システム間のリアルタイムデータは どう駆け抜けるのか？ ー データ連携を支える 「Apache Kafka」と「Kafka Connect」の世界

2026/5/23

OSC2026 Nagoya

株式会社SRA OSS

奥井一暁

会社紹介

株式会社SRA OSS

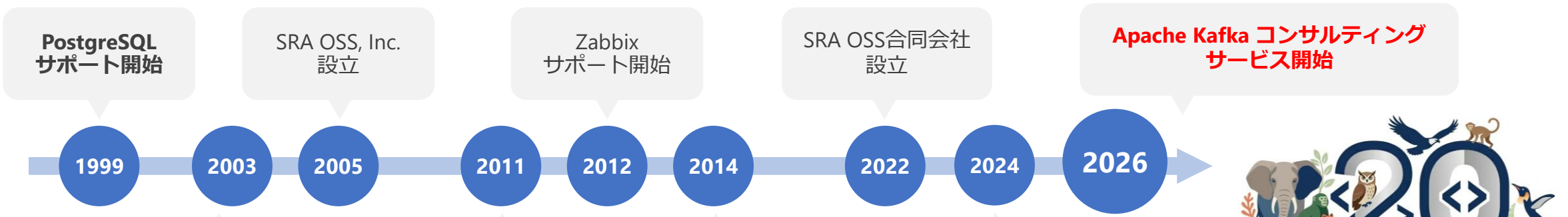
所 在 地: 東京都豊島区南池袋2-32-8
設 立 日: 2022年6月17日
株　　主: 株式会社SRA
　　株式会社NTTデータ
資 本 金: 7,000万円
社　　長: 稲葉 香理

事業内容

- オープンソースソフトウェア (OSS) 関連のサポート、製品開発・販売、構築・コンサル
- OSSの教育、開発、コミュニティ運営支援
- ソフトウェアの研究開発

顧　　問: 石井 達夫

技術顧問: 増永 良文 (お茶の水女子大学名誉教授)



おかげさまで20周年

自己紹介

- 奥井 一暁
- 株式会社SRA OSS
  OSS 事業本部 技術部 基盤技術グループ所属
- 昨年度から Kafka 担当
- 趣味: 楽器演奏

目次

- Kafka の概要
- データ構造
- データ送受信
- Kafka Connect の概要
- Kafka Connect の内部構造
- まとめ

Kafka の概要

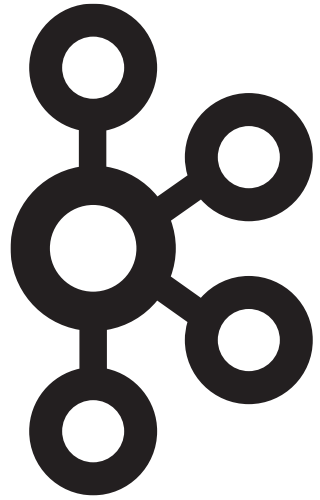
Apache Kafkaとは

- Apache Kafka（以下 Kafka）はストリーム処理を
  実現するうえで欠かせない、
  オープンソースの分散型ストリーミングシステム

- LinkedIn によって開発され、2011年オープンソース化
- 2012年にはApache Software Foundation (ASF)の
  TopLevel Project に

- 主な開発言語: Java と Scala

- 最新バージョン: 4.2.0



Apache Kafka のロゴ

2026/4/1 時点

About

Apache Kafka - A distributed event streaming platform

kafka.apache.org/

java streaming scala kafka

Readme
Apache-2.0, Apache-2.0 licenses found
Code of conduct
Contributing
Security policy
Activity
Custom properties
32.3k stars
1k watching
15.1k forks

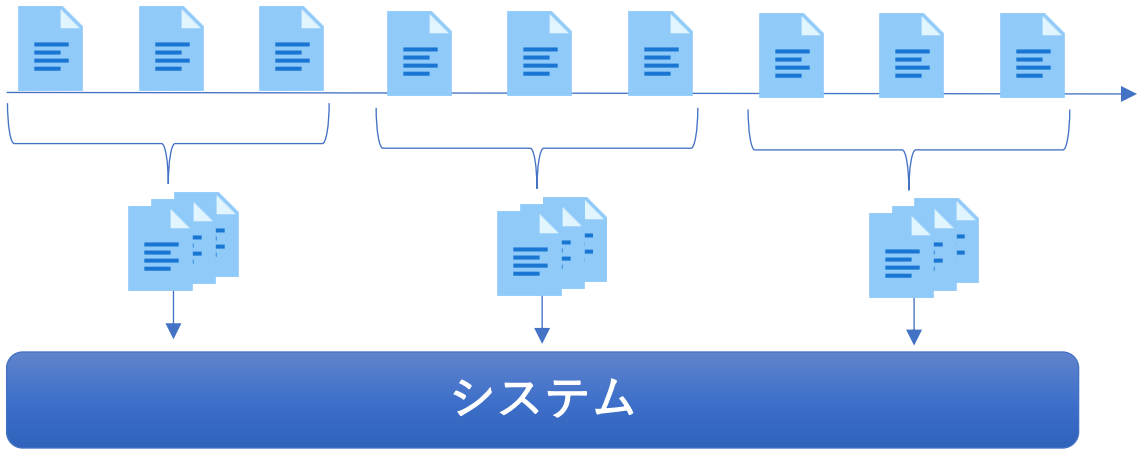
バッチ処理 VS ストリーム処理

バッチ処理

一定量のデータをまとめて一括で処理

特徴

- 大量データを効率的に処理可能
- ピーク時の負荷が高い
- リアルタイム性が低い

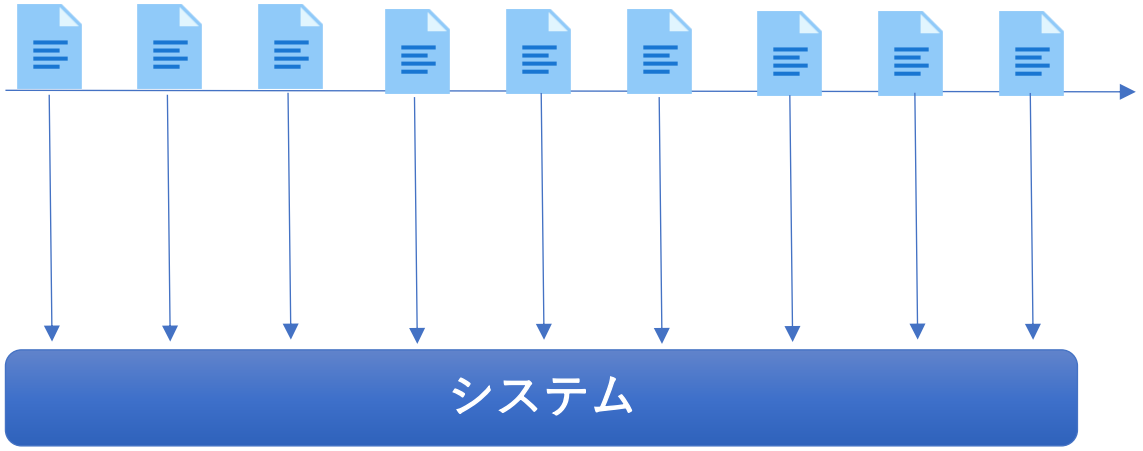


ストリーム処理 ← Kafka が活用される処理

生成され次第データを逐次処理

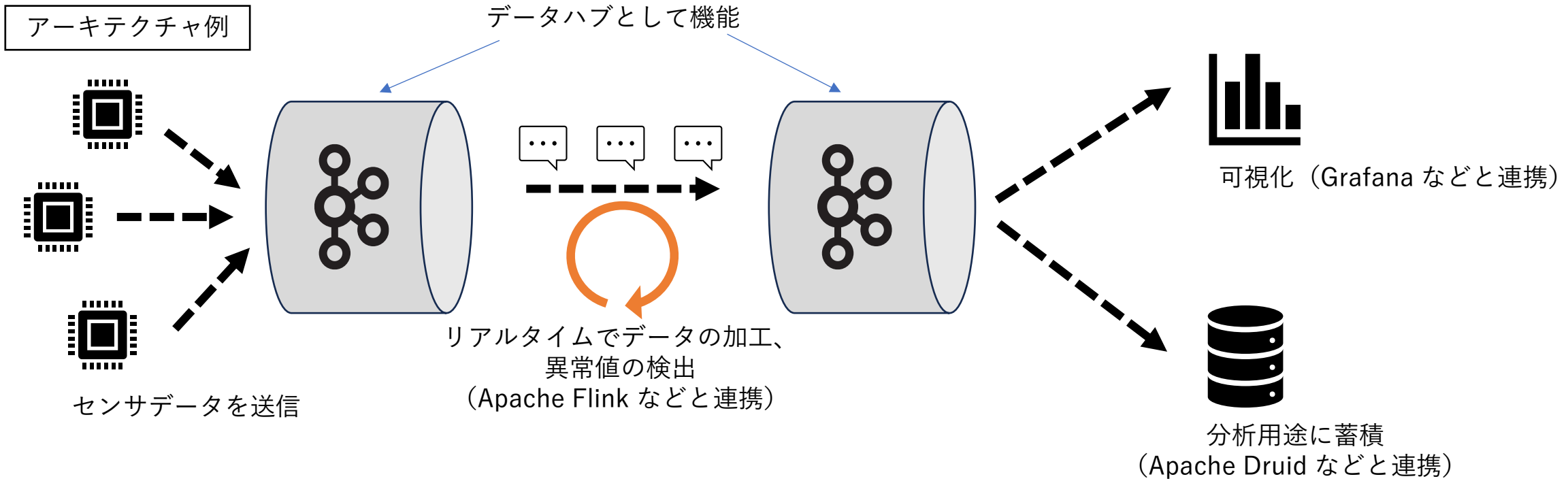
特徴

- リアルタイム性が高い
- ピーク時の負荷平滑化
- システム設計の難易度は高い



Kafka 活用例

- Kafka 活用の一例として、センサデータを収集し、異常を検知することが可能
- 収集したデータは、ほかのソフトウェアと連携することで分析にも利用可能



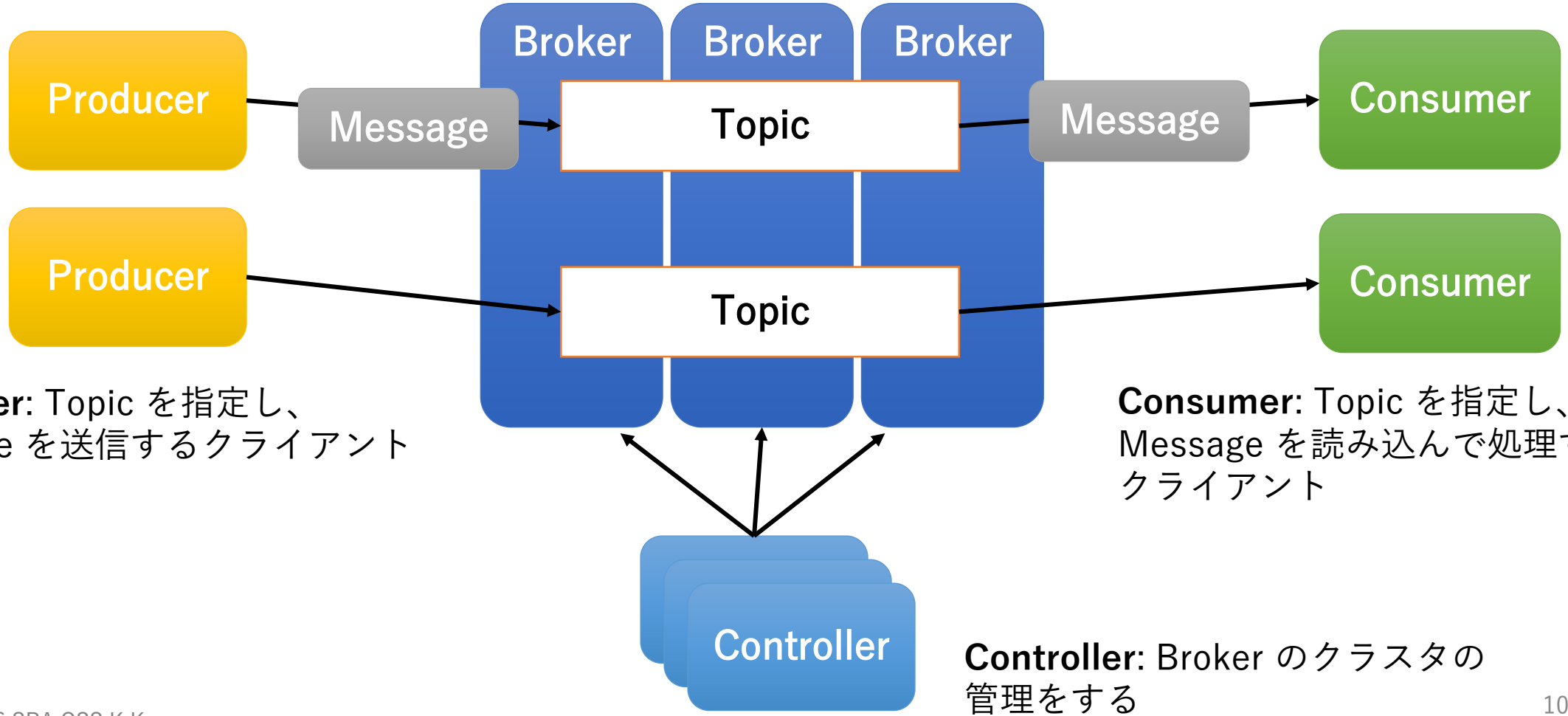
Kafkaの特徴

- **スケーラブル**
  - サーバ台数はいつでも増やすことができ、処理に応じて拡張できる
- **低レイテンシ**
  - 都度ディスクへの書き込みを行わないため、高速に処理できる
- **高可用性**
  - データは複数に複製されるため、障害時でもデータを保全する
  - 複数の送達保証レベルを実現するための機能がある
- **多様なシステム連携**
  - Kafka Connect によって、多様なシステムと連携できる

