Apache Kafka 入門: Kafka の基礎と Kafka クラスタ監視の要点

2025/11/22

OSC2025 Fukuoka

株式会社SRA OSS

奥井一暁

© 2025 SRA OSS K.K.

会社紹介

株式会社SRA OSS

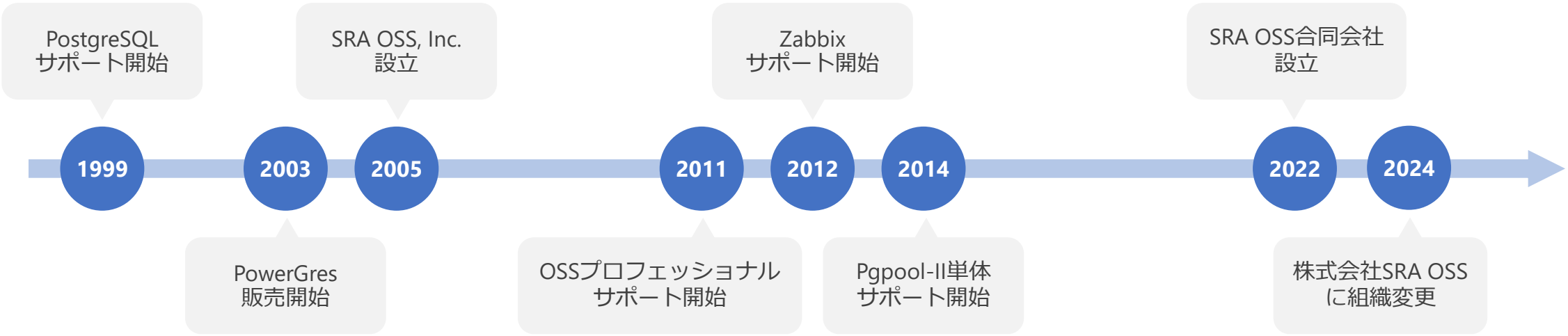
所 在 地: 東京都豊島区南池袋2-32-8
設 立 日: 2022年6月17日
株　　主: 株式会社SRA
株式会社NTTデータ
資 本 金: 7,000万円
社　　長: 稲葉 香理

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)



© 2025 SRA OSS K.K.

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
- 今年度から Kafka 担当
- 趣味: 楽器演奏

© 2025 SRA OSS K.K.

目次

- Kafka の概要
- データ構造
- データ送受信
- 耐障害性とスケーラビリティ
- 送達保証
- Kafka クラスタ監視
- まとめ

© 2025 SRA OSS K.K.

Kafka の概要

© 2025 SRA OSS K.K.

Apache Kafkaとは

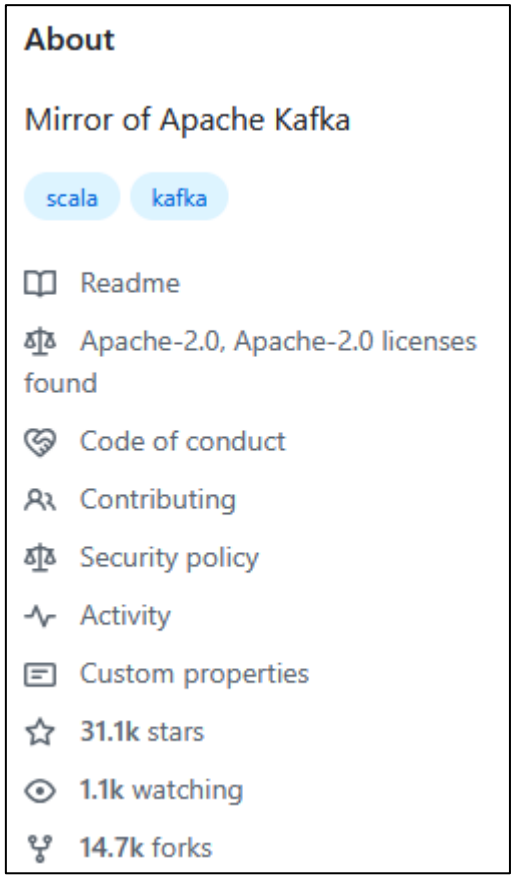
Apache Kafka（以下Kafka）はストリーム処理を実現するうえで欠かすことのできない、
オープンソースの分散メッセージングシステム

元々は 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: 4.1.0

2025/10/22 時点



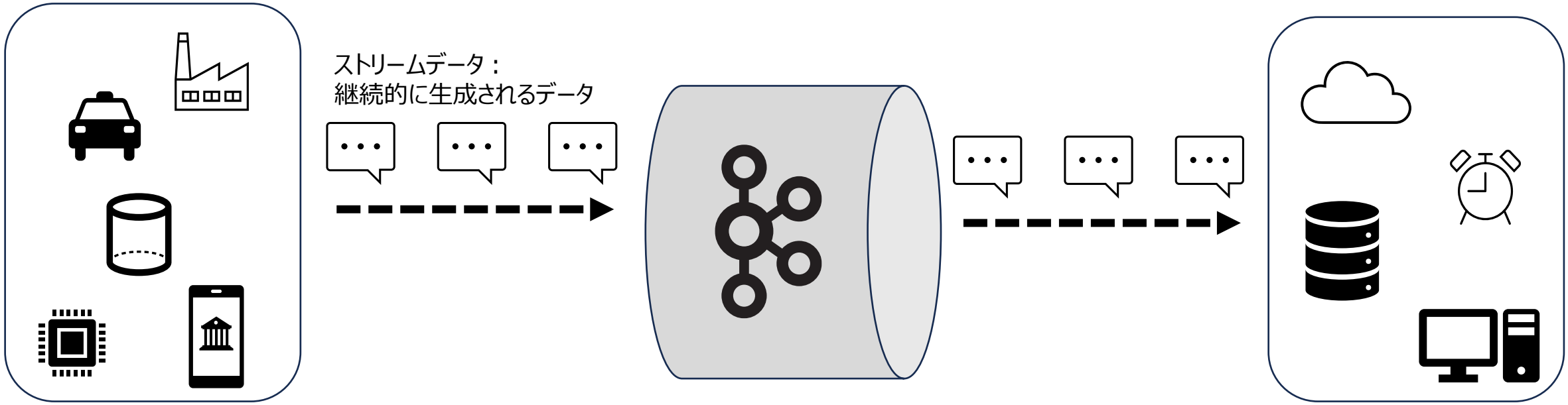
© 2025 SRA OSS K.K.

