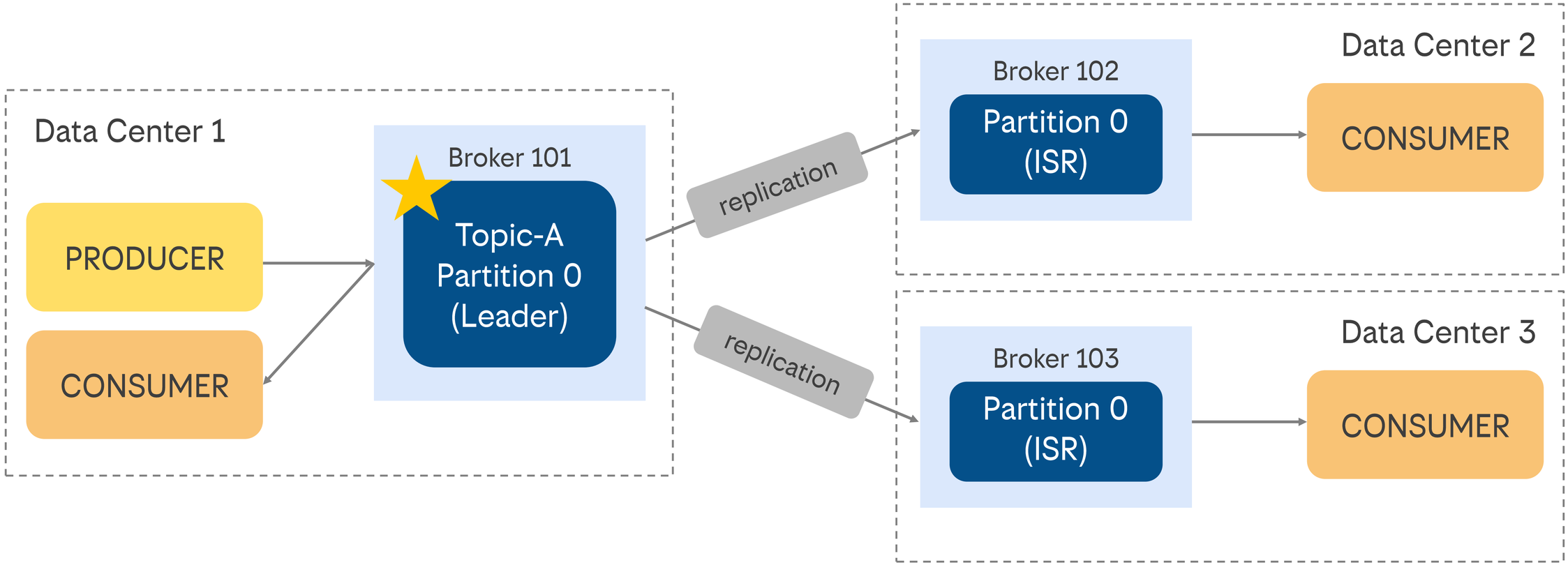
3. Неправильные параметры конфигурации, либо слишком долгая обработка в коде. (консюмер выпал из консюм группы)

Решение: Проверить кодовую базу и параметры конфигурации.

Чтение из ведомой реплики



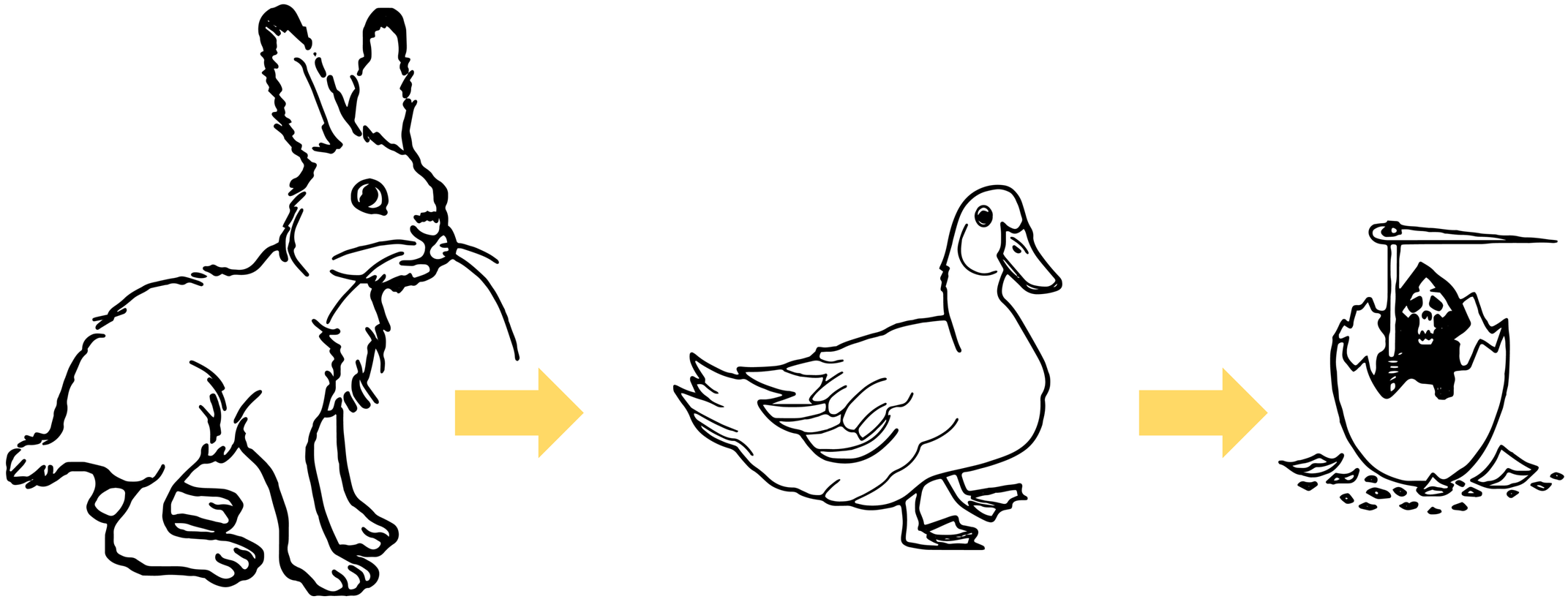
Чтение из ведомой реплики



Broker setting (must be version Kafka v2.4+):
rack.id config must be set to the data centre ID
rack.id=dc-2
replica.selector.class must be set
to org.apache.kafka.common.replica.**RackAwareReplicaSelector**

Consumer Client setting (v2.4+):
Set client.rack to the data centre ID the consumer is launched on
client.rack=dc-2

Вложенность библиотек



Библиотека предприятия **Confluent.Kafka .Net** **librdkafka**

Насколько глубока кроличья нора

```
"Kafka": {
  "Consumer": {
              "log_level": 7,
              "debug": "all"
  },
  "Producer": {
              "log_level": 7,
              "debug": "all"
  }
},
```

```
"log_level": [1...7]
```

Consumer	Producer	Description
all	all	Все логи
	broker	
topic	topic	
	msg	
consumer		
cgrp		
fetch		

generic, broker, topic, metadata, queue, msg, protocol, cgrp, security, fetch, feature, interceptor, plugin, all

```
IConfiguration configuration = new ConfigurationBuilder()
    .AddJsonFile("./appsettings.json")
    .Build();

var pConfig = configuration.GetSection(«Kafka:Producer").Get<ProducerConfig>();

var cConfig = configuration.GetSection(“Kafka:Consumer").Get<ConsumerConfig>();
```

Хендлер обработки ошибки

```
builder.SetErrorHandler((consumer, error) =>
    {
        if (e.Code == ErrorCode.UnknownTopicOrPart)
        {
            DoSmth();
        }

        _logger.LogError(e.Reason);
    })
    .Build();
```

Хендлер получения статистики

```
"statistics.interval.ms": 10
```

```
builder.SetStatisticsHandler((consumer, jsonStatisticsString) =>
    {
        Console.WriteLine(jsonStatisticsString);
    })
    .Build();
```

```
{
  "name": "rdkafka#producer-1",
  ...
  "topics": {
  "test": {
    "topic": "test",
    "partitions": {
      "0": {
        "partition": 0,
        "broker": 3,
        "leader": 3,
          ...
      }
    }
  }
}
```

https://docs.confluent.io/platform/7.5/clients/librdkafka/html/md_STATISTICS.html

Хендлер логирования

```
builder.SetLogHandler((consumer, logMessage) => {
    _logger.Information("Topic {topic}. {message}", topic, m.Message);

    if (Regex.IsMatch(m.Message, ".*maximum poll interval.*exceeded.*"))
    {
        cancellationTokenSource.Cancel();
    }
});
```

Баг с логированием

Kafka Library Crashes when using ProducerConfig with SetLogHandler / SetErrorHandler #1834

New issue

Open | 3 of 8 tasks | albertherd opened this issue on Jun 9, 2022 · 4 comments

albertherd commented on Jun 9, 2022 • edited

Description

When creating a ProducerConfig and your application is running containerized (using the default Dockerfile from VS 2022), when a log is generated and is passed to the API, the application just terminates. It even ignores the try-catch (maybe there is no exception being raised), the application just terminates.

- I reproduced this in nuget package version 1.8.2
- OS is windows, but Docker container is Linux
- You don't need a Kafka instance running
- No logs are emitted
- Tested on different machines, both Windows 10 and Windows 11

How to reproduce

- Clone my repo - https://github.com/albertherd/KafkaProducerConfigIssue
- Run it
- When the swagger UI comes up, run the main endpoint
- The app will try to communicate with a non-existant kafka instance, will try to log and the app will immediately exit.