Kafka の主要なコンポーネント

Message: Kafka でやり取りされる
一つ一つのストリームデータ

Broker: Message の送受信、保存を担う
Topic: Broker 内で Message を種類ごとに管理・保存する単位



Producer: Topic を指定し、
Message を送信するクライアント

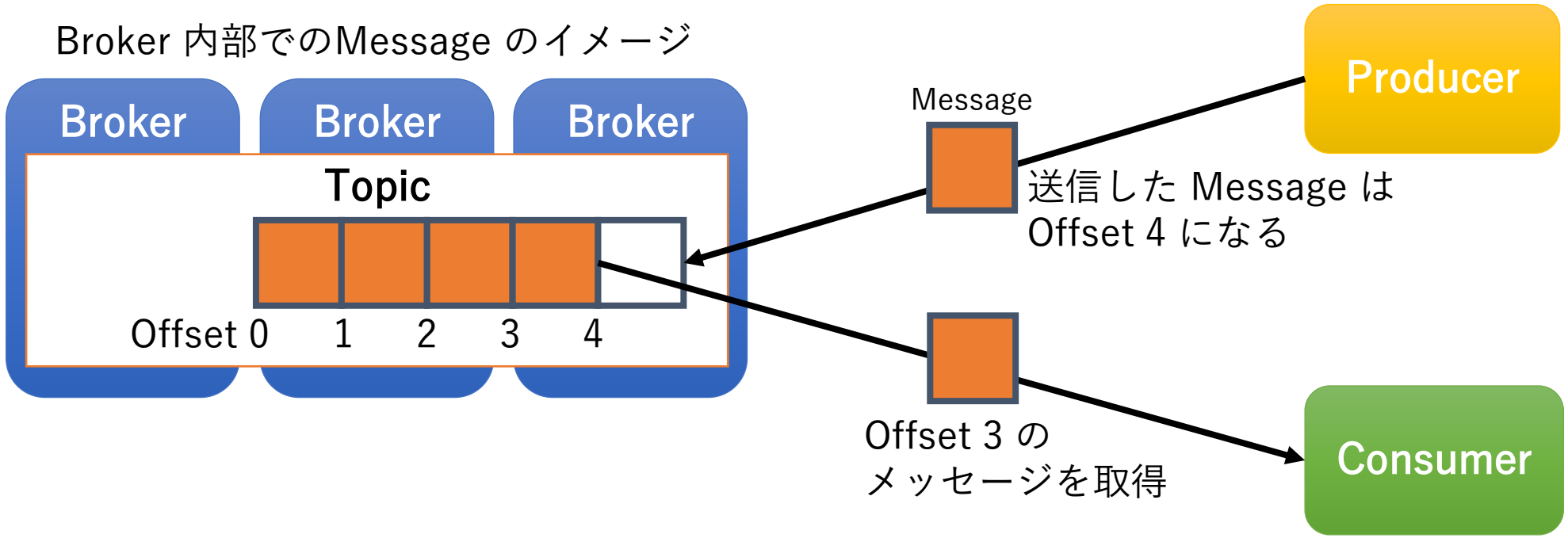
Consumer: Topic を指定し、
Message を読み込んで処理する
クライアント

Controller: Broker のクラスタの
管理をする

データ構造

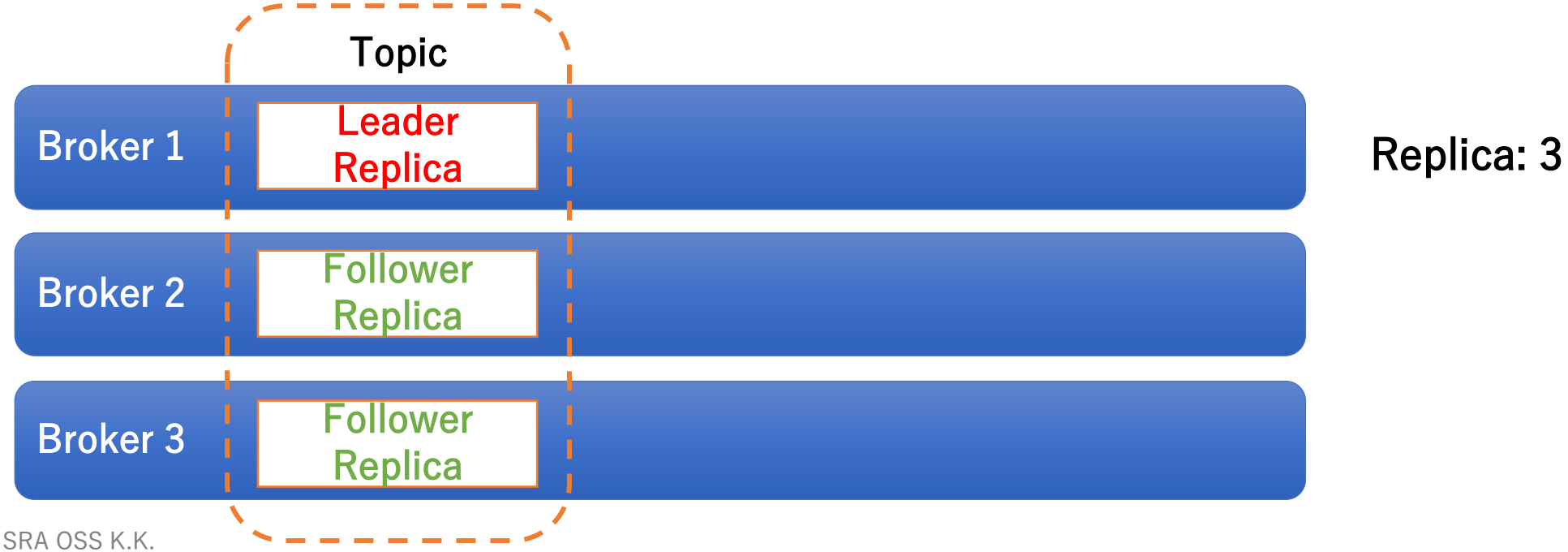
Message

- Message は Key と Value を持つ形式
- Broker は Topic に書き込まれた Message をファイルに保存し永続化



Replication

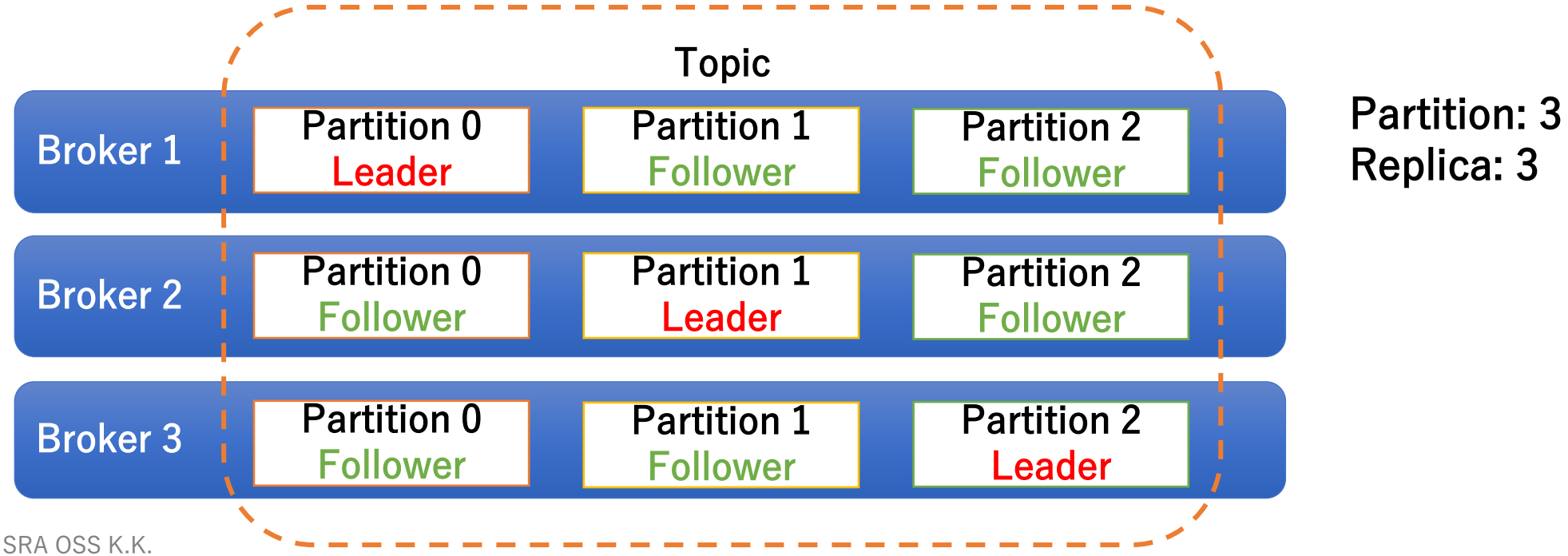
- データの複製 (Replica) を作ることで耐障害性を持たせる機能
- Leader Replica:
  - Producer、Consumer は Leader Replica を持つ Broker と Message をやり取りする
- Follower Replica:
  - Leader の複製で、障害時は Leader に昇格する



Partition

Topic を複数 Partition に分割することで処理を分散するための機能

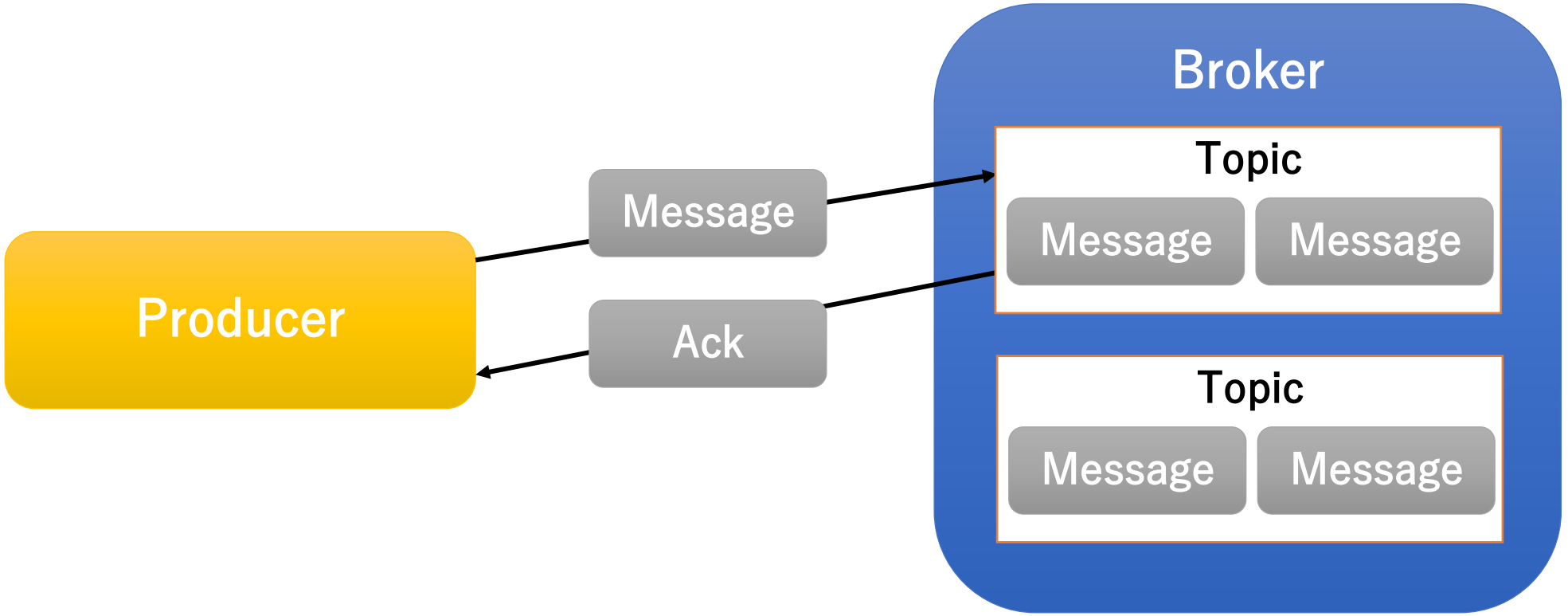
- 送信する Message は例えば Key によって
  各 Partition に分散される
- Partition ごとに Leader Replica を持つ Broker を分けることで
  Client とやり取りする Broker を分散させる