ストリーム処理

Kafkaが活用される処理として、ストリーム処理という考え方がある

従来主流だったバッチ処理に比べ、到着したデータを逐次処理するという違いがある

ストリーム処理時のデータの送信先としてKafkaを利用することで、データの取り回しが容易に



生成元

1つあたりのデータサイズは小さいことが多い

画像やセンサデータなど種類は様々

利用先

センサの異常検知

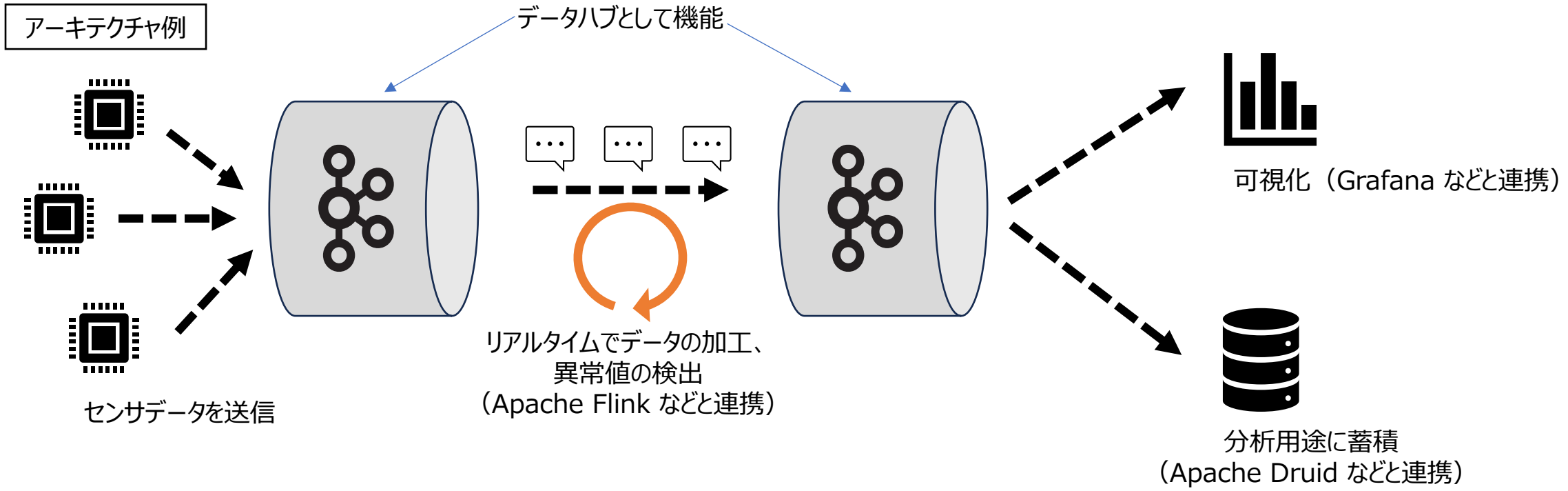
DBやその他システムへの蓄積など

© 2025 SRA OSS K.K.

Kafka活用例：センサデータの活用

Kafka活用の一例として、センサデータを収集して異常を検知することなどが可能

収集したデータはほかのソフトウェアとも連携して分析などにも利用できる



© 2025 SRA OSS K.K.

Kafkaの特徴

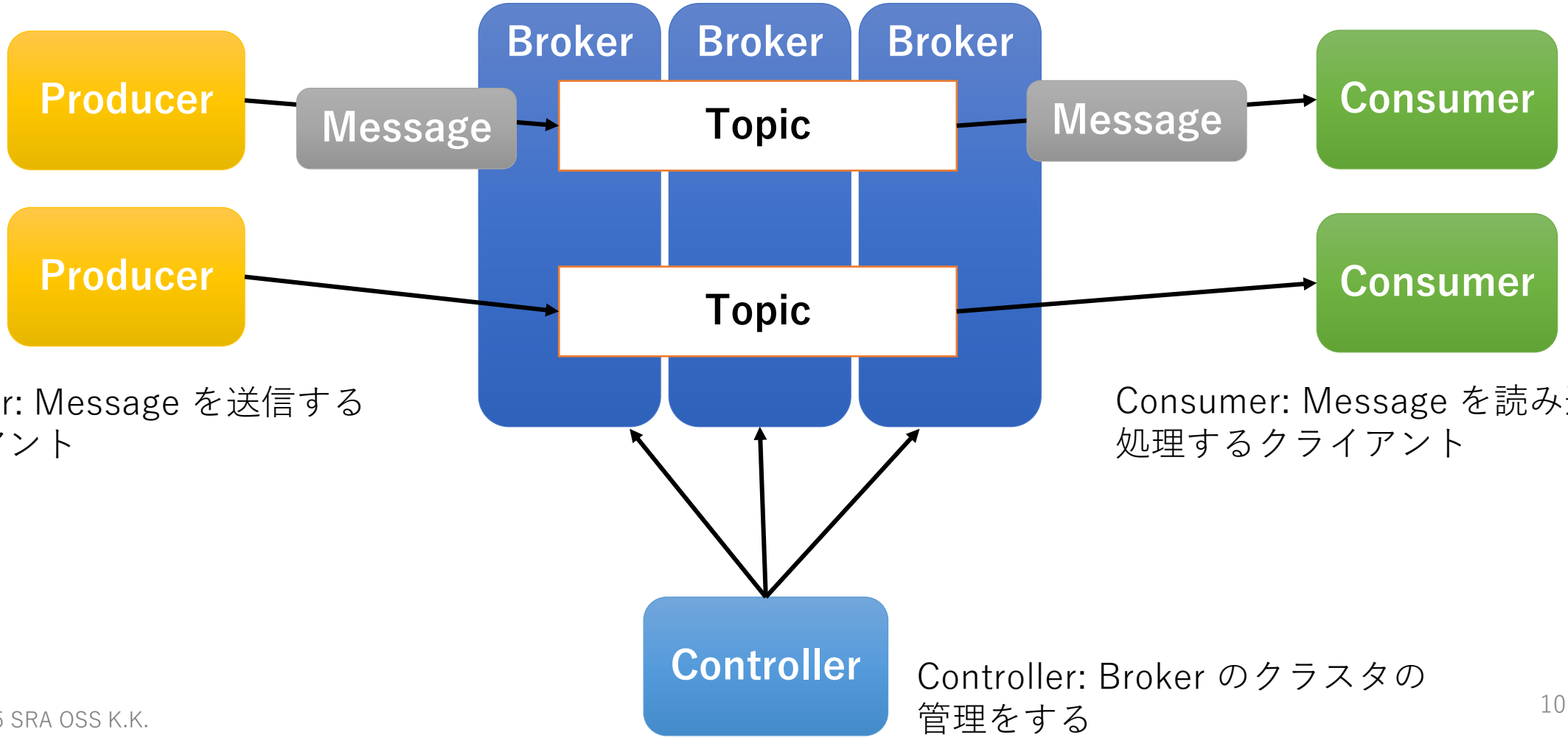
- スケーラブル
  - サーバ台数はいつでも増やすことができ、処理に応じて拡張できる
- 低レイテンシ
  - 都度ディスクへの書き込みを行わないため高速に処理できる
- 高可用性
  - データは複数に複製されるため障害時にもデータを保全する
  - 複数の送達保証レベルを実現するための機能がある
- 多様なシステム連携
  - 様々なクライアントライブラリをサポートし、多様なシステムと連携できる