Assignees
No one assigned

Labels
bug LOW

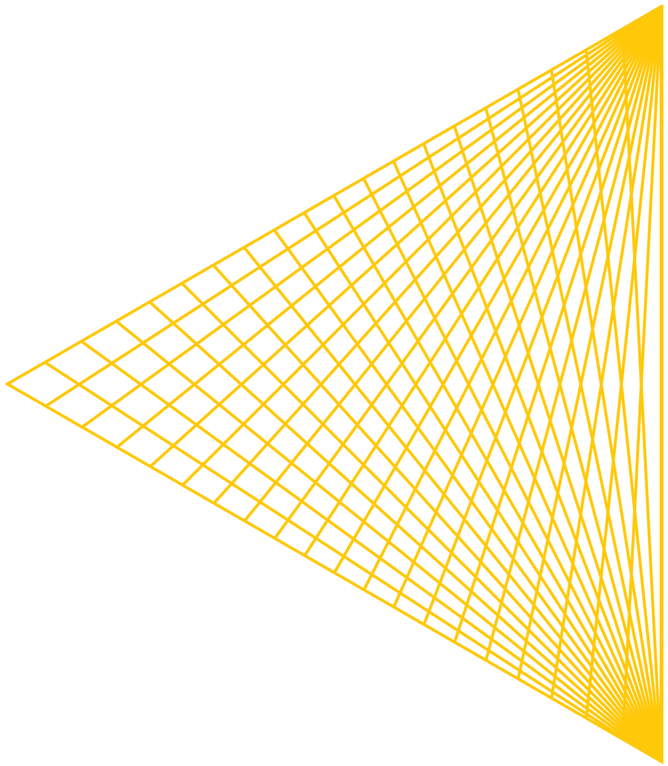
Projects
None yet

Milestone
No milestone

Development
No branches or pull requests

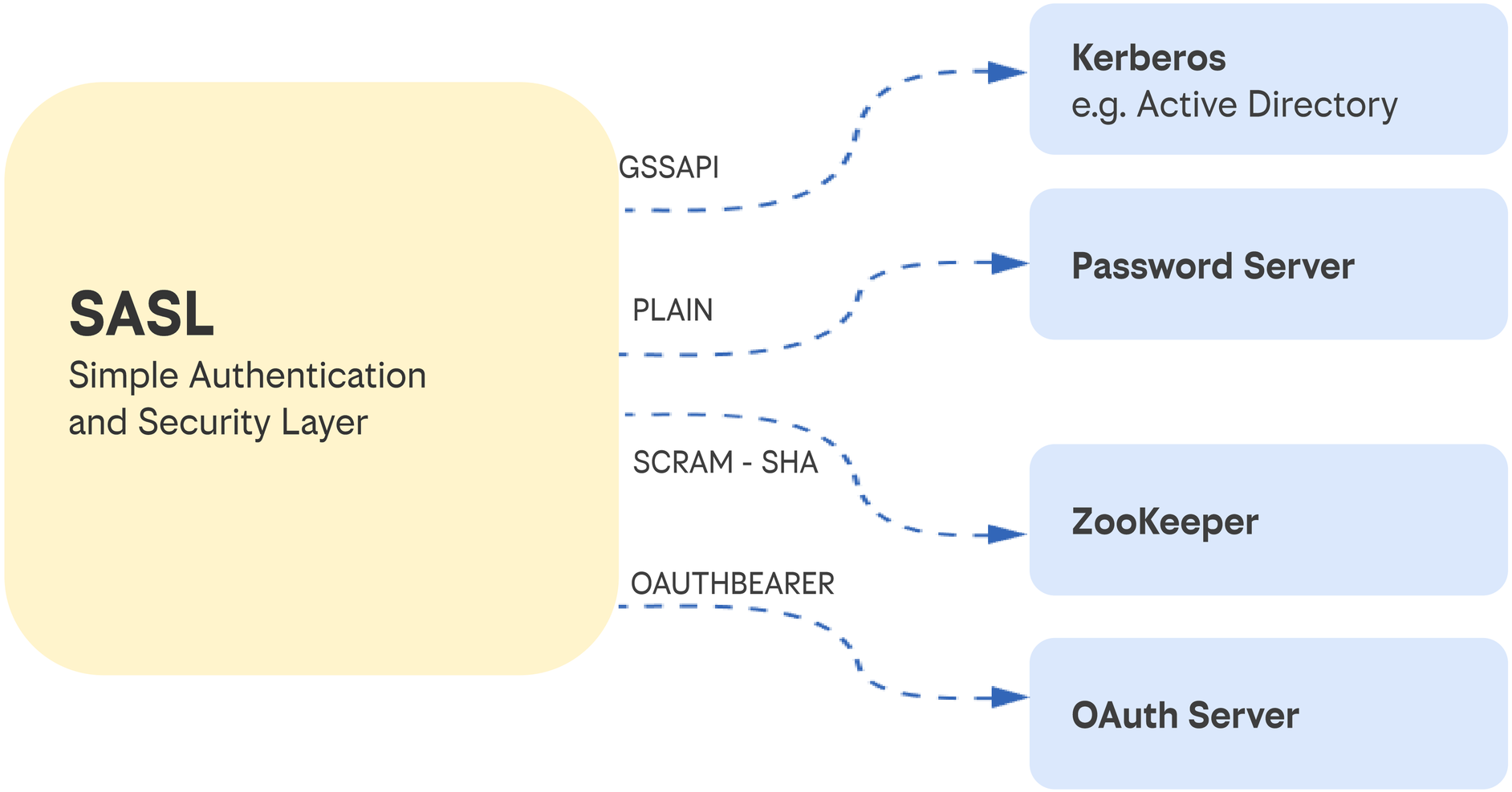
Notifications Customize
Subscribe
You're not receiving notifications from this thread.

https://github.com/confluentinc/confluent-kafka-dotnet/issues/1834



Аутентификация

Типы аутентификации (SASL)



GSSAPI (Kerberos authentication)

1 **Проблемный debug под windows, Kerberos аутентификация работает только под linux или в docker.**

GSSAPI (Kerberos authentication)

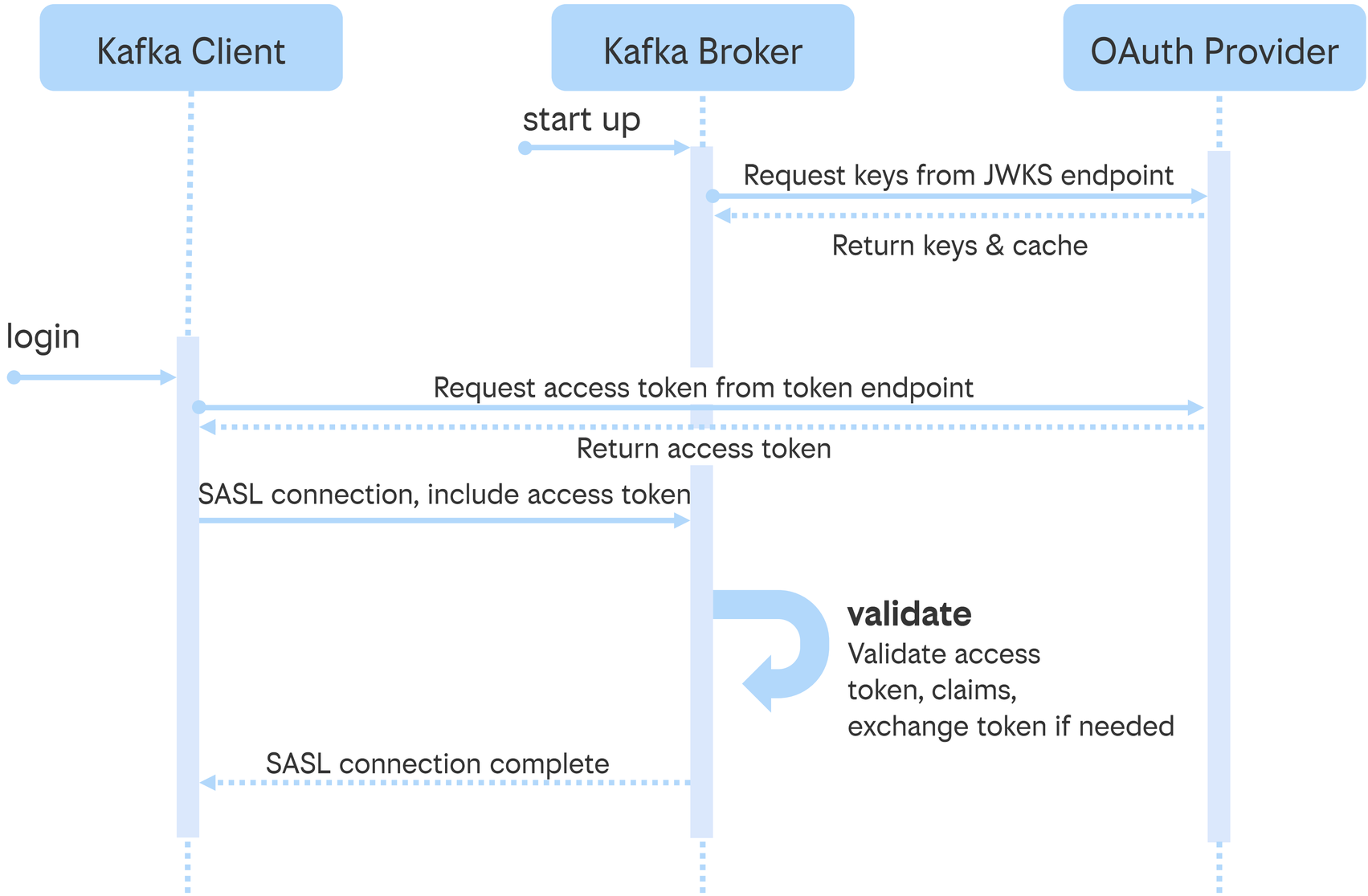
1. Проблемный debug под windows, Kerberos аутентификация работает только под linux или в docker.