データ送受信

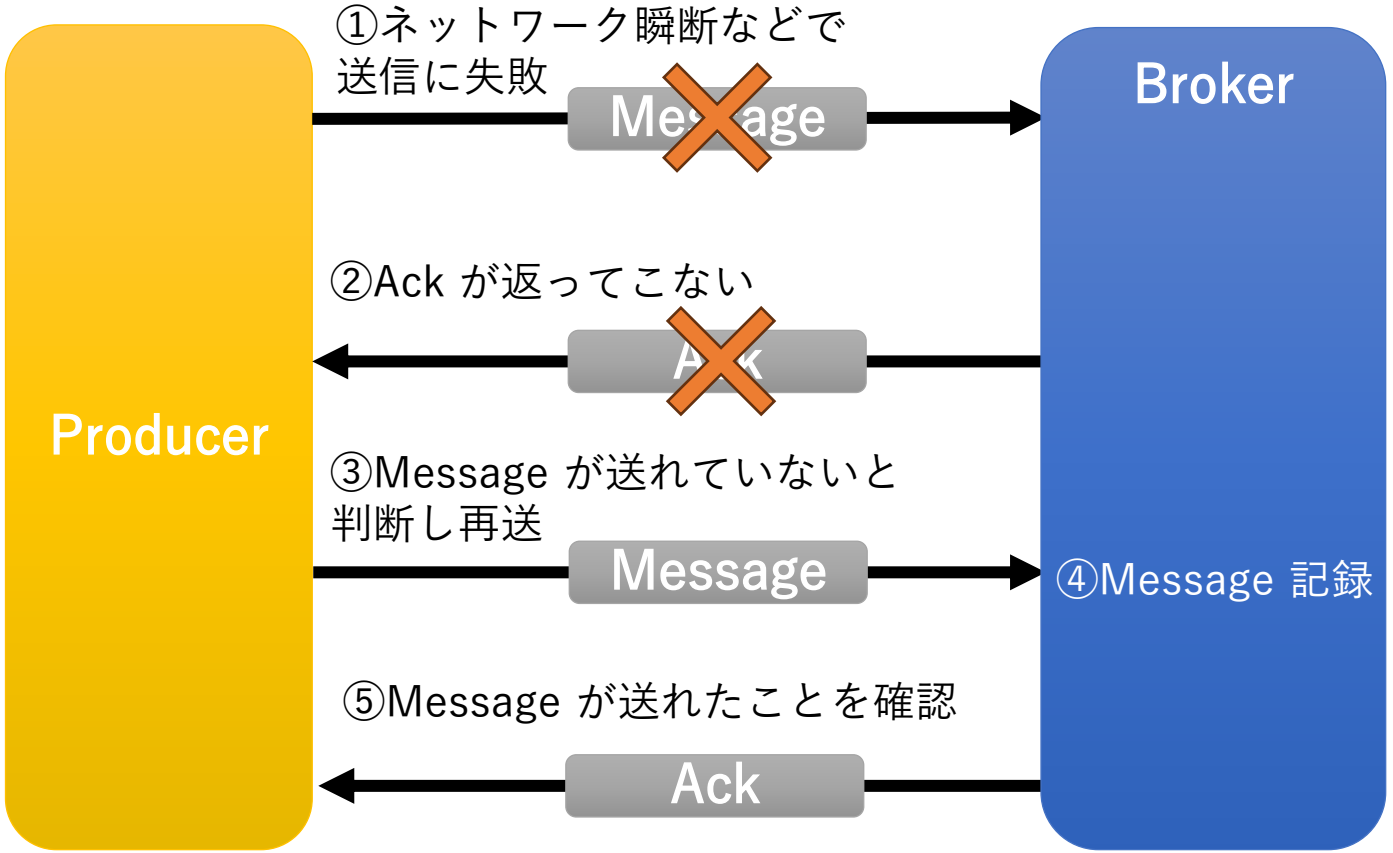
送信の流れ

- Producer は Push 型で Topic を指定して Message を送信する
- Broker は Message を受信したことをProducer に返答する (Ack)



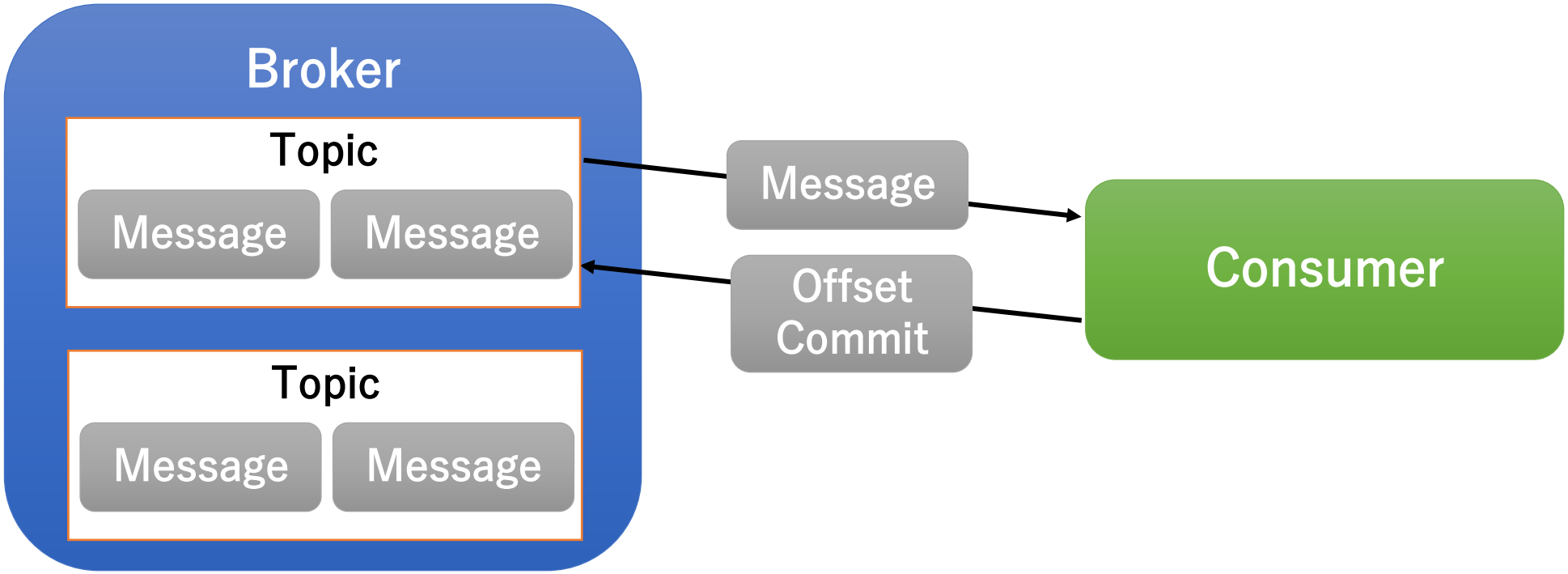
Ack

- Broker が Message を受けとったときに Producer に送られる通知
- Ack を活用することで Message を失うことなく送信が可能



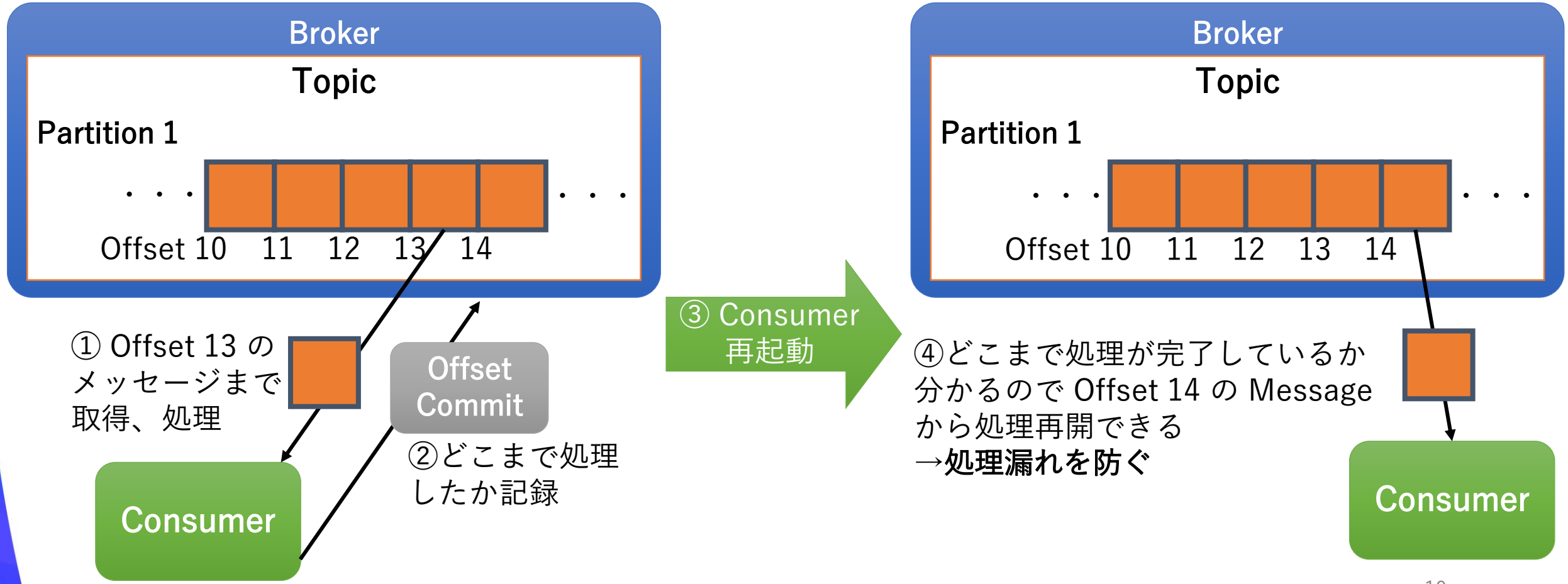
受信の流れ

- Consumer は Pull 型で Topic を指定して Message を受信する
- Consumer は Broker にどこまで読み込んだかを通知する (Offset Commit)