© 2025 SRA OSS K.K.

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
(Record と呼ぶこともある)

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: Message を送信する
クライアント

Consumer: 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

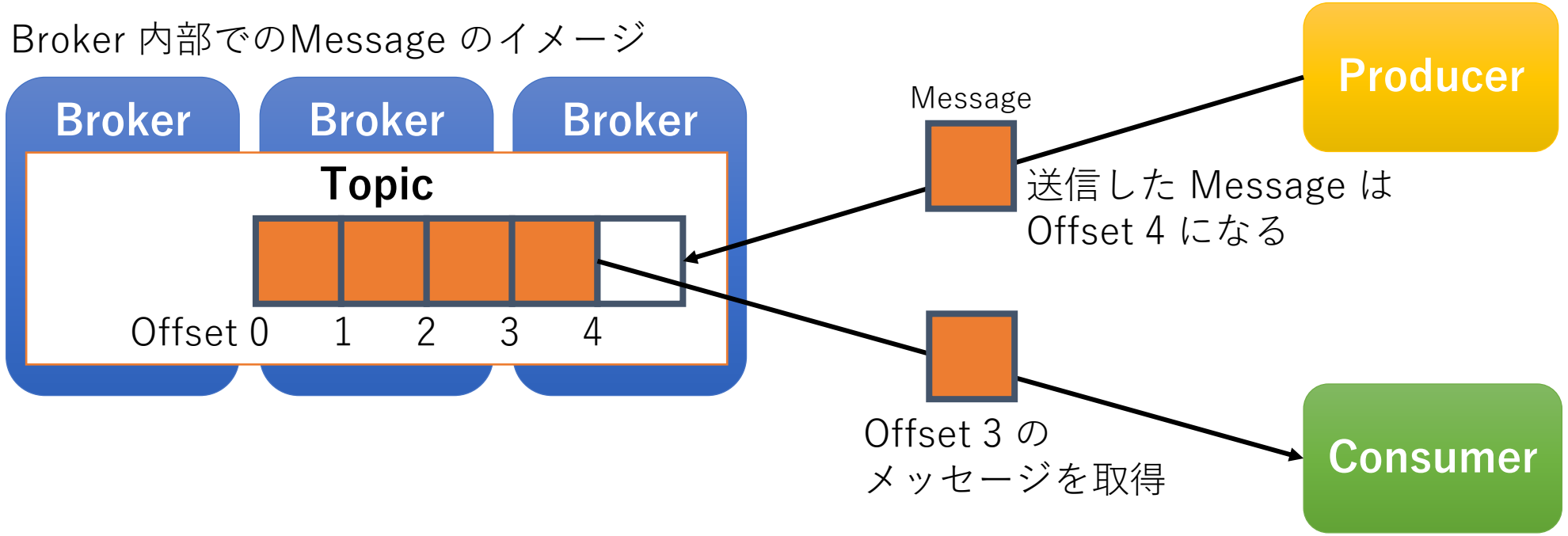
© 2025 SRA OSS K.K.

データ構造

© 2025 SRA OSS K.K.