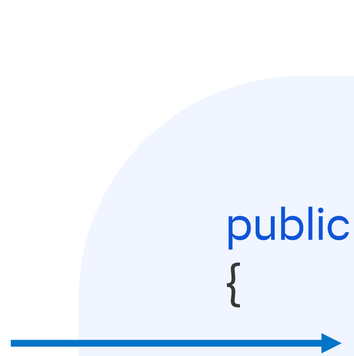
2. Нужно генерировать файл keytab.

GSSAPI (Kerberos authentication)

1. Проблемный debug под windows, Kerberos аутентификация работает только под linux или в docker.
2. Нужно генерировать файл keytab.
3. Нужны небольшие танцы с бубном по поводу dockerfile.

OAUTHBEARER





```
public ProducerService()
{
    void OauthCallback(IClient client, string cfg)
    {
        try
        {
            var token = retrieveToken(config[“RefreshToken"]);
            client.OAuthBearerSetToken(token.TokenValue, token.Expiration, token.Principal);
        }
        catch (Exception e)
        {
            client.OAuthBearerSetTokenFailure(e.ToString());
        }
    }

    _producer = new ProducerBuilder<string, string>(pp)
        .SetOAuthBearerTokenRefreshHandler(OauthCallback)
        .Build();
}
```

```
public ProducerService()
{
        void OauthCallback(IClient client, string cfg)
        {
            try
            {
                var token = retrieveToken(config[“RefreshToken"]);
                client.OAuthBearerSetToken(token.TokenValue, token.Expiration, token.Principal);
            }
            catch (Exception e)
            {
                client.OAuthBearerSetTokenFailure(e.ToString());
            }
        }

        _producer = new ProducerBuilder<string, string>(pp)
            .SetOAuthBearerTokenRefreshHandler(OauthCallback)
            .Build();
}
```

```
public ProducerService()
{
        void OauthCallback(IClient client, string cfg)
        {
            try
            {
                var token = retrieveToken(config[“RefreshToken"]);
                client.OAuthBearerSetToken(token.TokenValue, token.Expiration, token.Principal);
            }
            catch (Exception e)
            {
                client.OAuthBearerSetTokenFailure(e.ToString());
            }
        }

        _producer = new ProducerBuilder<string, string>(pp)
            .SetOAuthBearerTokenRefreshHandler(OauthCallback)
            .Build();
 }
```

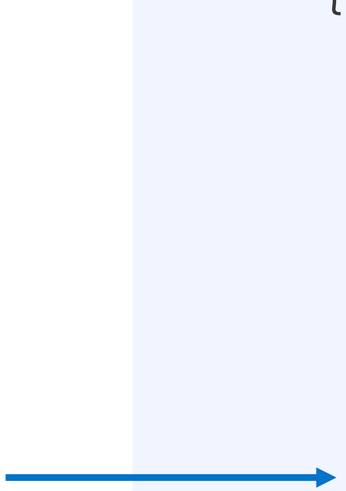
```
public ProducerService()
{
        void OauthCallback(IClient client, string cfg)
        {
            try
            {
                var token = retrieveToken(config[“RefreshToken"]);
                client.OAuthBearerSetToken(token.TokenValue, token.Expiration, token.Principal);
            }
            catch (Exception e)
            {
                client.OAuthBearerSetTokenFailure(e.ToString());
            }
        }

        _producer = new ProducerBuilder<string, string>(pp)
            .SetOAuthBearerTokenRefreshHandler(OauthCallback)
            .Build();
 }
```

```
public ProducerService()
{
        void OauthCallback(IClient client, string cfg)
        {
            try
            {
                var token = retrieveToken(config[“RefreshToken"]);
                client.OAuthBearerSetToken(token.TokenValue, token.Expiration, token.Principal);
            }
            catch (Exception e)
            {
                client.OAuthBearerSetTokenFailure(e.ToString());
            }
        }

        _producer = new ProducerBuilder<string, string>(pp)
            .SetOAuthBearerTokenRefreshHandler(OauthCallback)
            .Build();
}
```

Баги библиотеки librdkafka (set oauth token)

```
return new OAuthBearerToken
{
    TokenValue = token?.AccessToken,
    Expiration = expiresAt.ToUnixTimeMilliseconds(),

    //          To support confluent-kafka-dotnet
    //          public static void OAuthBearerSetToken(this IClient client, string tokenValue, long lifetimeMs,
    //                                                 string principalName, IDictionary<string, string> extensions = null)
    //          {
    //              client.Handle.LibrdkafkaHandle.OAuthBearerSetToken(tokenValue, lifetimeMs, principalName,
                                                                                                    extensions);
    //          }
    //          principalName = null works on windows, but suddenly crashes on unix systems
    //          use principalName = empty string
    //          Issue: https://github.com/confluentinc/confluent-kafka-dotnet/issues/2158

    Principal = string.Empty
};
```

Баги библиотеки librdkafka (set oauth token)

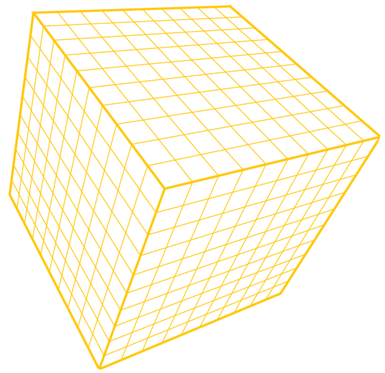


```
void OauthCallback(IClient client, string cfg)
        {
            try
            {
                var config = configuration.GetSection("Kafka:OAuthSettings");
                var token = retrieveToken(config[“RefreshToken"]);
                client.OAuthBearerSetToken(token.TokenValue, token.Expiration, "");
            }
            catch (Exception e)
            {
                client.OAuthBearerSetTokenFailure(e.ToString());
            }
        }
```

Publish

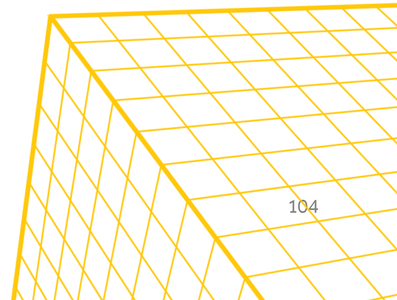
ProduceAsync() – асинхронный с ожиданием, отправка событий по одному

Produce() – публикует батч без явного ожидания, необходимо реализовывать колбеки для того, чтобы отловить было ли успешно отправлено событие



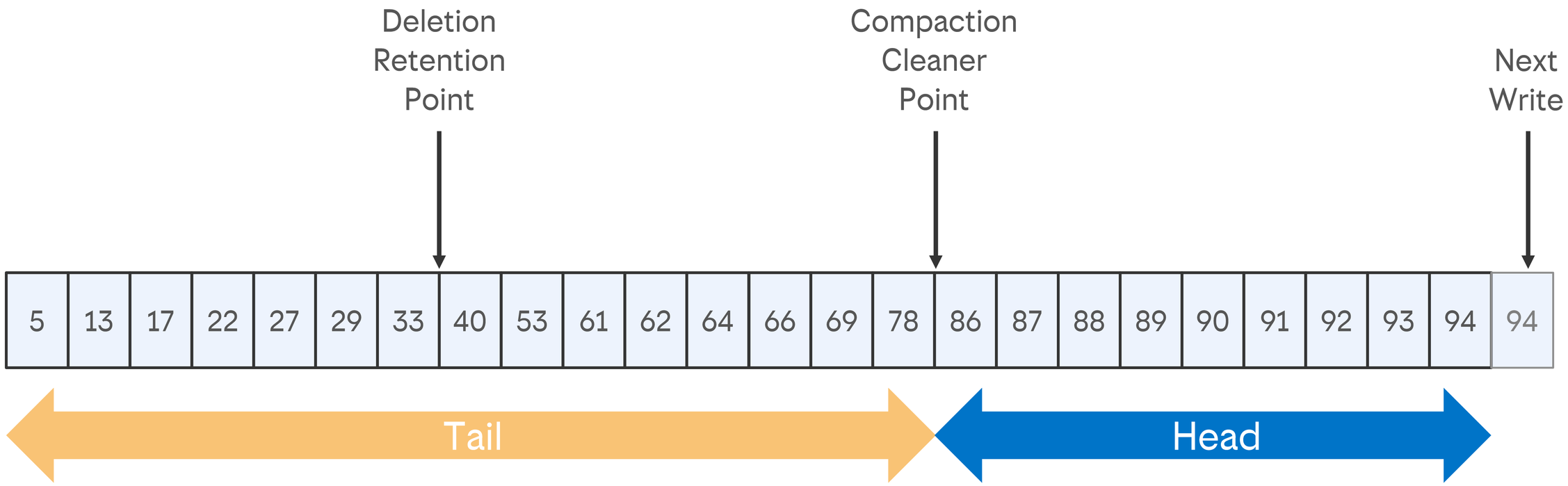
```
producer.Produce(topic,
    kafkaMessage,
    (deliveryReport) =>
    {
        if (deliveryReport.Error.Code != ErrorCode.NoError)
        {
            _logger.Error(“Delivery failed”);
        }
});
```

```
producer.ProduceAsync(topic, kafkaMessage);
```