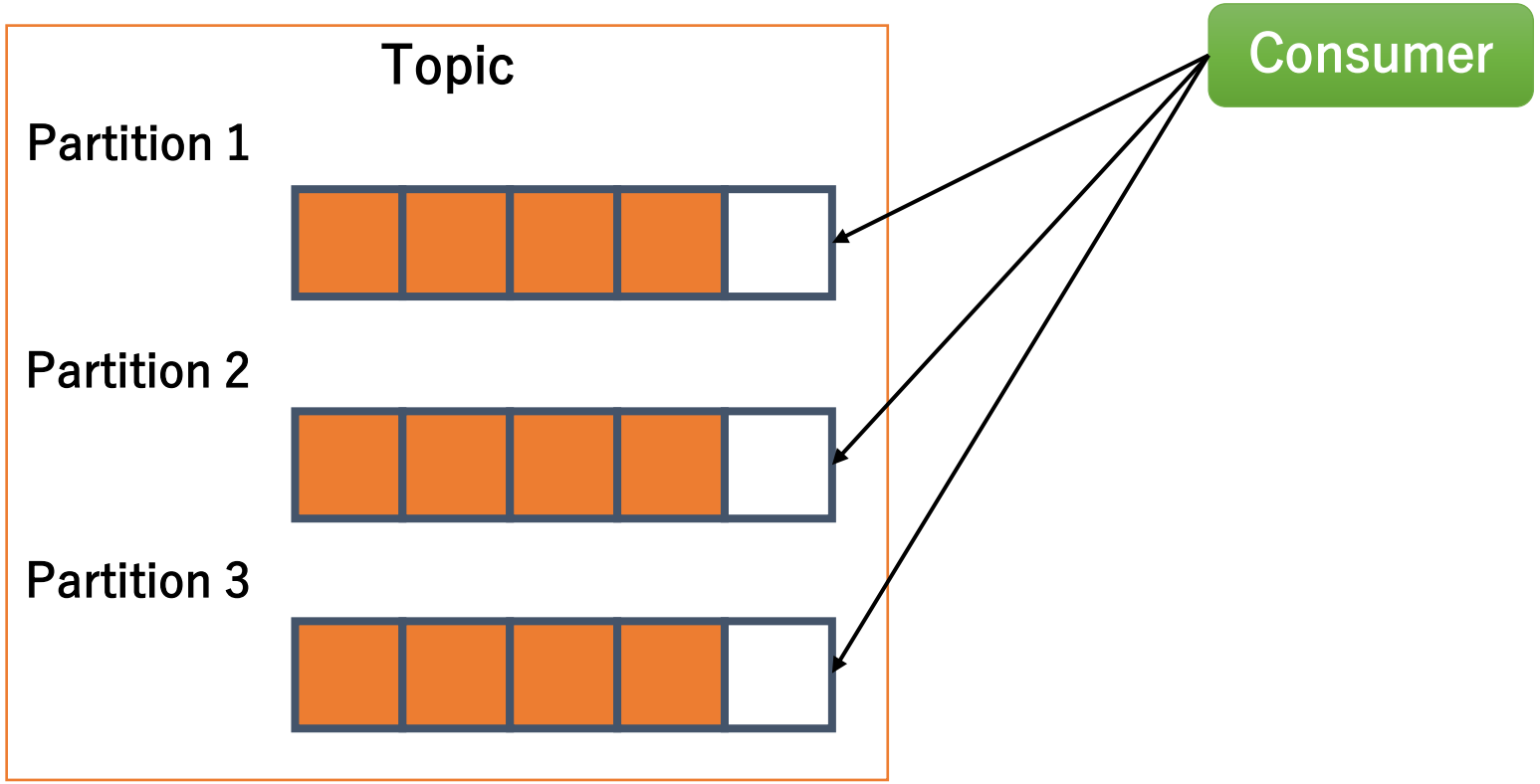
Offset Commit

- Consumer が Broker に対してどのメッセージまで処理が完了したかという記録を残すこと
  →それによって Consumer 再起動時に処理漏れを防ぐことができる