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

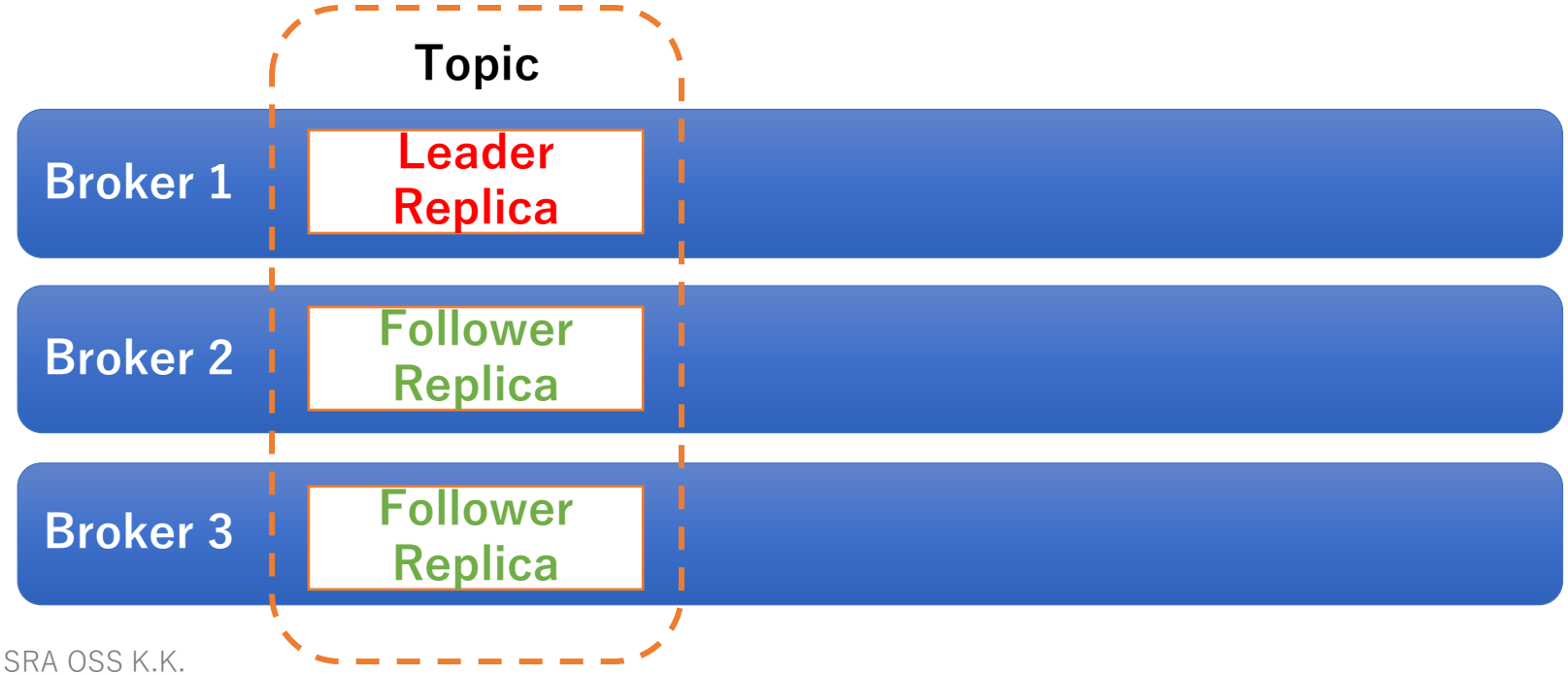


© 2025 SRA OSS K.K.