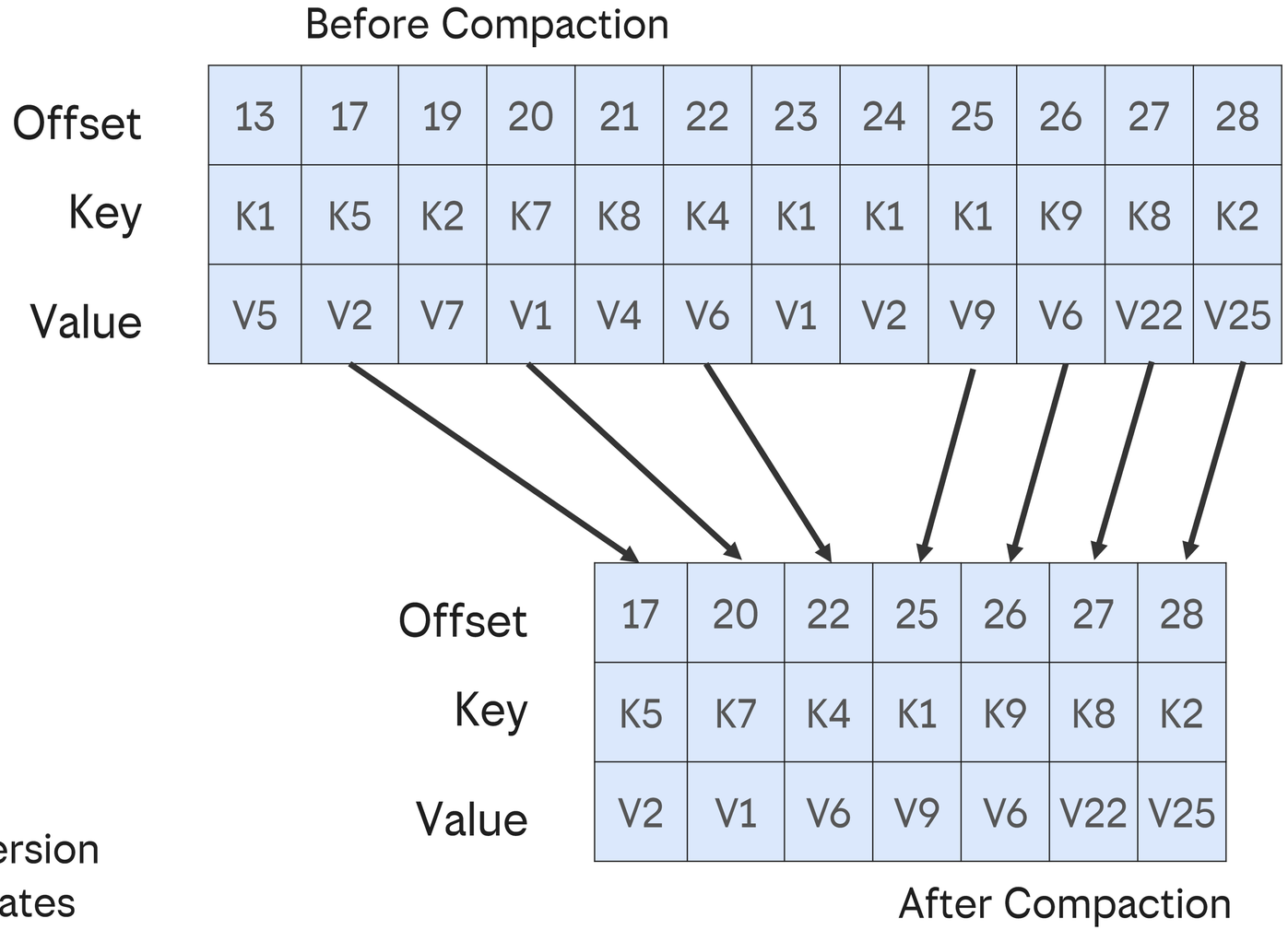


Log Compaction

Kafka Log Compaction Structure



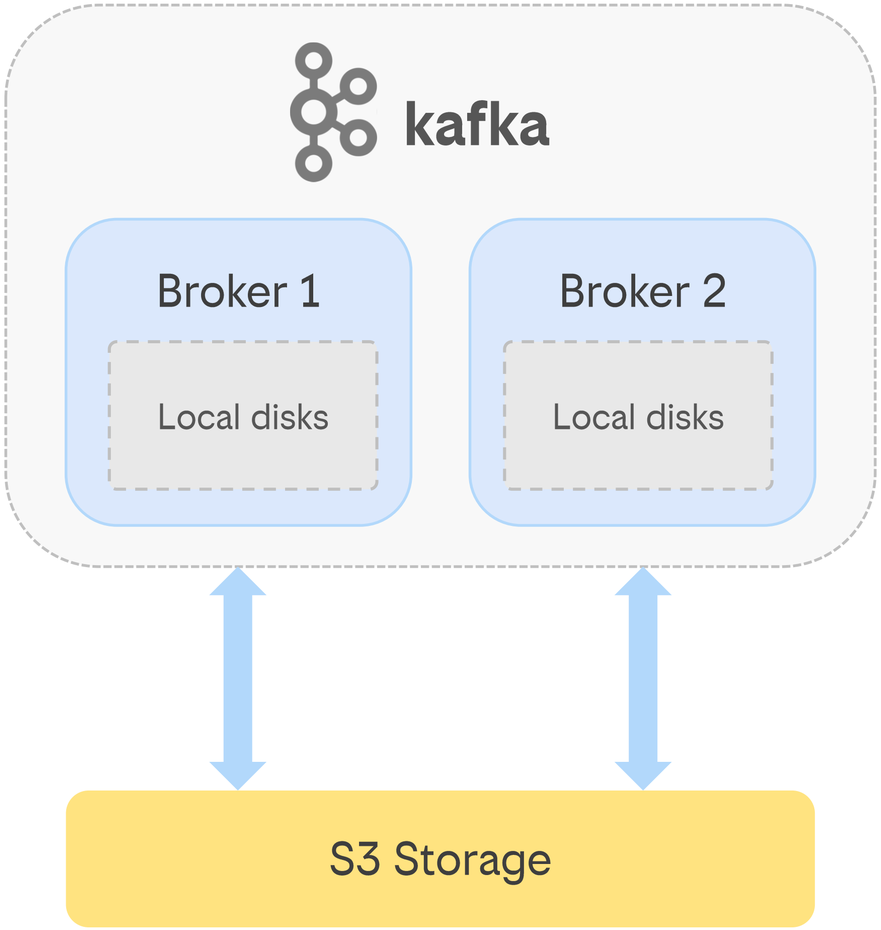
Log Compaction



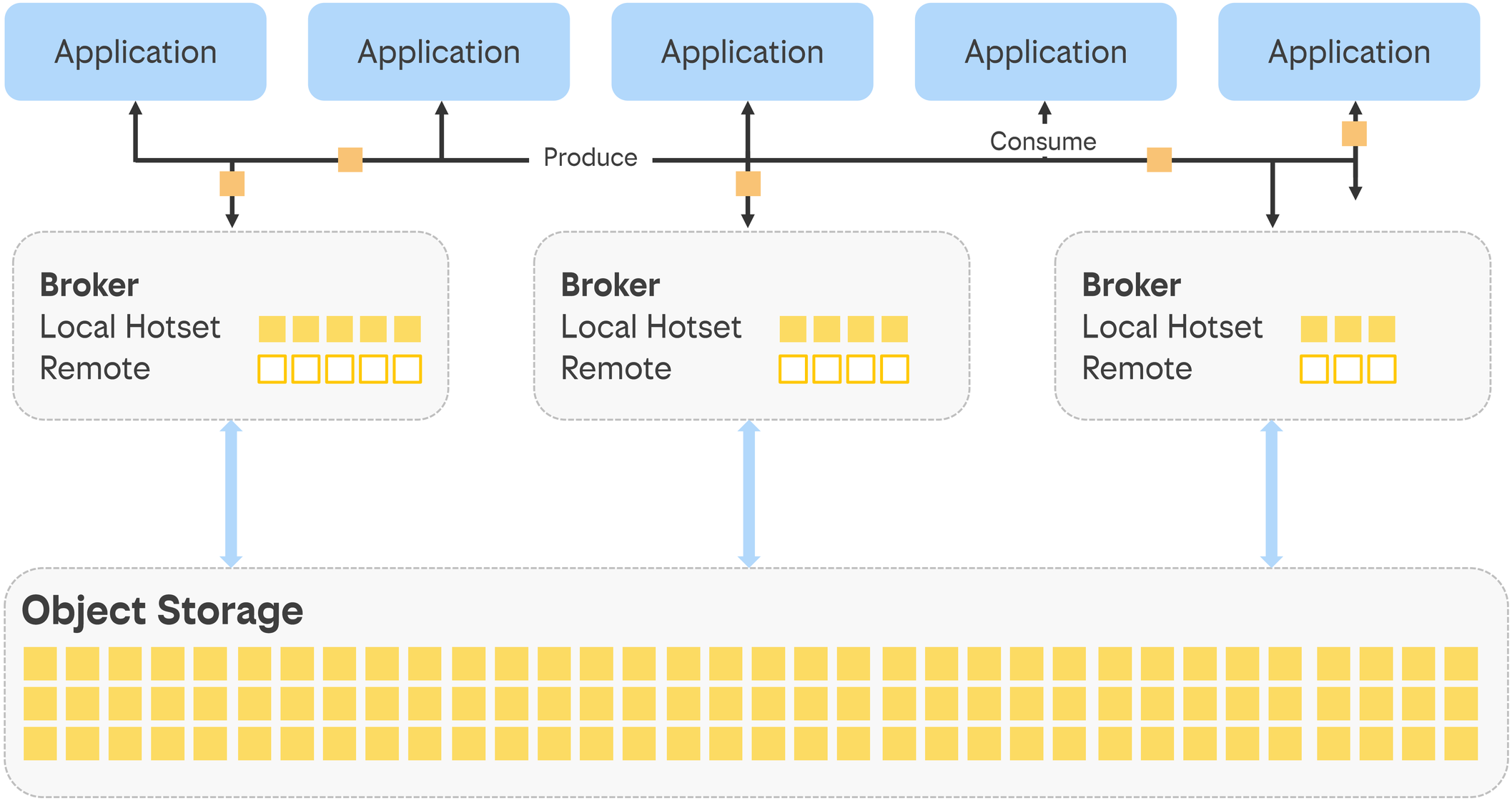
Cleaning

Only keeps latest version of key. Older duplicates not needed.