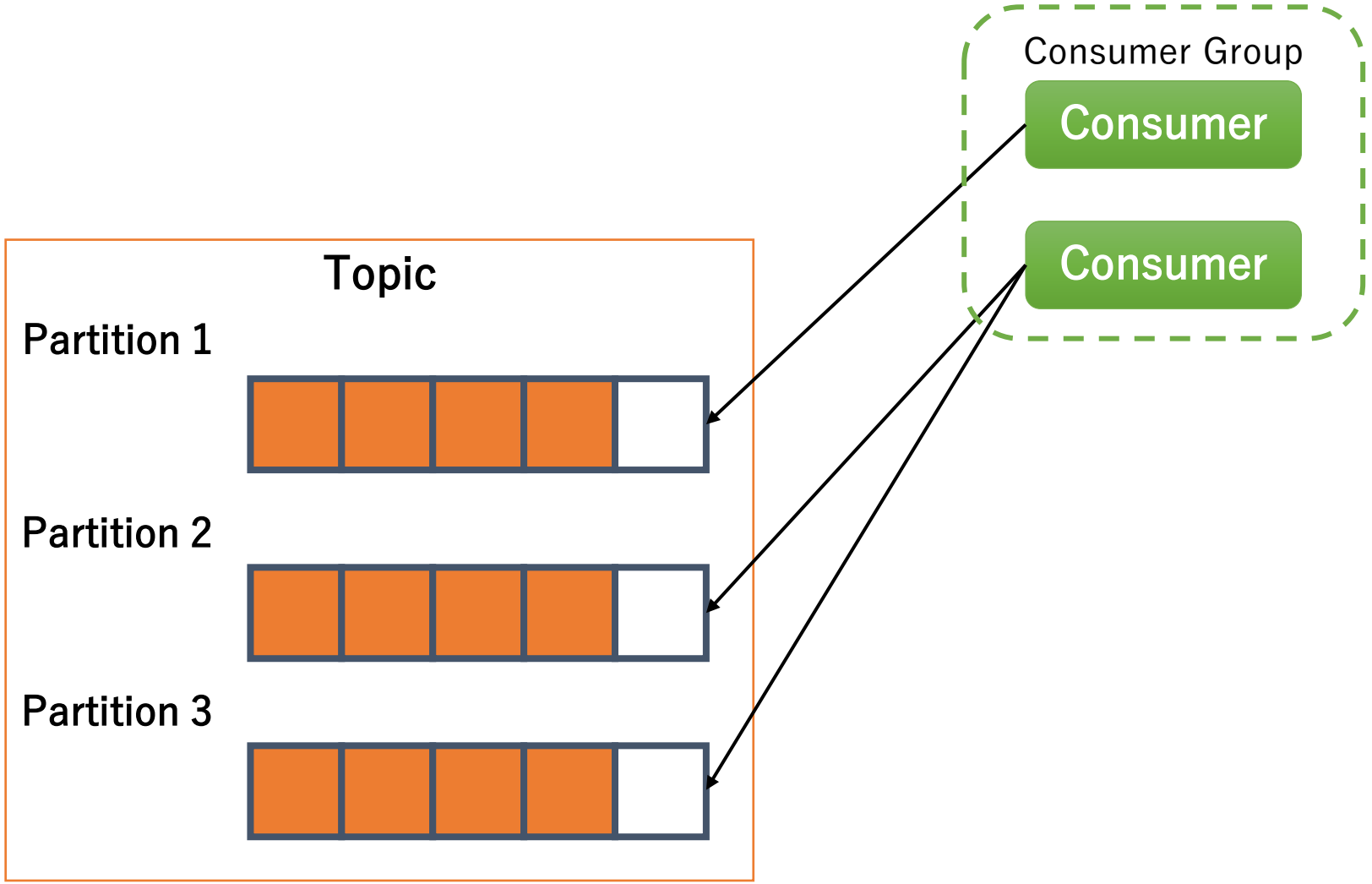


受信 Partition の自動アサイン

Consumer が1台の場合:
すべてのPartition にアサインされる

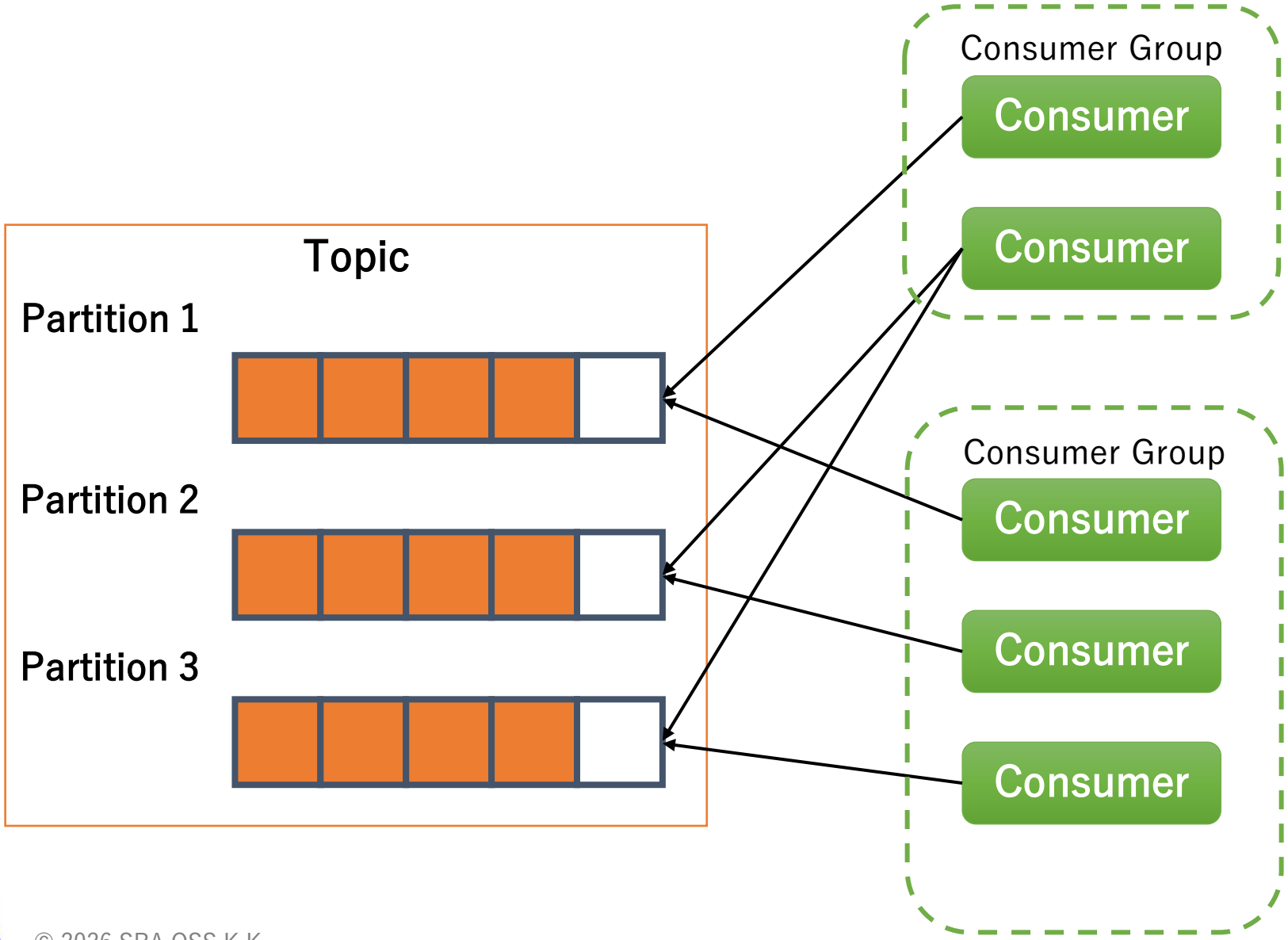


受信 Partition の自動アサイン



複数台を Consumer Group という単位でまとめると、Group 内で分担して Topic 全体の Message を受信するようにアサインされる

受信 Partition の自動アサイン



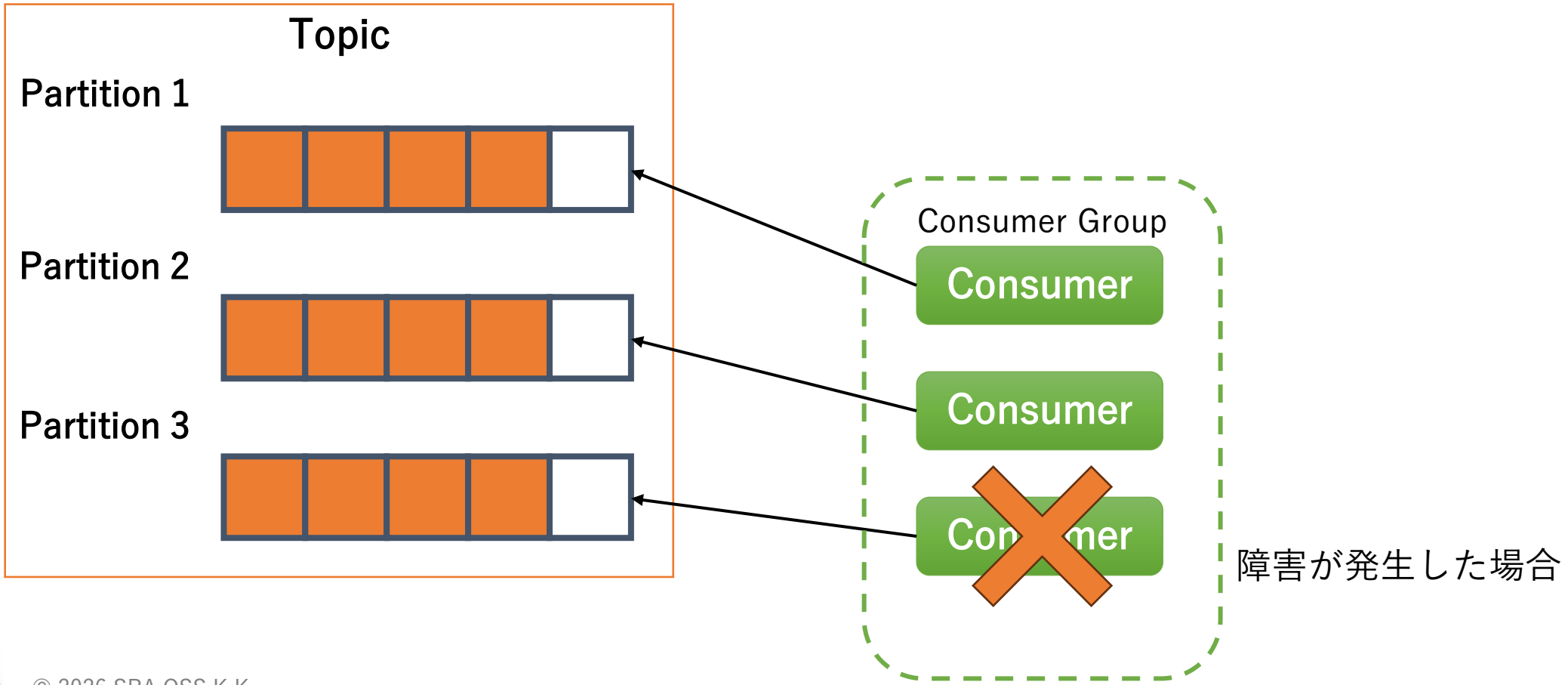
複数の Group を作ることも可能

Consumer Group の概念によって
処理を分散することができる

Consumer Group の概念は
耐障害性にも関わってくる

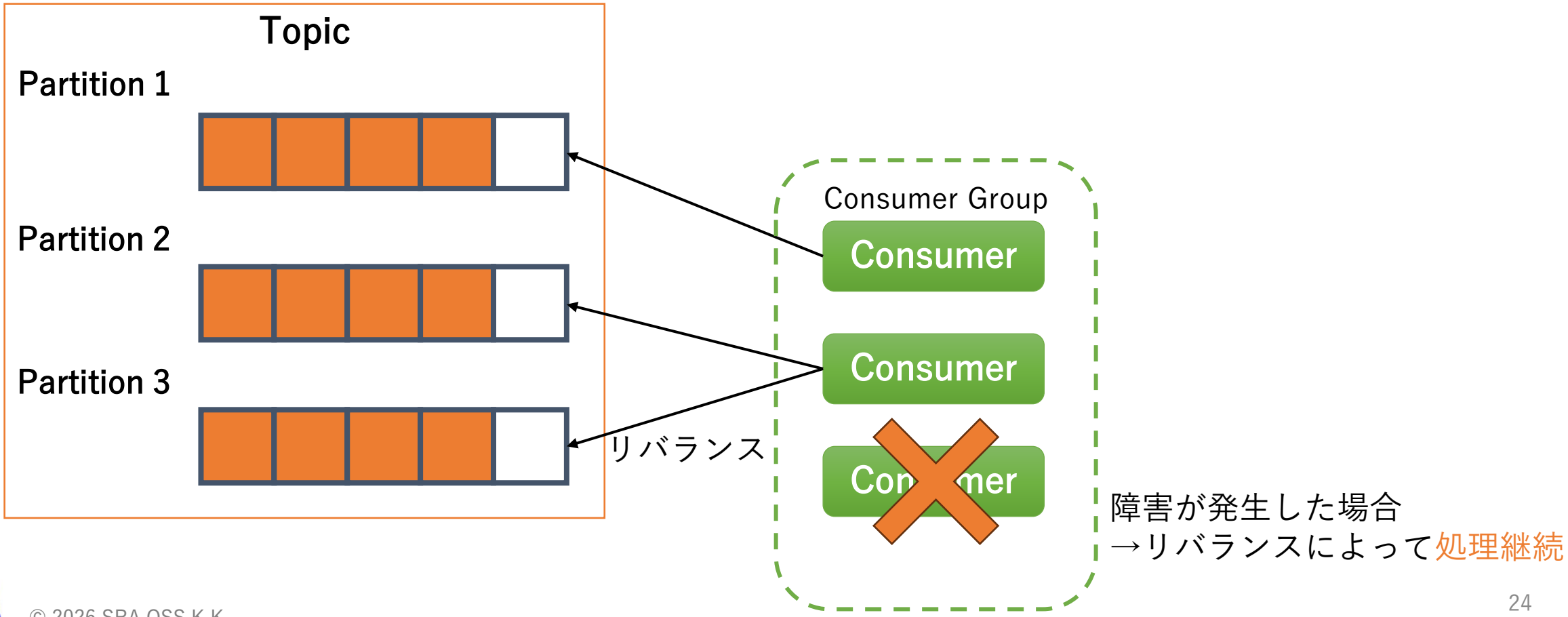
Consumer Group リバランス

- Consumer の増減があった時に Partition のアサインを再度行う機能
  - Partition を引き継ぐ際は Offset Commit の記録位置から処理を再開する



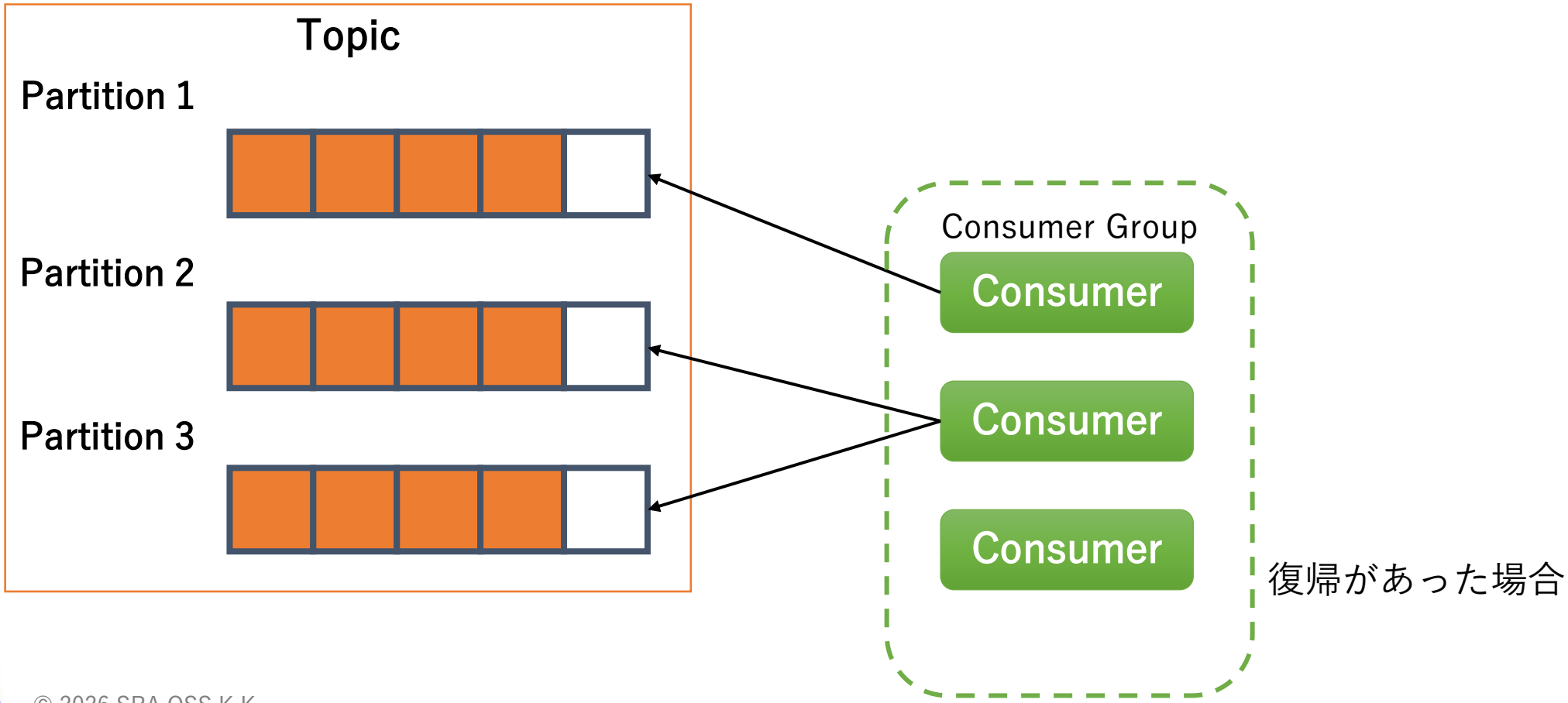
Consumer Group リバランス

- Consumer の増減があった時に Partition のアサインを再度行う機能
  - Partition を引き継ぐ際は Offset Commit の記録位置から処理を再開する



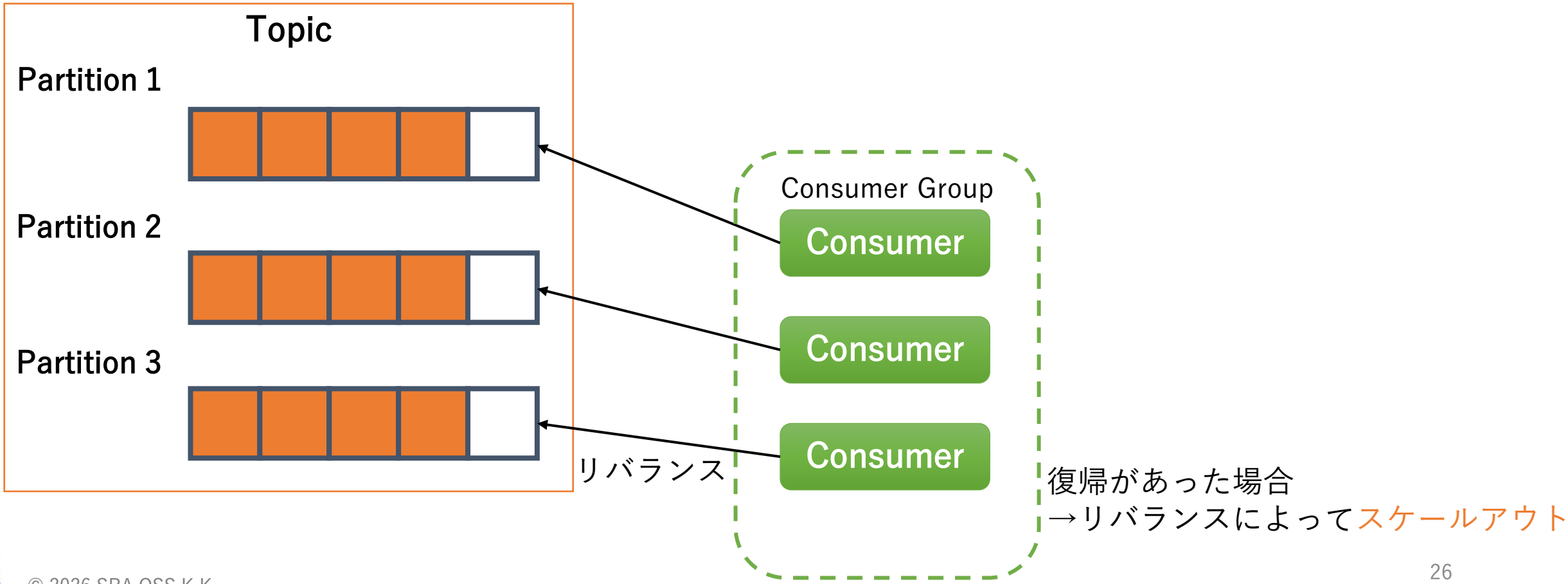
Consumer Group リバランス

- Consumer の増減があった時に Partition のアサインを再度行う機能
  - Partition を引き継ぐ際は Offset Commit の記録位置から処理を再開する