Replication

Replica: データの複製

- 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



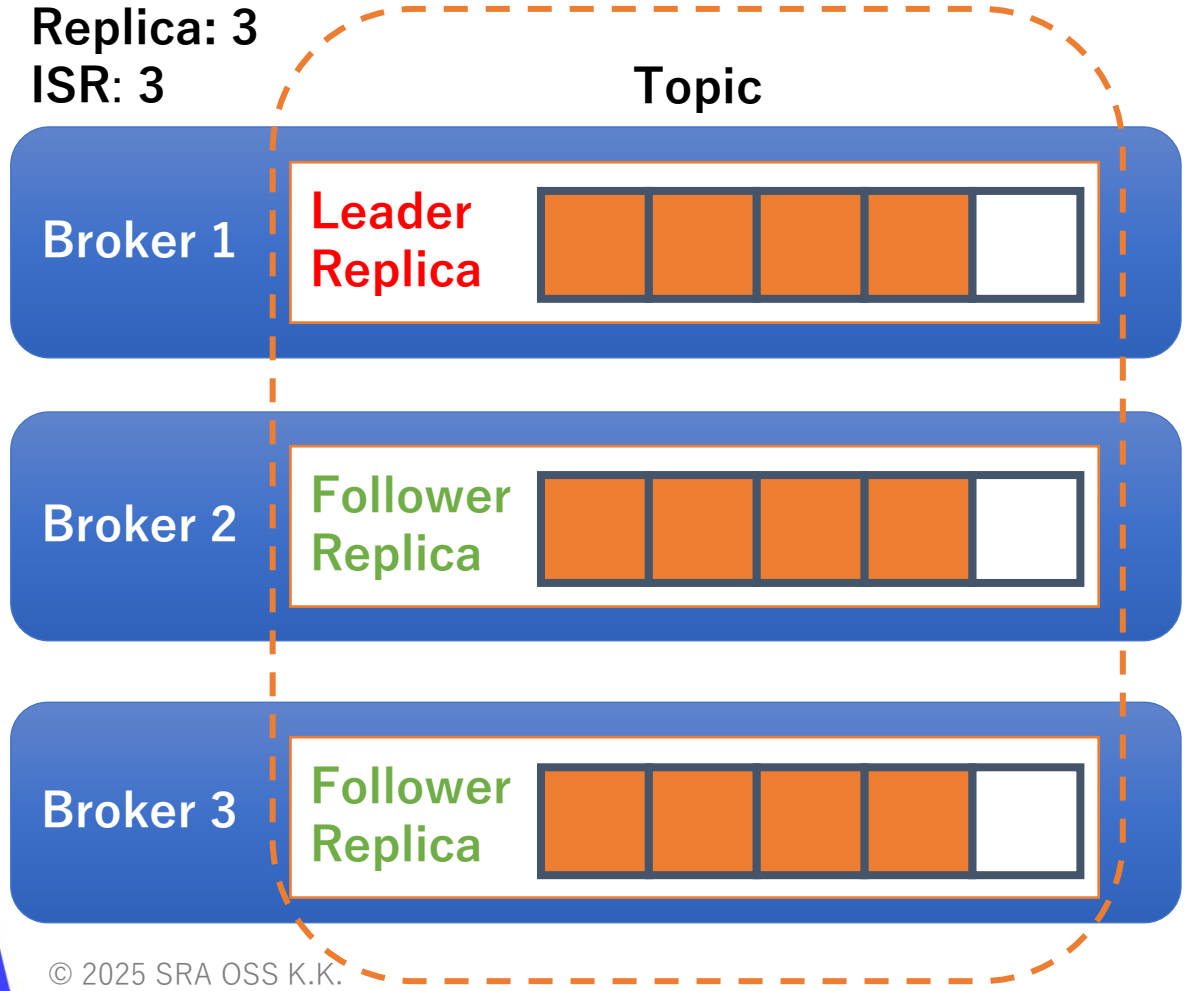
Replica: 3

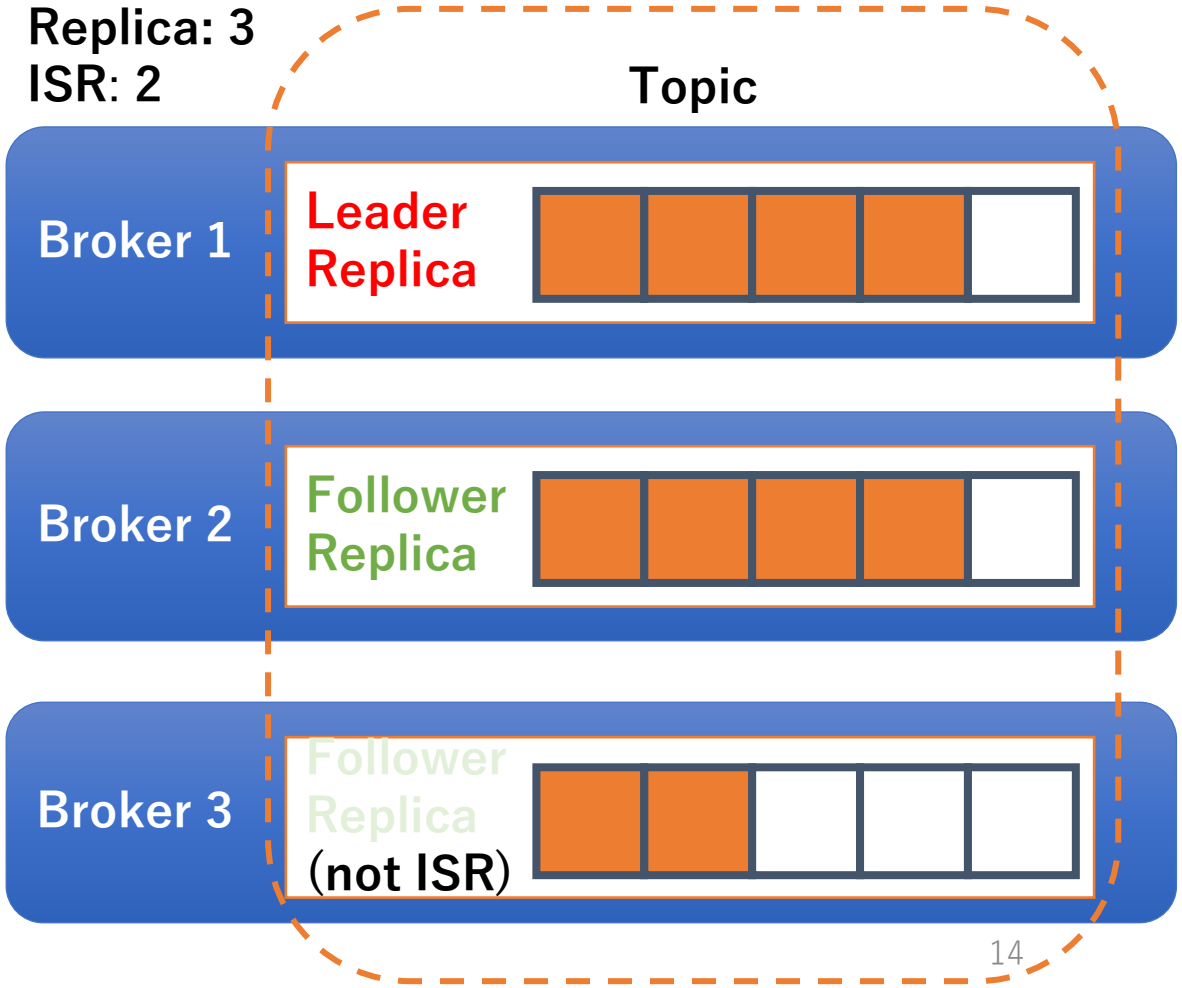
データ複製による耐障害性

© 2025 SRA OSS K.K.

複製の同期について

- In-sync Replicas (ISR): 複製が同期できている Replica のこと



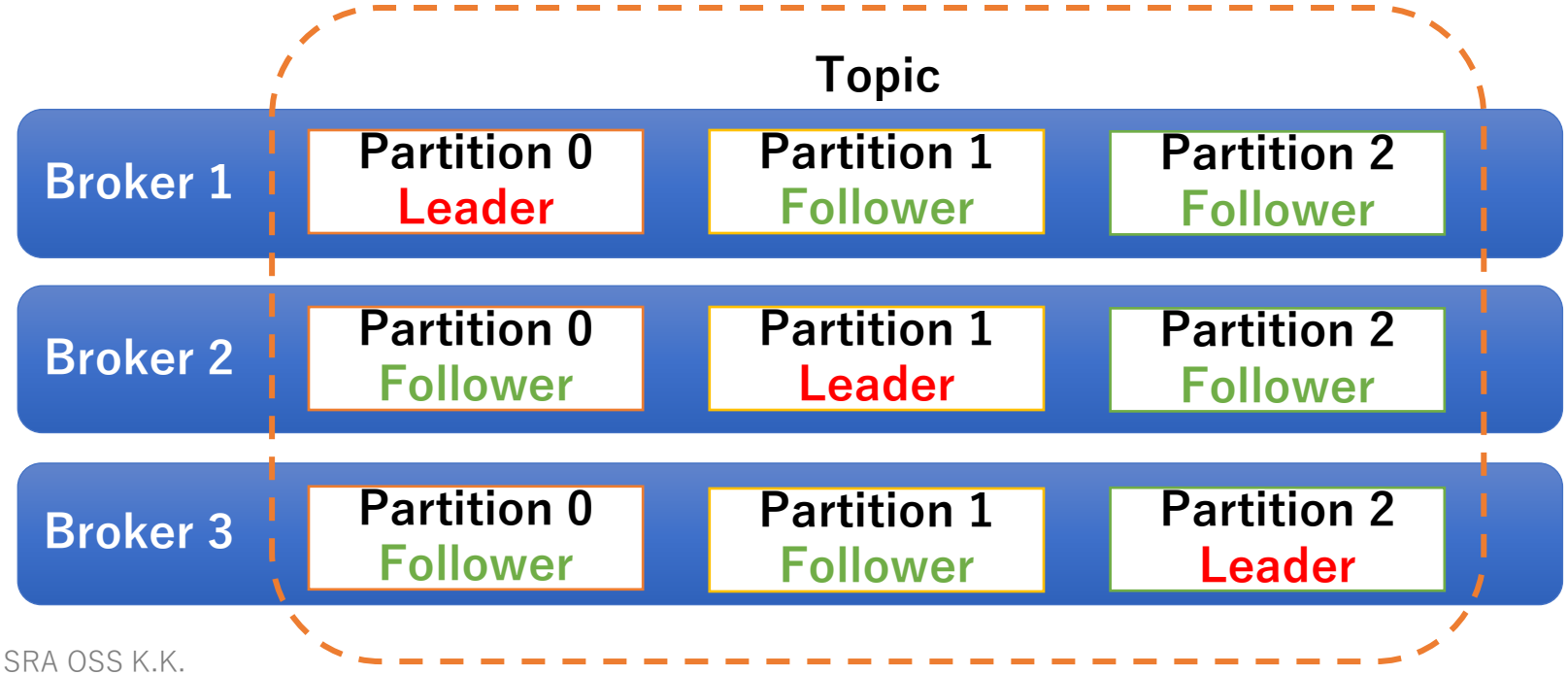


© 2025 SRA OSS K.K.