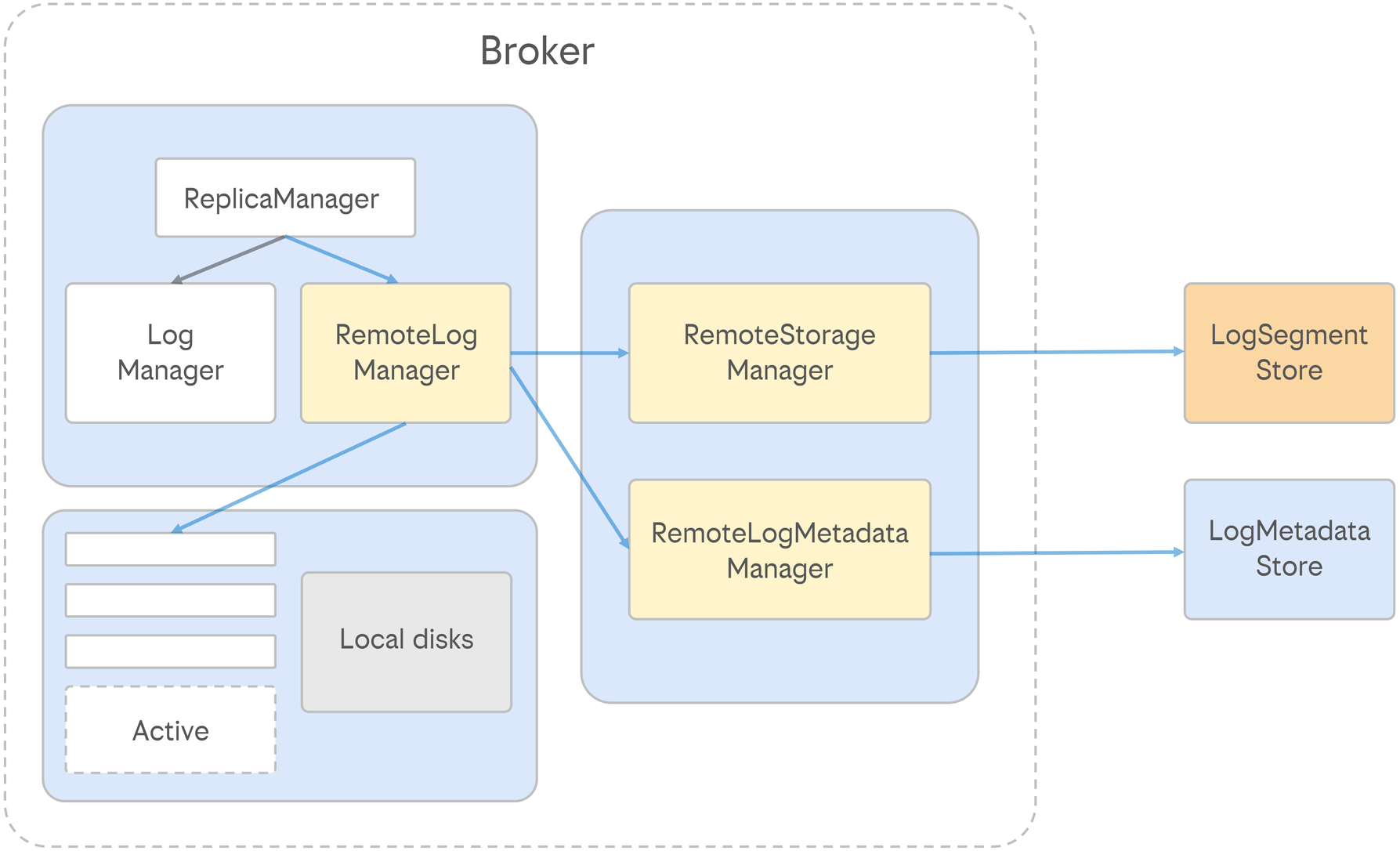
Многоуровневое хранилище (tiered storage)



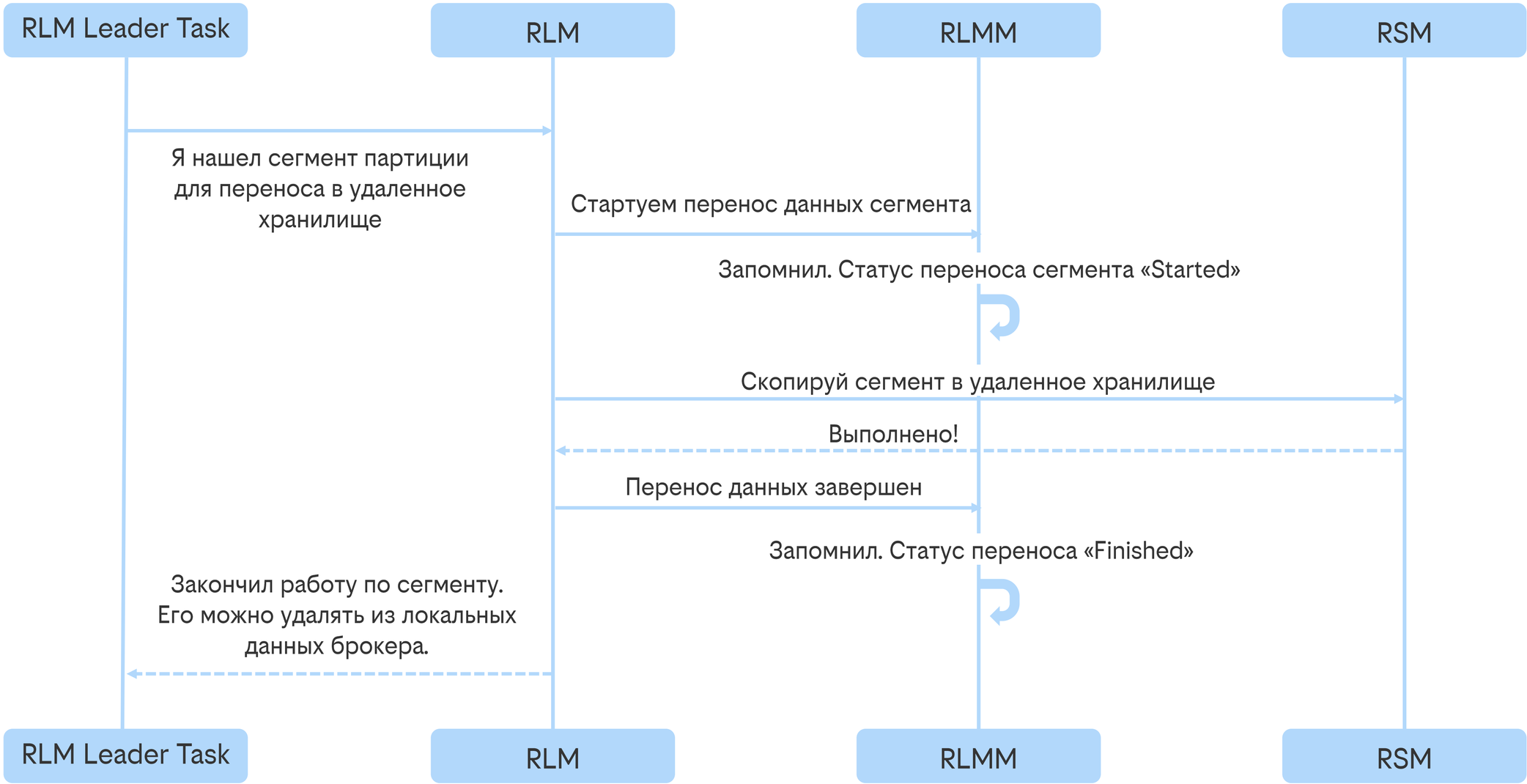
Многоуровневое хранилище (tiered storage)