Consumer Group リバランス

- Consumer の増減があった時に Partition のアサインを再度行う機能
  - Partition を引き継ぐ際は Offset Commit の記録位置から処理を再開する



送達保証

- Kafka では、設定および実装によって以下の送達保証が実現できる

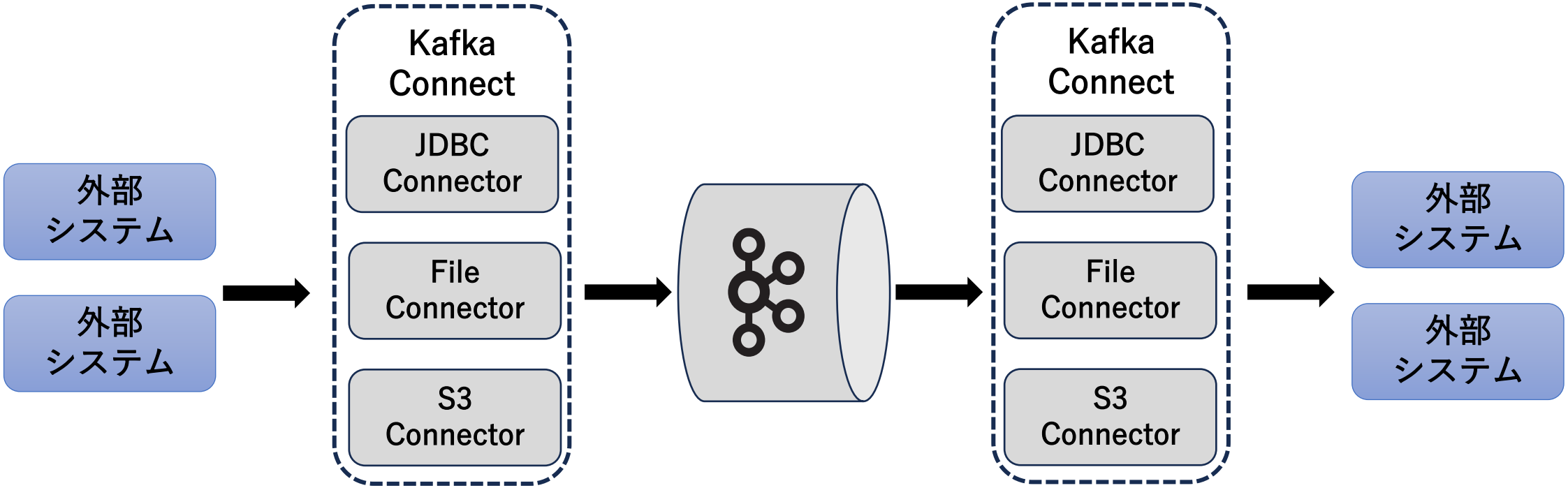
送達保証	説明
at least once	Message は失われないが、重複の可能性がある
at most once	Message は失われる可能性があるが、重複はしない
exactly once	Message は失われたり重複したりせず一度だけ配送される

- どの送達保証で実装するかはユーザ次第
  - システムの要件、許容できるパフォーマンスなどから決定

Kafka Connect の概要

Kafka Connect とは

- Kafka Connect は Kafka と外部システムのデータ連携を容易にする機能
- 通常、外部システムのデータを Kafka に転送するには、それぞれのシステムに合わせたクライアントを実装する必要があるが、Kafka Connect を用いることでクライアントを実装することなく、外部システムのデータを Kafka に流すことができる

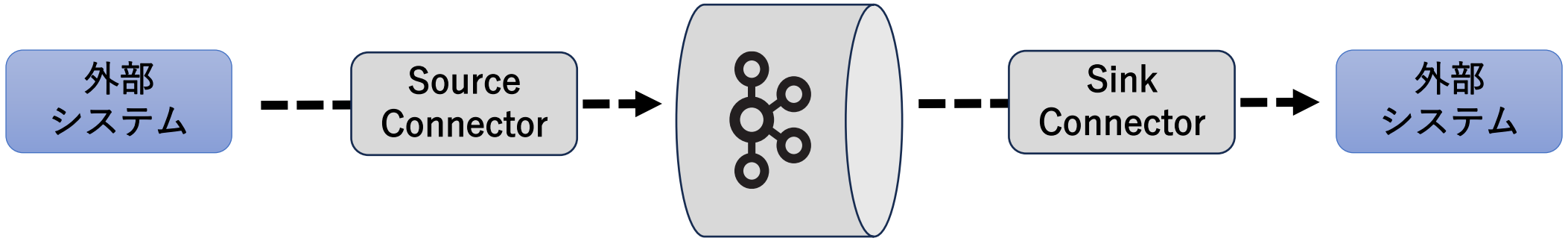


Kafka Connect の特徴

- サーバーを増やしてスケールアウト
- サーバー障害時に処理をリバランス
- 豊富な Connector が展開されている
- Transformer を用いて転送時に簡易的な加工処理も可能
- Connector によっては exactly once のサポートもある
- Connector の起動等の操作を REST API によって行うことが可能

Connector の分類

- **Source Connector**
  - 外部システムのデータを Kafka の Topic へ転送する Connector
- **Sink Connector**
  - Kafka の Topic のデータを外部システムへ転送する Connector



Kafka Connect の種類

- **Kafka Connect には多数のサードパーティ製の Connector がある**

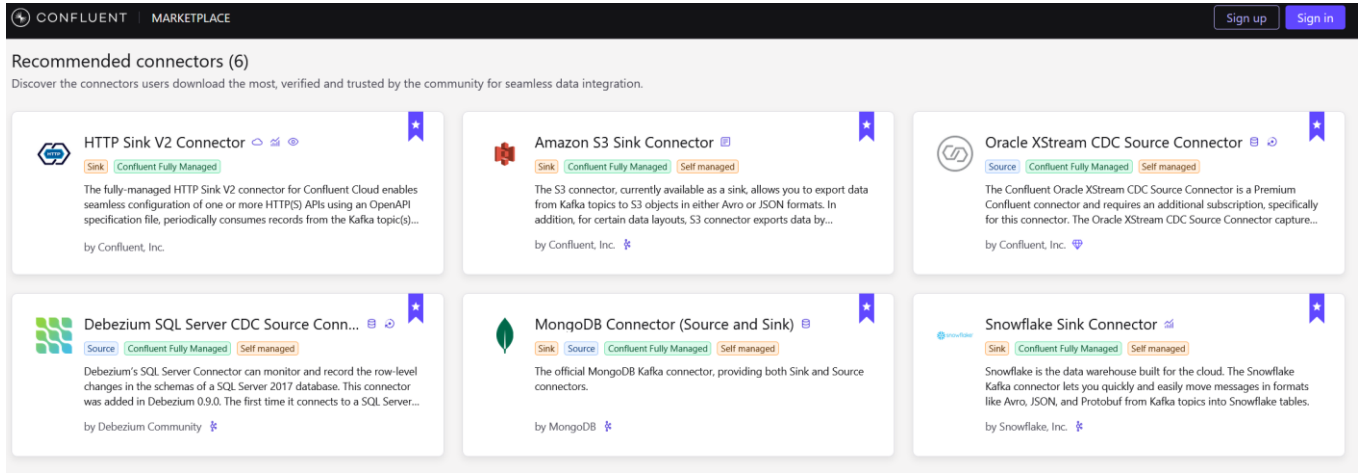
  ※ 基幹部分はコミュニティで開発しているが、各システムと組み合わせるために実装の一部は行う必要がある。
  その実装の一部に関しては自分たちで作ったものか、サードパーティー製のものを使う

  ※ 全体がKafka 公式で用意されているのは File Connector というサンプル的な Connector のみ

- 例）
  - Debezium
  - JDBC Connector
  - Camel Connector
  - HDFS Connector
  - Elastic Search Connector
  - Iceberg Connector
  - SnowFlake Connector
  - …

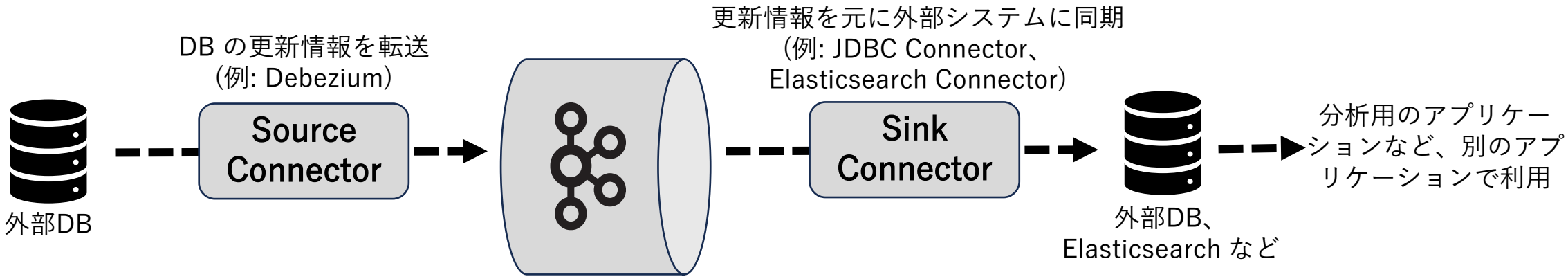
Confluent (Kafka のマネージドサービスを提供する企業) が Connector をまとめているサイト
https://www.confluent.io/hub/

Confluent 以外が開発した Connector も提供されており、用途に応じて柔軟に選べる



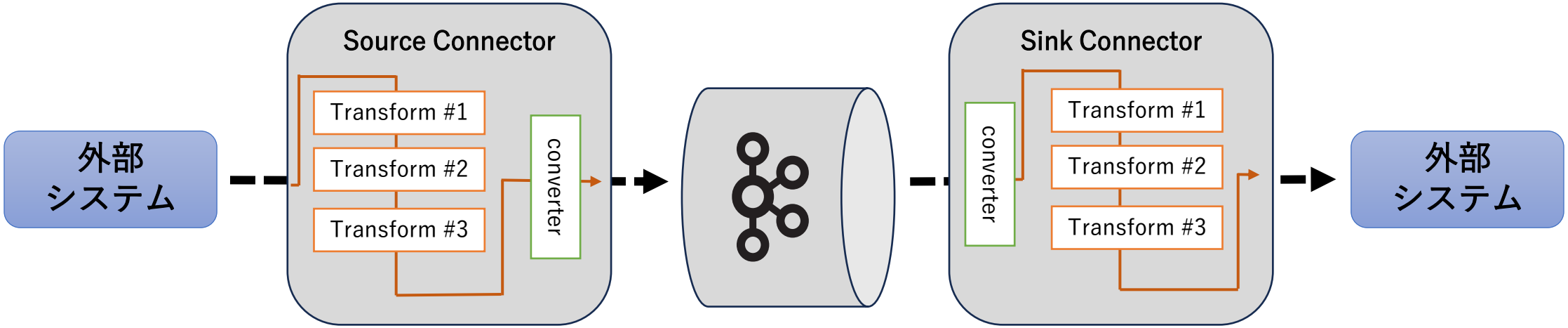
Kafka Connect のユースケース

- Change Data Capture (CDC)
- CDC はデータベースの更新情報を取得し、追跡できるようにする仕組み
- Kafka Connect を使う場合は、
  - 更新情報を Source Connector で Kafka に転送
  - Kafka に転送された更新情報を Sink Connector で別のデータベース等に転送
- 同期先は Elasticsearch など、分析用途に使うためのシステムにするという使い方も考えられる



レコードの加工処理

- Kafka Connect では Kafka への入出力前後において、レコードの簡易的な加工処理を行うことが可能
- Transform は複数※設定して処理を行うことも可能　※コネクタごとに最大10まで
- レコードの Drop、Filter、Flatten などに利用可能



- 高度な加工処理を行う場合は Kafka Streams や Flink を用いる

Kafka Connect の内部構造

Connector の動作モード

本番環境では distributed mode を利用するが、
standalone mode を利用すれば簡単に検証ができる

	distributed mode	standalone mode
サーバー台数	複数台	1台
分散の機能	利用可	利用不可
用途	本番環境用	開発、検証用
実行方法	bin/connect-distributed.sh (起動スクリプト)と REST API を用いて実行	bin/connect-standalone.sh (起動スクリプト) で実行 ※bin/connect-standalone.sh と REST API を用いる方法も使えるが、bin/connect-standalone.sh のみで完結できる