Partition

Partition: Topic を分割した単位

- 送信する Message は 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



Partition: 3
Replica: 3

分散による
スケールアウトと
耐障害性

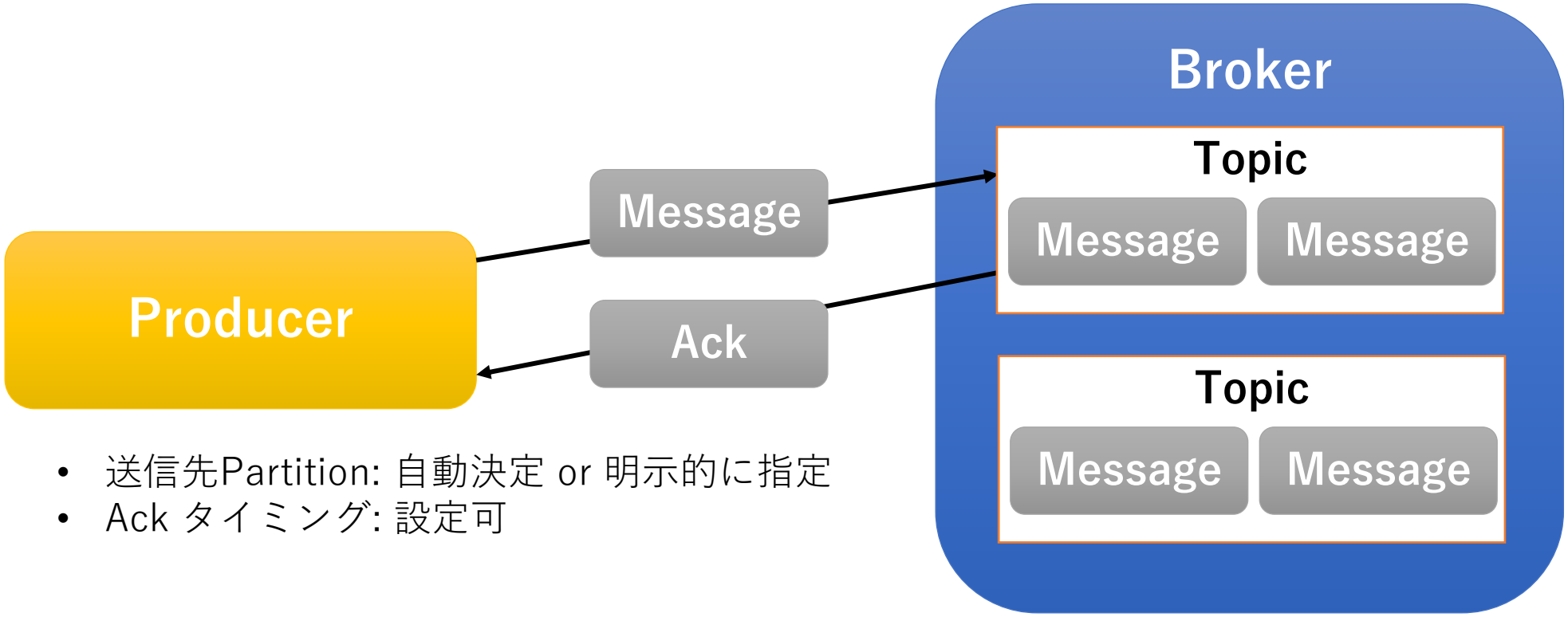
© 2025 SRA OSS K.K.

データ送受信

© 2025 SRA OSS K.K.

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)



- 送信先Partition: 自動決定 or 明示的に指定
- Ack タイミング: 設定可

© 2025 SRA OSS K.K.

送信先 Partition の決定方法

- Partitioner による自動決定
  - デフォルトのPartitioner
    - Message の Key をハッシュ値にして Partition 数で割った余りを元に決定する
    - 同じ Key の Message が同じ Partition に送信される
  - ラウンドロビン Partitioner
    - ラウンドロビンに決定する
  - 自作 Partitioner も利用可能
- 送信時に明示的に指定

Kafka では同一 Partition 内のMessage 順序は保証される
→Message の順序が重要なデータは同じ Partition に行くようにする

© 2025 SRA OSS K.K.

Acks

Producer 側の設定で、どこまで書き込みの確認が取れてから
クライアントに返答 (ack) するかを指定する設定値

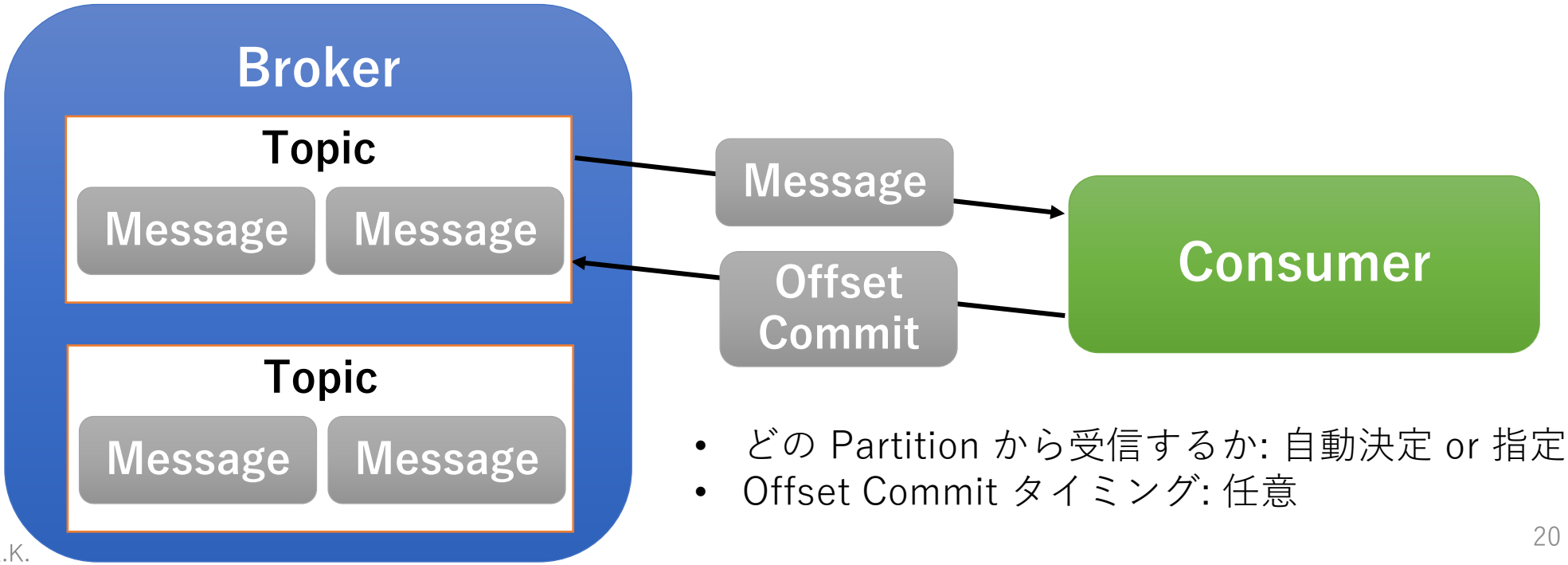
- Acks=0
  - Broker まで到達したかの確認さえしない
- Acks=1
  - Leader にまでメッセージが到達した時点で書き込みの完了と判断
  - Leader はメッセージを受け取った時点で ack を返す
- Acks=all (-1)
  - 全ての同期が出来ている Replica (In-Sync Replicas: ISR) に
    メッセージが到達した時点で書き込みの完了と判断
  - 全ての ISR へのメッセージ到達を Leader が判断し ack を返す

- Acks の設定は**送達保証**にも関わってくる→後のスライドで説明

© 2025 SRA OSS K.K.