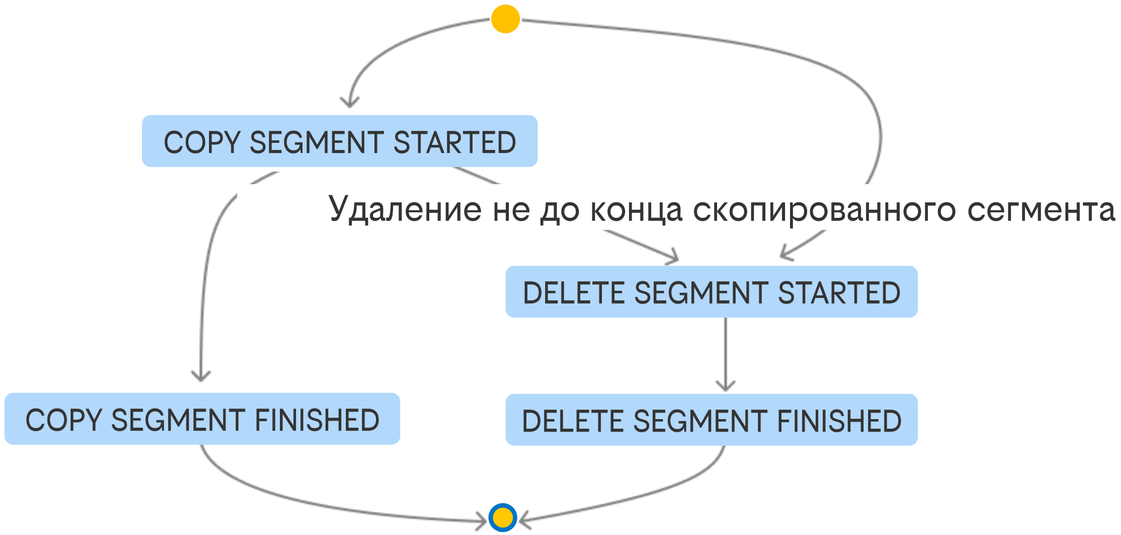
Многоуровневое хранилище (tiered storage)

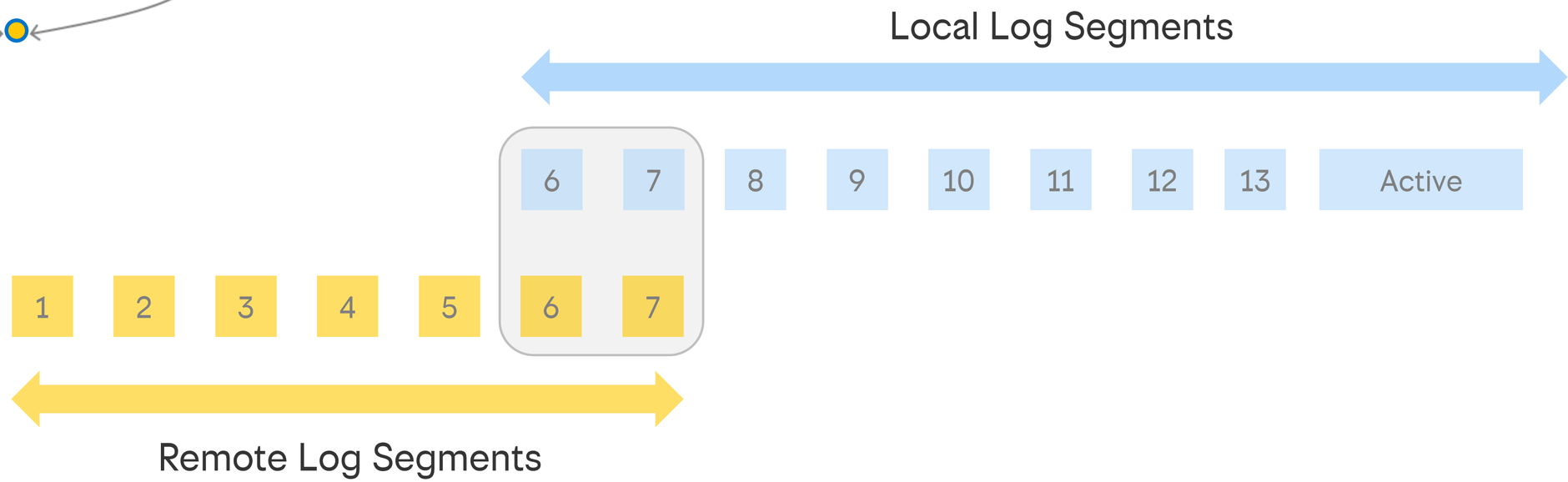


Многоуровневое хранилище (tiered storage)



Многоуровневое хранилище (tiered storage)

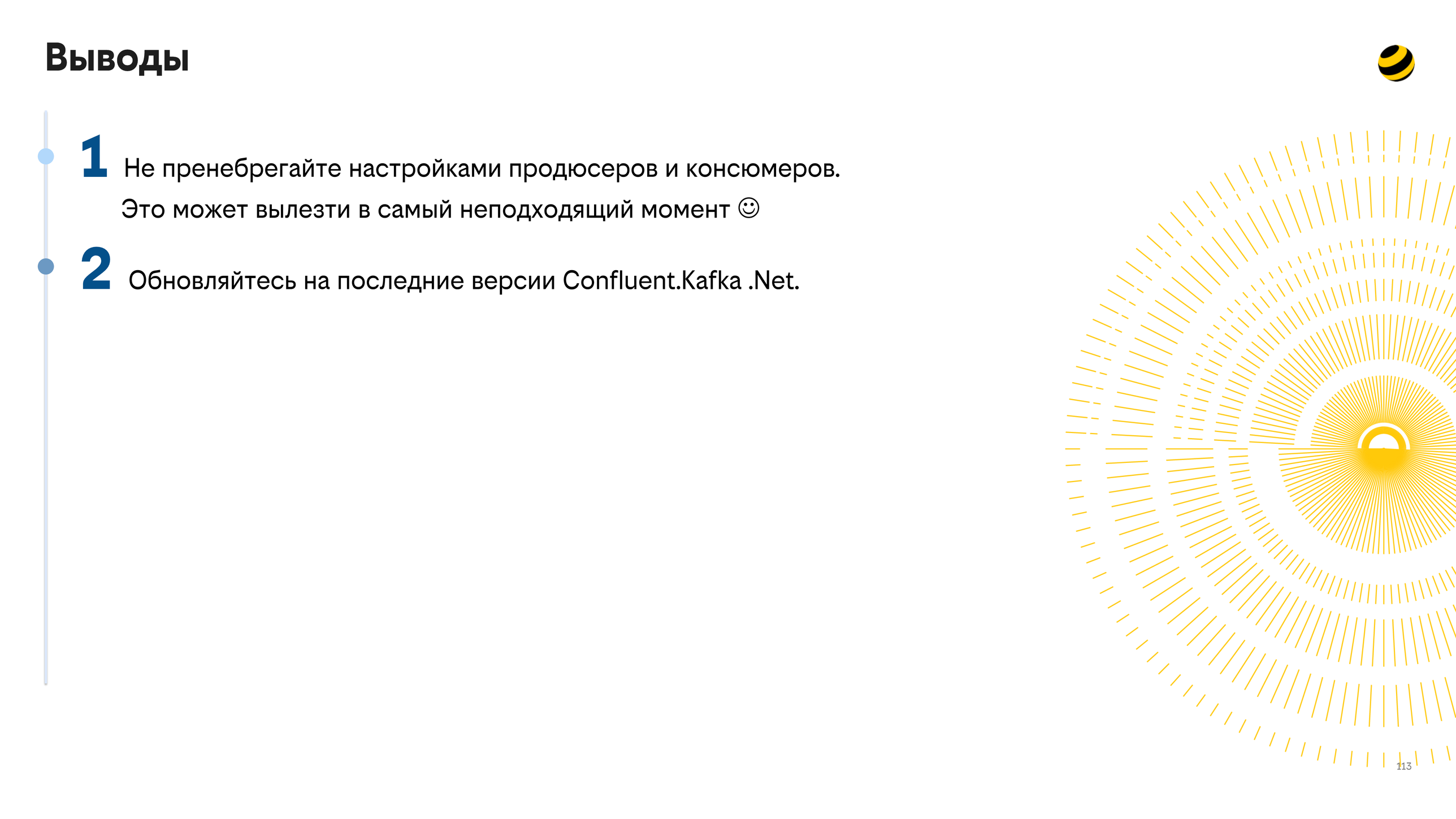




Выводы

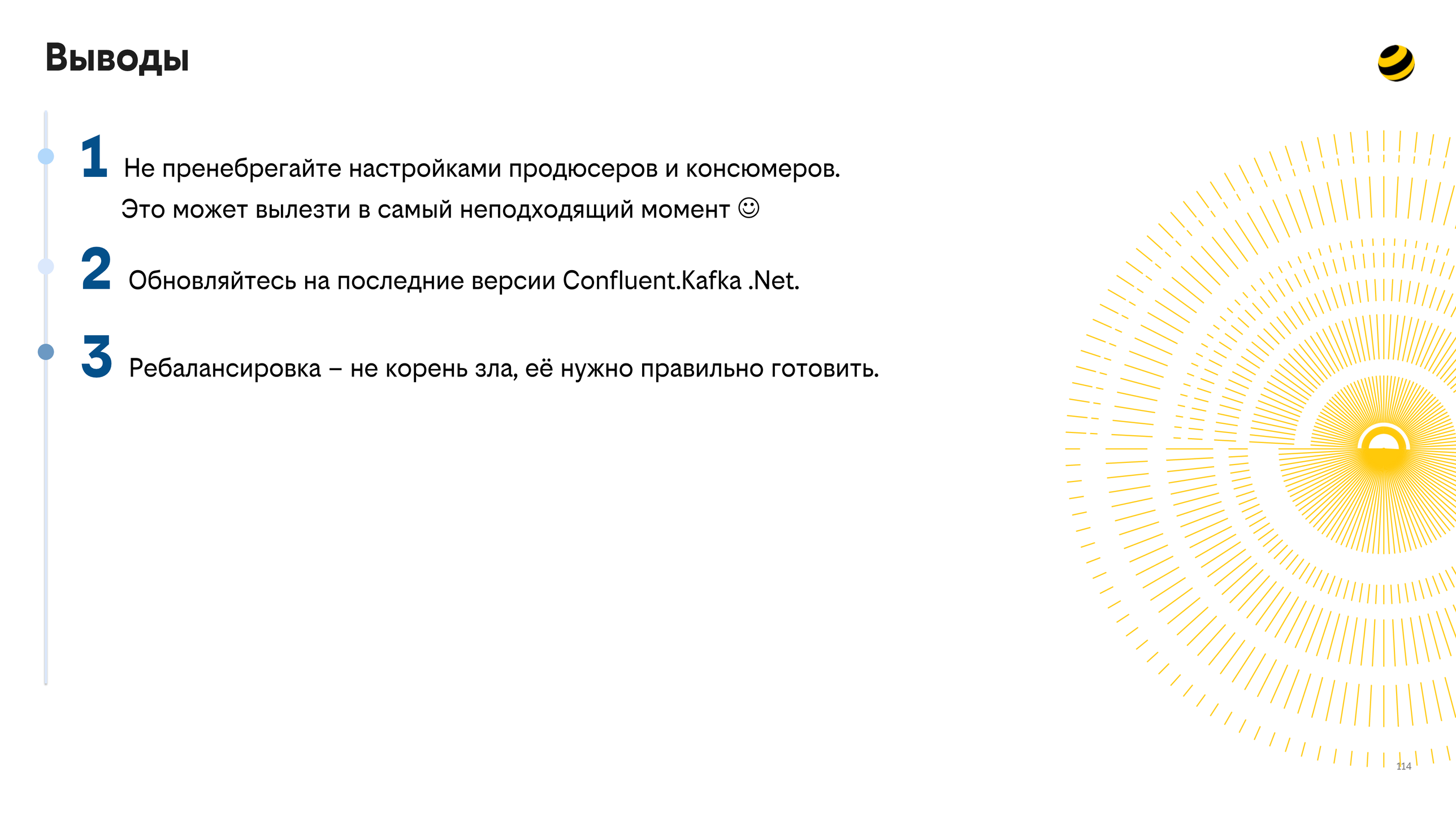
1. Не пренебрегайте настройками продюсеров и консюмеров.
   Это может вылезти в самый неподходящий момент ☺