実行例

- standalone mode での実行例

```
$ bin/connect-standalone.sh config/connect-standalone.properties ¥
config/connect-file-source.properties
```

  - 起動スクリプトのみで完結

- distributed mode での実行例

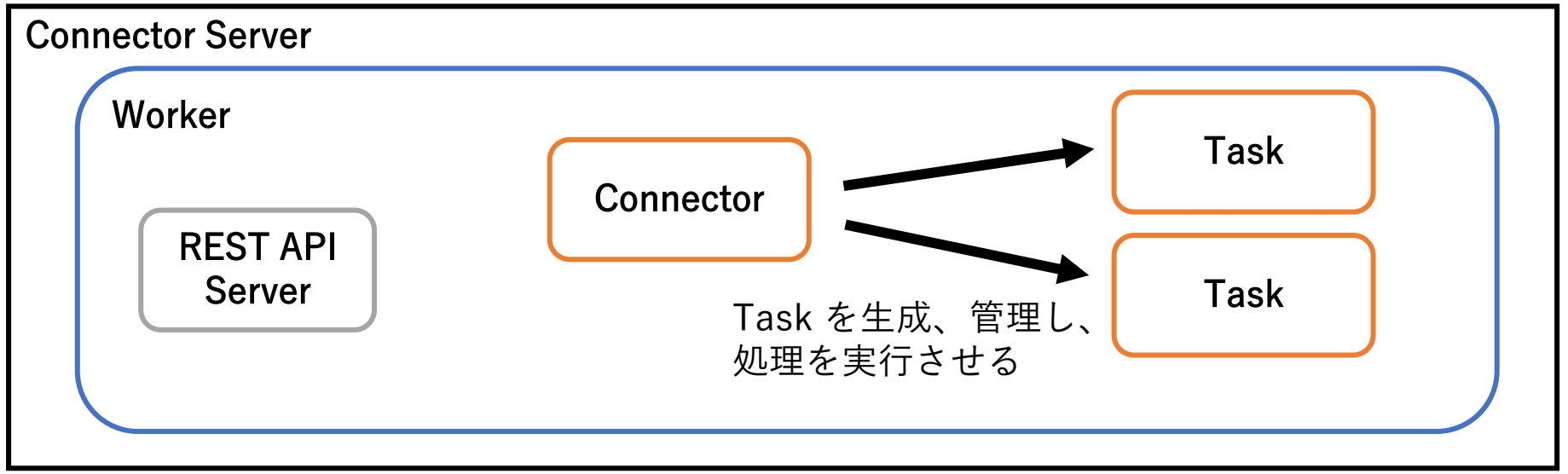
```
$ bin/connect-distributed.sh config/connect-distributed.properties

$ curl -X POST "http://localhost:18083/connectors" ¥
-H 'Content-Type: application/json' -d @jdbc_source_postgresql.json
```

  - 起動スクリプト実行後 REST API による操作が必要

standalone mode

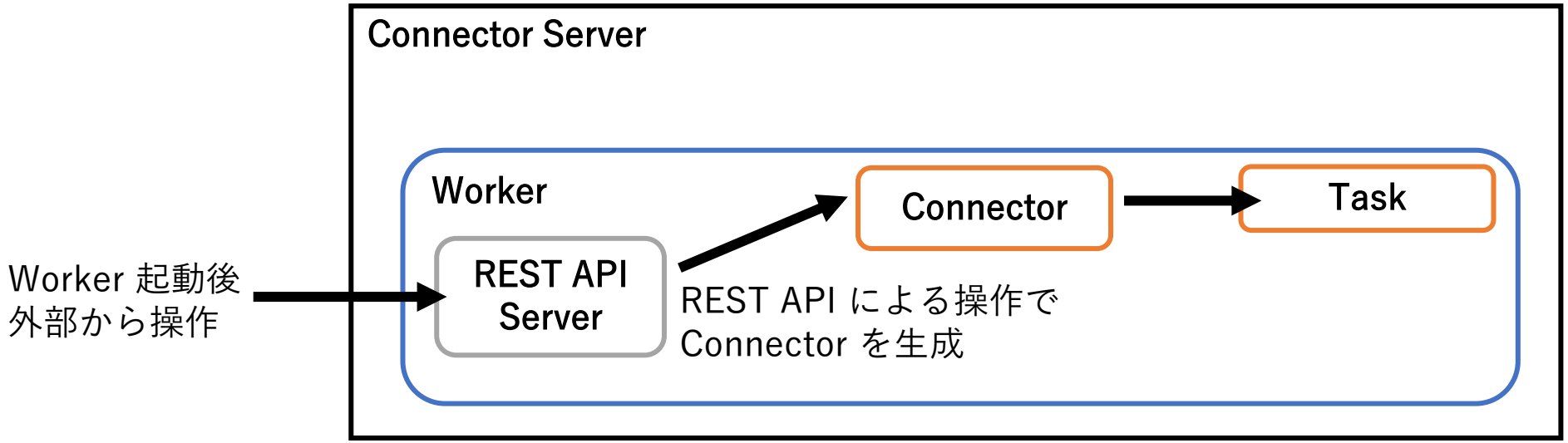
- standalone mode で動かすには bin/connect-standalone.sh を実行
- bin/connect-standalone.sh によって Worker という Java プロセスを起動し、さらに Worker 上で Connector というスレッドを作成できる
- Connector は Task というスレッドを作成して管理し、処理を実行させる

※複数の Connector を同時に実行することも可能
※各システム向けに Kafka Connect を実装したものも Connector というが、ここで作成された Task を管理するスレッドも Connector という



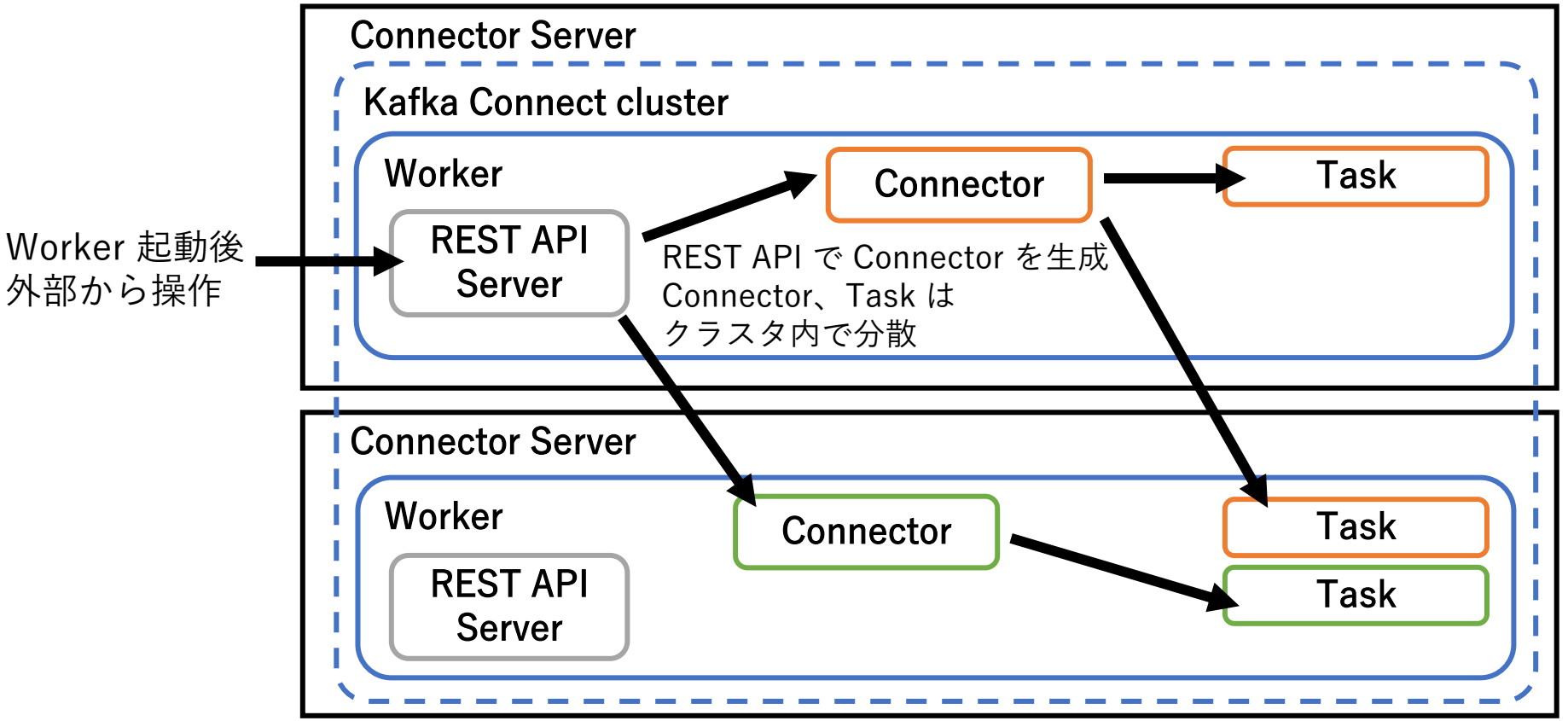
distributed mode (1/2)

- distributed mode で動かすには、まず bin/connect-distributed.sh を用いてWorker を起動
- その後、REST API で操作し Connector を起動
- Connector が Task を作成するのは standalone mode と同様



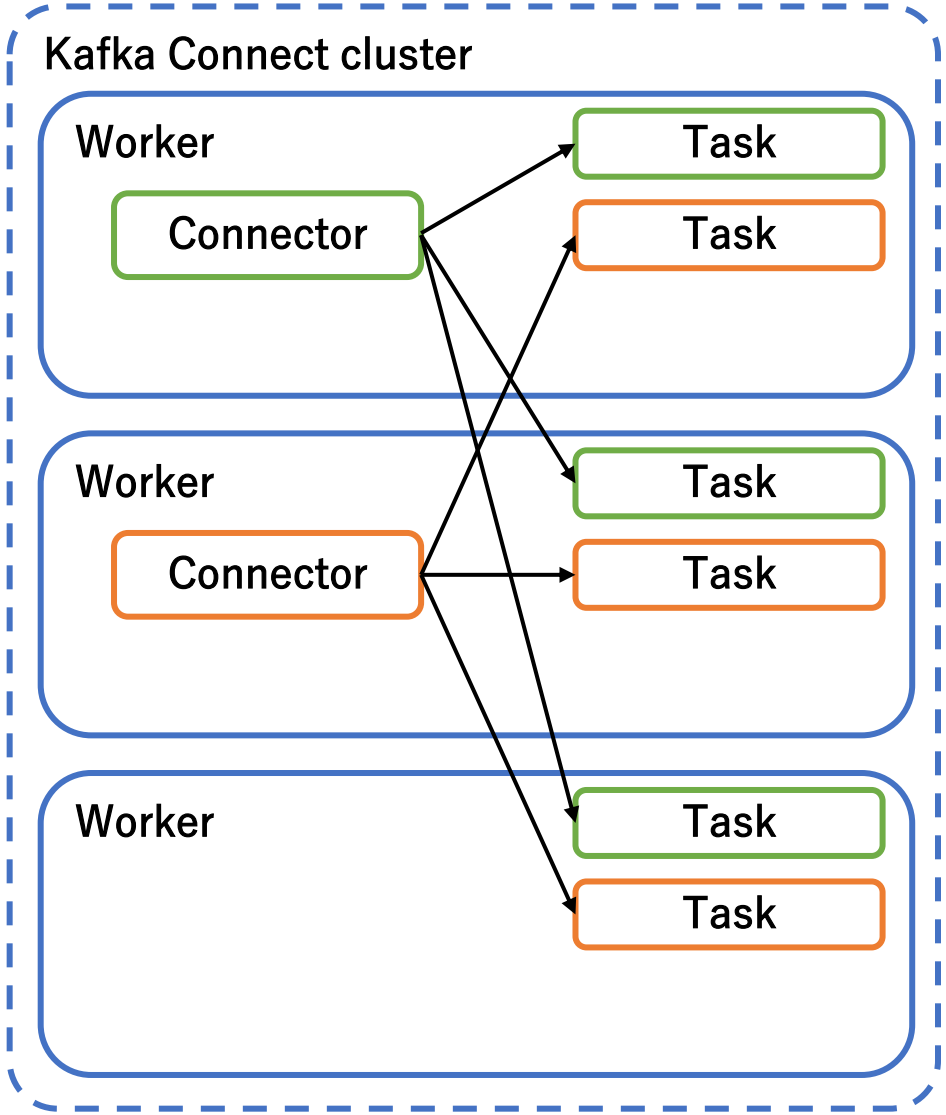
distributed mode (2/2)