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)

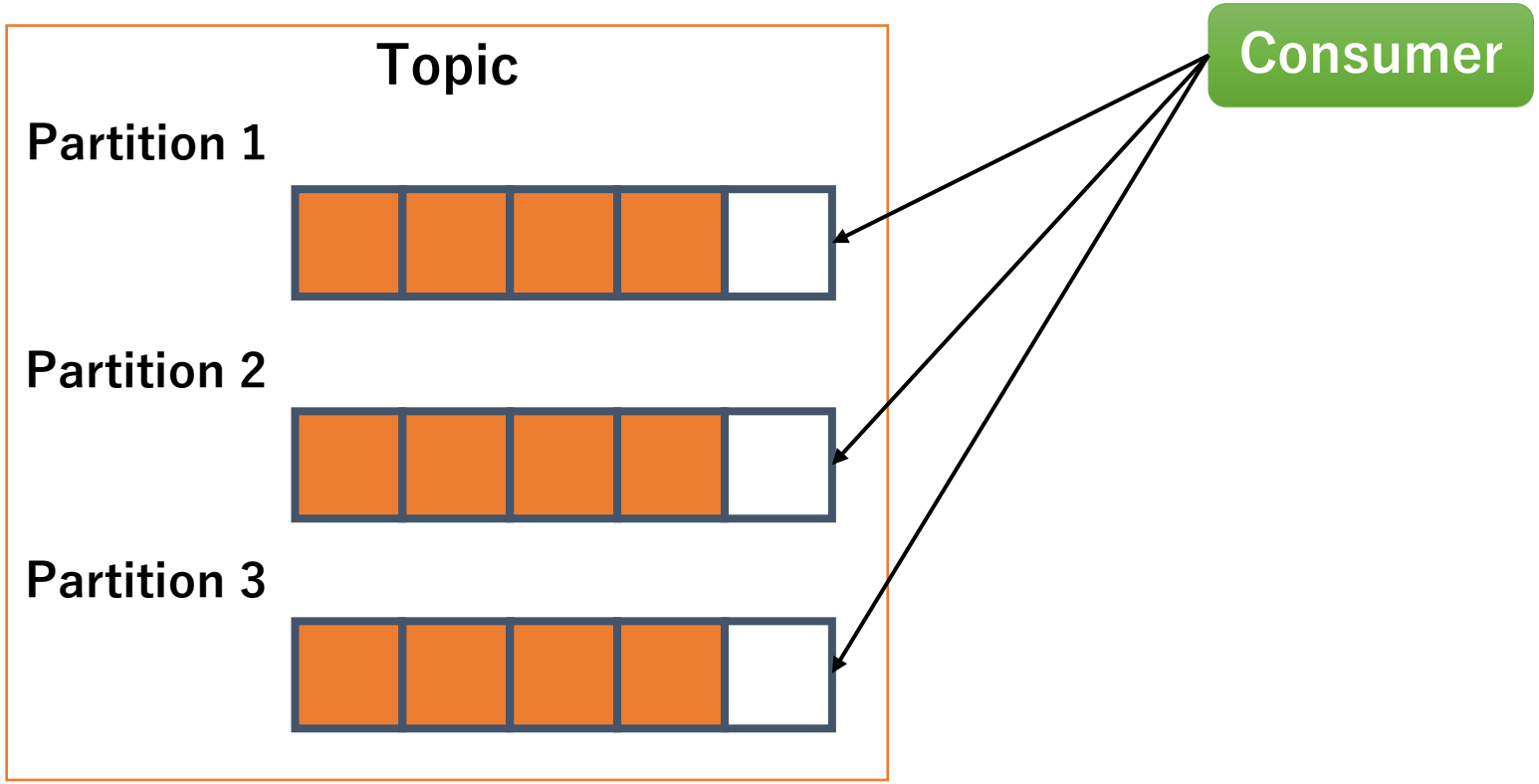


- どの Partition から受信するか: 自動決定 or 指定
- Offset Commit タイミング: 任意

© 2025 SRA OSS K.K.