Выводы



1. Не пренебрегайте настройками продюсеров и консюмеров.
   Это может вылезти в самый неподходящий момент ☺
2. Обновляйтесь на последние версии Confluent.Kafka .Net.

Выводы

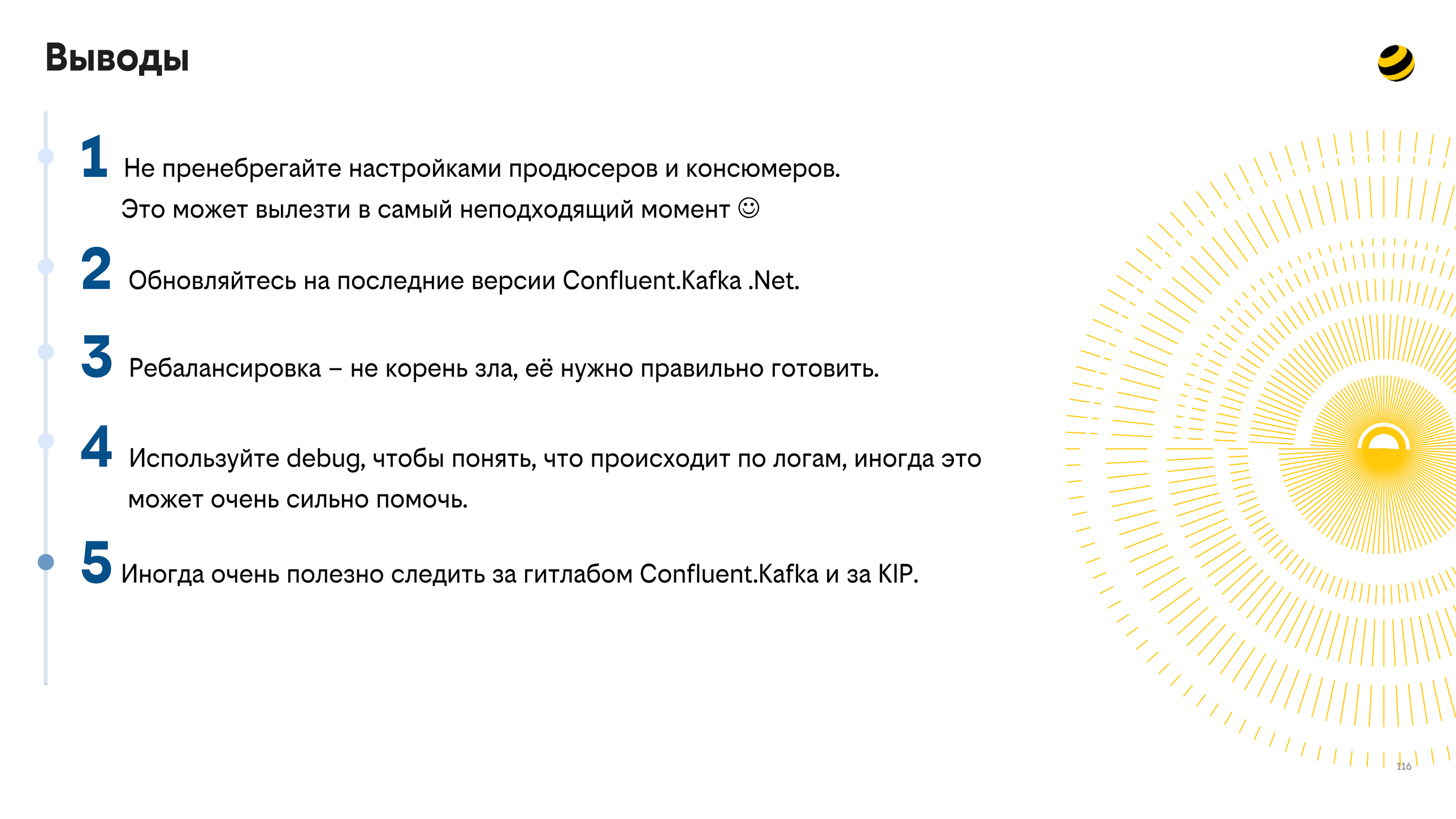


1. Не пренебрегайте настройками продюсеров и консюмеров.
   Это может вылезти в самый неподходящий момент ☺
2. Обновляйтесь на последние версии Confluent.Kafka .Net.
3. Ребалансировка – не корень зла, её нужно правильно готовить.

Выводы

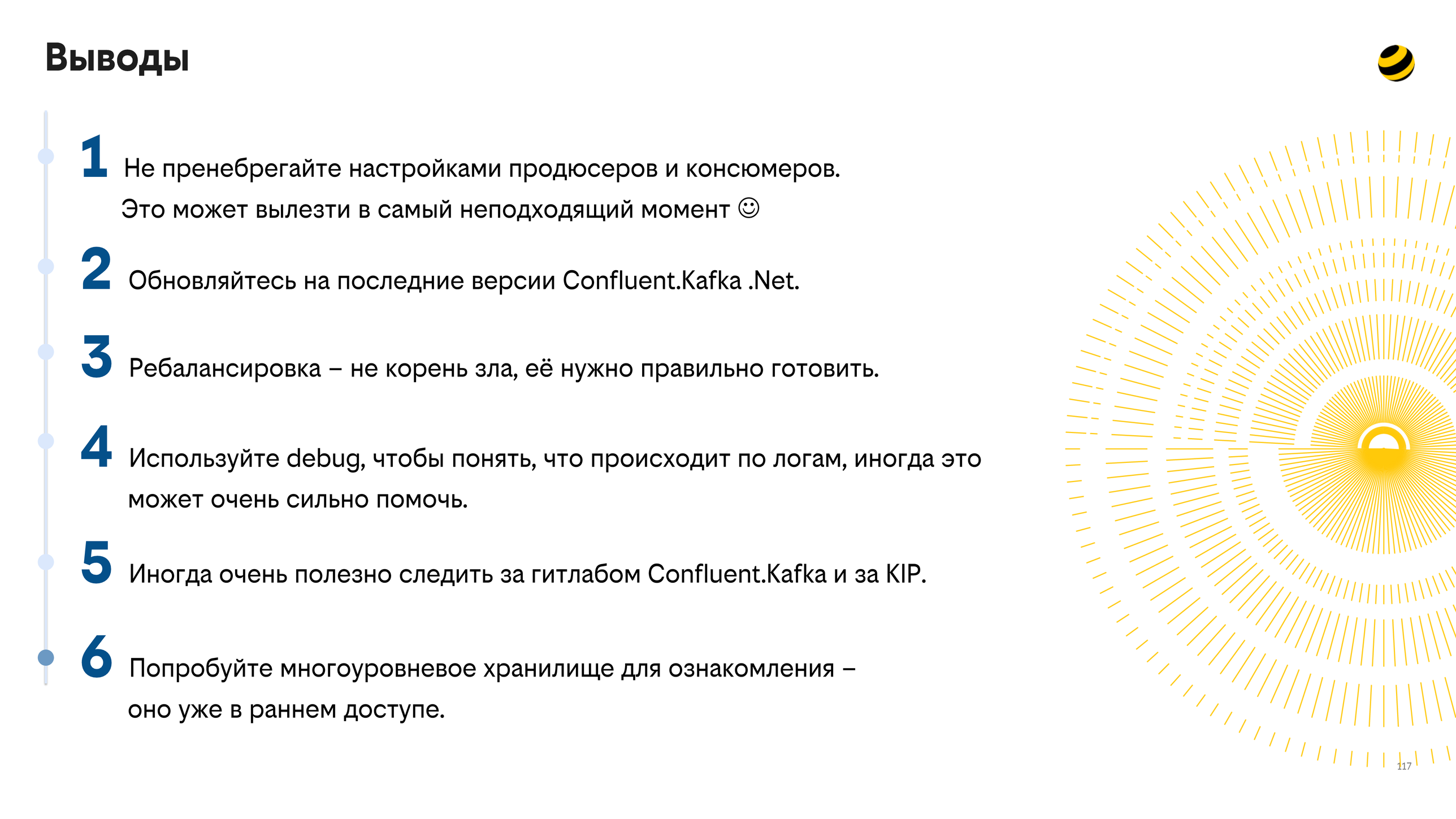
1. Не пренебрегайте настройками продюсеров и консюмеров.
   Это может вылезти в самый неподходящий момент ☺
2. Обновляйтесь на последние версии Confluent.Kafka .Net.
3. Ребалансировка – не корень зла, её нужно правильно готовить.
4. Используйте debug, чтобы понять, что происходит по логам, иногда это может очень сильно помочь.