- distributed mode でも Worker や Connector 等、生成されるものは同じだが、Worker を複数起動し、クラスタを構成することができる
- クラスタ内で負荷が均等になるように Connector と Task は分散される



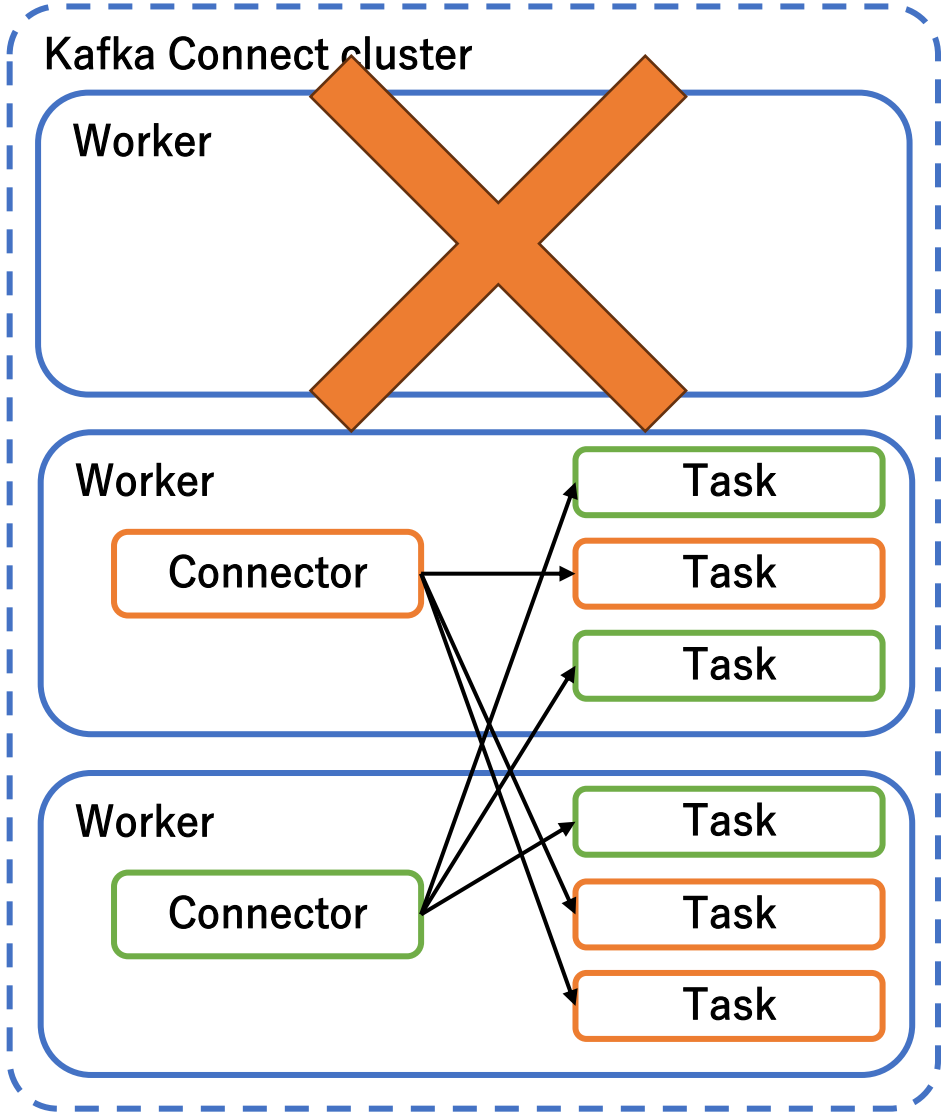
障害時の挙動

- Worker 障害時は Connector、Task を再配置（リバランス）し処理を継続
- 内部的には Consumer Group と同じ仕組みでリバランス
  - Worker の Group を作って
    障害時は Group 内でリバランス、etc.
- 引き継ぐ際は内部 Topic から情報を受け取る



障害時の挙動

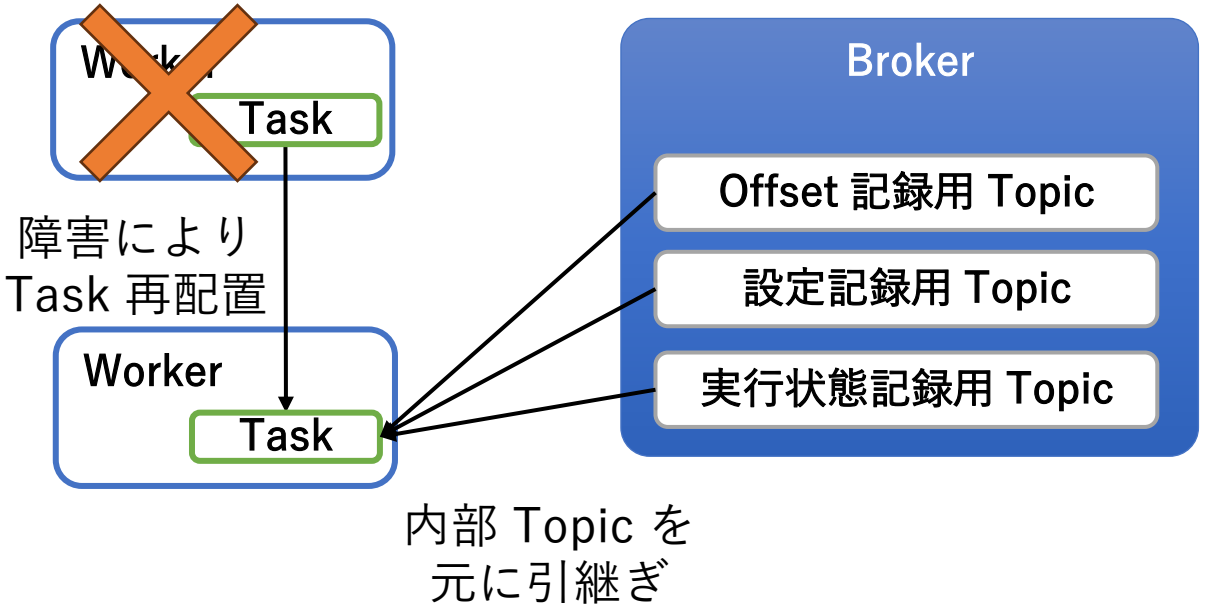
- Worker 障害時は Connector、Task を再配置（リバランス）し処理を継続
- 内部的には Consumer Group と同じ仕組みでリバランス
  - Worker の Group を作って
    障害時は Group 内でリバランス、etc.
- 引き継ぐ際は内部 Topic から情報を受け取る



内部 Topic による引継ぎ

- Connector や Task を別の Worker が引き継ぐ際に必要な情報が記録された Topic (Topic 名は Worker の設定値)
  - Offset 記録用 Topic (Source Connector が利用)
  - 設定記録用 Topic
  - 実行状態記録用 Topic

※Sink Connector は Kafka の Topic からデータを取り出すので、すでに説明した Offset Commit の仕組みを利用する。



まとめ

Kafka 基礎と Kafka Connect まとめ

- **Kafka はストリーム処理を行うための分散型ストリーミングシステム**
  - リアルタイムでデータの処理ができる
  - データハブとして利用できる
- **データを漏れなく送受信するための仕組み**
  - Ack、Offset Commit によって、メッセージの欠損を防ぐことができる
- **耐障害性、スケーラビリティ**
  - Partition を複数の Broker で複製し、耐障害性を高められる
  - Consumer Group を形成し、Consumer の増減に対応できる
- **Kafka Connect は Kafka と外部システムのデータ連携を容易にする機能**
  - 簡単なデータ加工ができ、Connector によっては exactly once のサポートもある
- **複数の Worker でクラスタを構築し、並列度をあげつつ可用性を担保**

SRA OSS の Apache Kafka 関連サービス

- SRA OSSでは「Kafkaの使い方が分からない」「導入を支援して欲しい」というお客様に向けて、**「Apache Kafka コンサルティングサービス」**を実施しています
- Kafka にお困りのお客様はぜひお気軽にお声がけください

【SRA OSS】新サービス開始「Apache Kafka コンサルティングサービス」～
リアルタイムデータ基盤の構築支援で企業のデータ活用を促進 ～

Home > 会社概要 > プレスリリース・インフォメーション > 【SRA OSS】新サービス開始「Apache Kaf...

プレスリリース PDF 版 712 KB

2026 年 3 月 17 日
株式会社SRA OSS

株式会社SRA OSS(東京都豊島区、代表取締役社長: 稲葉 香理、以下 SRA OSS) は、オープンソースの分散型ストリーミングシステム「Apache Kafka®」について、PoC、設計、導入等を支援するコンサルティングサービスを開始します。

近年、企業のデジタルトランスフォーメーション（DX）推進やAI活用の拡大に伴い、リアルタイムでデータを収集・連携・処理するデータ基盤へのニーズが急速に高まっています。
こうした背景を受け、SRA OSSはApache Kafkaの活用支援を開始します。

Apache Kafkaは、大量データの保存・処理を効率的に行うために設計された分散型ストリーミングシステムです。従来のメッセージキューと比べて高いスループットと拡張性、耐障害性を備えており、マイクロサービス連携やログ集約、IoTデータ処理、リアルタイム分析基盤など、幅広い分野で活用が進んでいます。

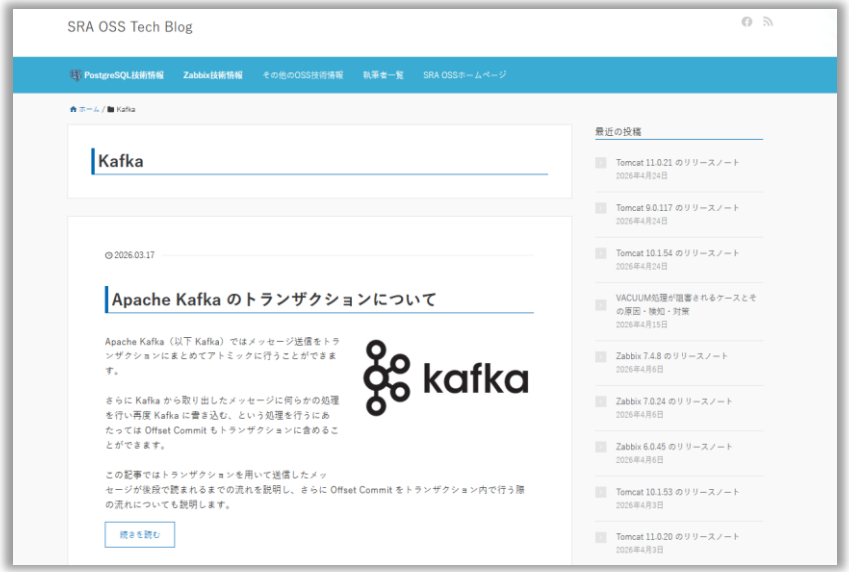
プレスリリース情報（2026/3/17）

https://www.sraoss.co.jp/prod_serv/consulting/kafka/

今後もサポートサービスなどを順次拡大予定です

SRA OSS 技術情報

- SRA OSS Tech Blog
  - https://www.sraoss.co.jp/tech-blog/
  - Apache Kafkaをはじめ、さまざまなOSSの技術情報を掲載中
- SRA OSS 公式 Youtube チャンネル
  - https://www.youtube.com/c/sraoss-official
  - 過去のセミナー動画を公開中



SRA OSS

– RECRUIT –

【 Apache Kafka エンジニア 】
【 インフラ系OSSエンジニア 】

OSSのように、オープンで情熱的であれ。

全世界のエンジニア・OSSコミュニティの揺るぎない信念とたゆまぬ努力に敬意を。
私たちも情熱を持ち、会社の枠を超えてオープンであり続けます。

Our Value

- OSS貢献
- カスタマーファースト
- マルチジョブ/マルチキャリア

株式会社SRA OSS 人事担当
✉ personnel@sraoss.co.jp

業務内容、待遇、カジュアル面談などお気軽にお問合せください。

SRA OSS

ご清聴ありがとうございました。

お問い合わせ: sales@sraoss.co.jp 03-5979-2701