Выводы



1. Не пренебрегайте настройками продюсеров и консюмеров.
   Это может вылезти в самый неподходящий момент ☺
2. Обновляйтесь на последние версии Confluent.Kafka .Net.
3. Ребалансировка – не корень зла, её нужно правильно готовить.
4. Используйте debug, чтобы понять, что происходит по логам, иногда это может очень сильно помочь.
5. Иногда очень полезно следить за гитлабом Confluent.Kafka и за KIP.

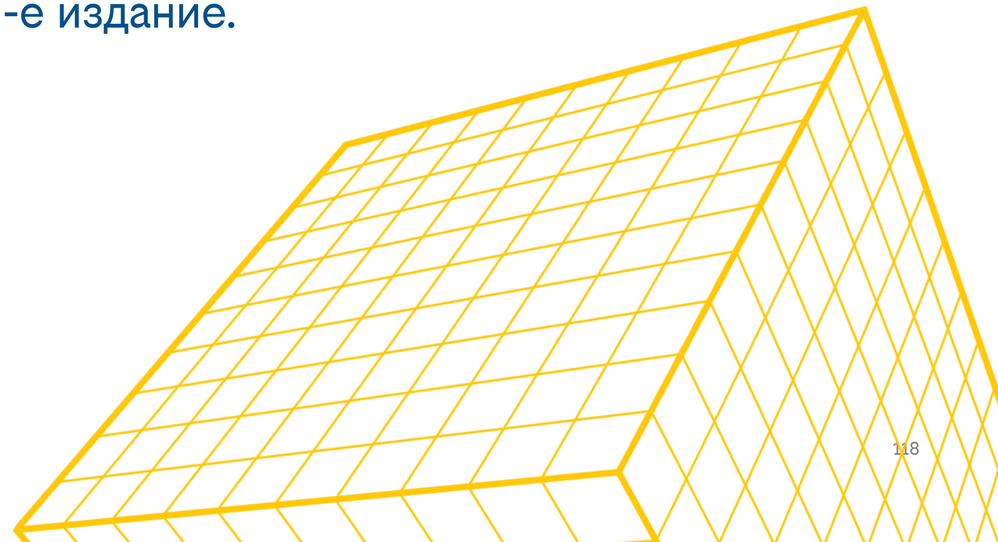
Выводы



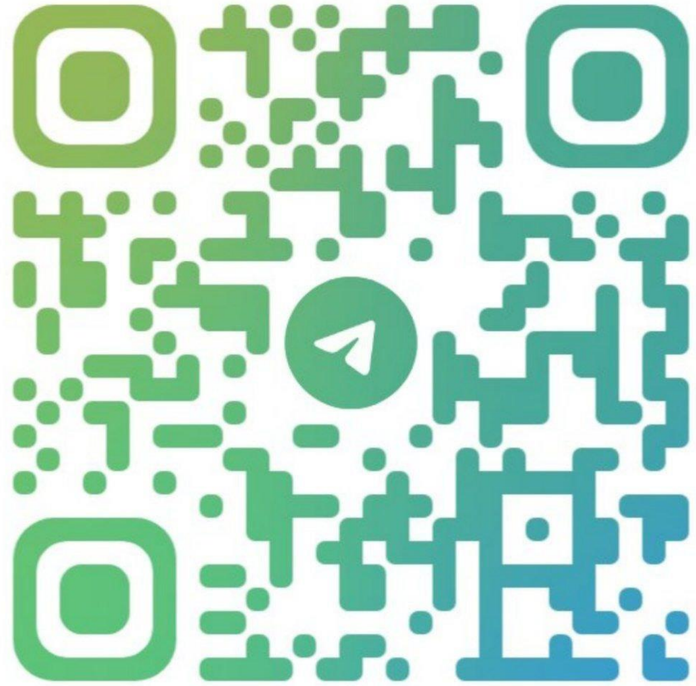
1. Не пренебрегайте настройками продюсеров и консюмеров.
   Это может вылезти в самый неподходящий момент ☺
2. Обновляйтесь на последние версии Confluent.Kafka .Net.
3. Ребалансировка – не корень зла, её нужно правильно готовить.
4. Используйте debug, чтобы понять, что происходит по логам, иногда это может очень сильно помочь.
5. Иногда очень полезно следить за гитлабом Confluent.Kafka и за KIP.
6. Попробуйте многоуровневое хранилище для ознакомления –
   оно уже в раннем доступе.

Полезные ссылки

- Григорий Кошелев — Kafka: от теории к практике
- Присматриваемся к tiered storage
- Kafka partition strategy
- KIP-405: Kafka Tiered Storage
- Kafka Topic Replication
- Kafka Producer - C# Sync vs Async
- RabbitMQ vs Kafka Part 6 - Fault Tolerance and High Availability with Kafka
- Гвен Шапира. Apache Kafka. Потоковая обработка и анализ данных. 2-е издание.
- Apache Kafka 3.1.0: что нового?

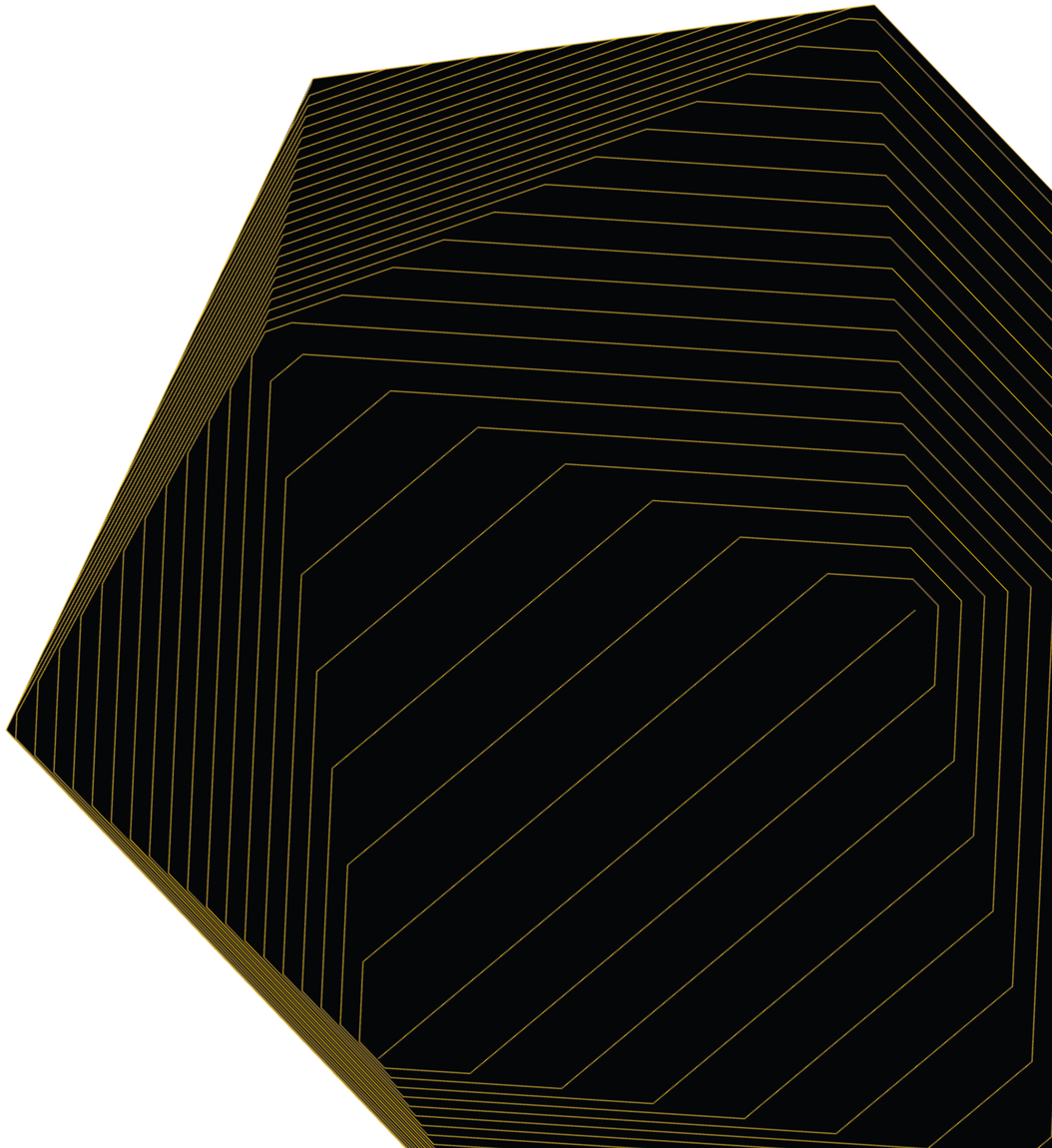


Контакты



@CHESHIRCHEG

Ваши вопросы



Спасибо!