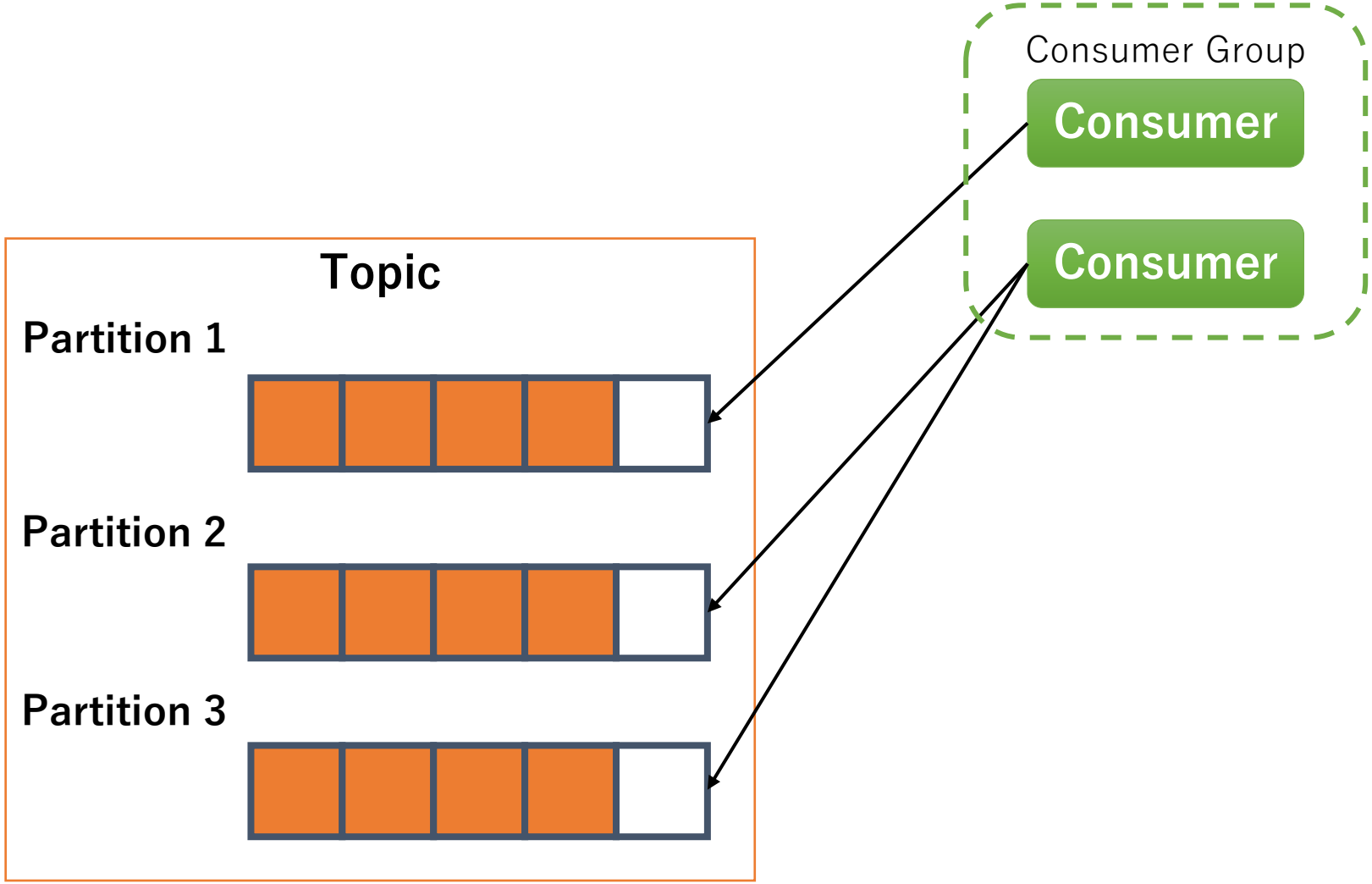
受信 Partition の自動アサイン

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



© 2025 SRA OSS K.K.

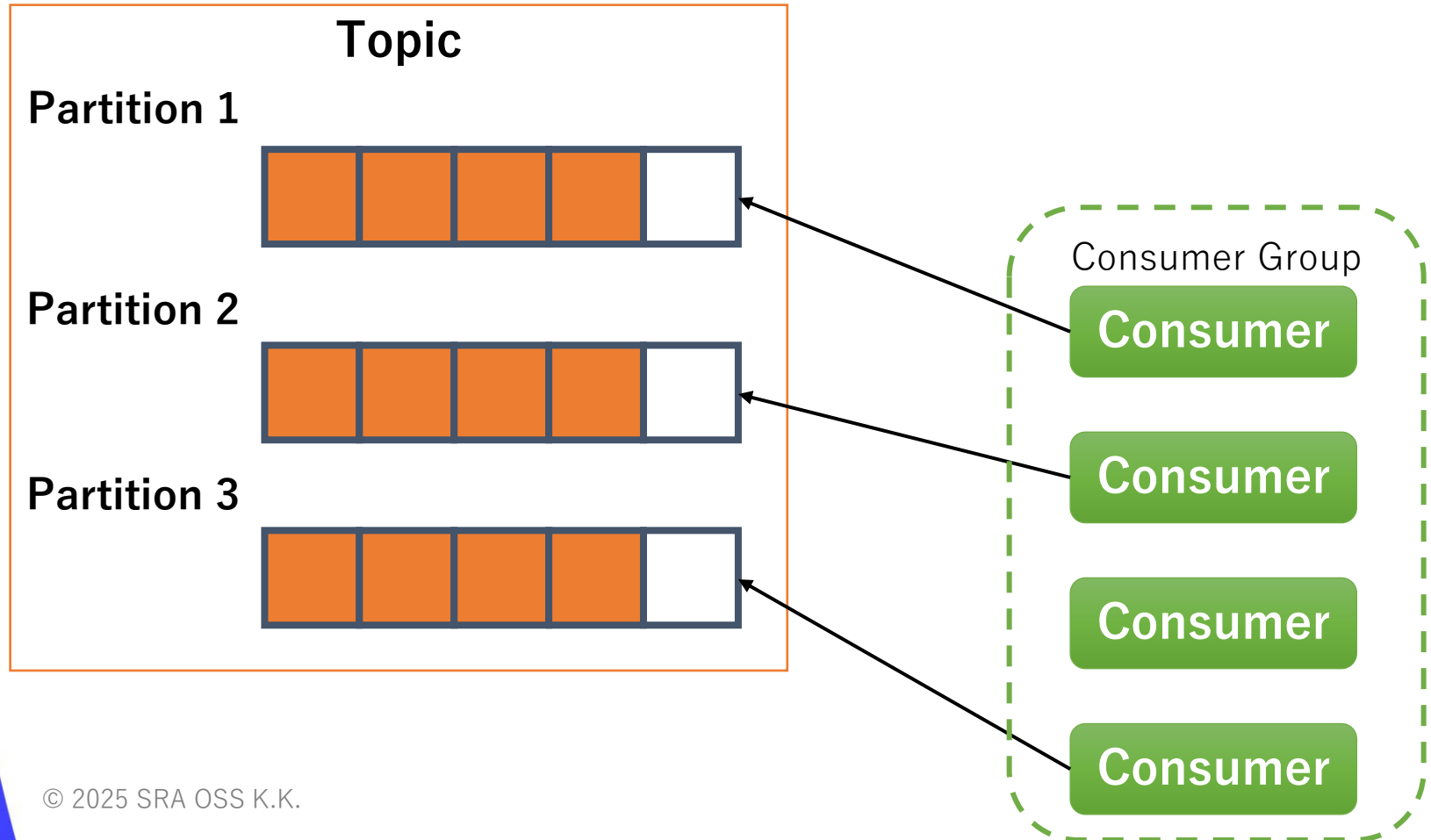
受信 Partition の自動アサイン



複数台を Consumer Group と呼ばれる単位でまとめると、Group 内で分担するようにアサインされる

© 2025 SRA OSS K.K.

受信 Partition の自動アサイン

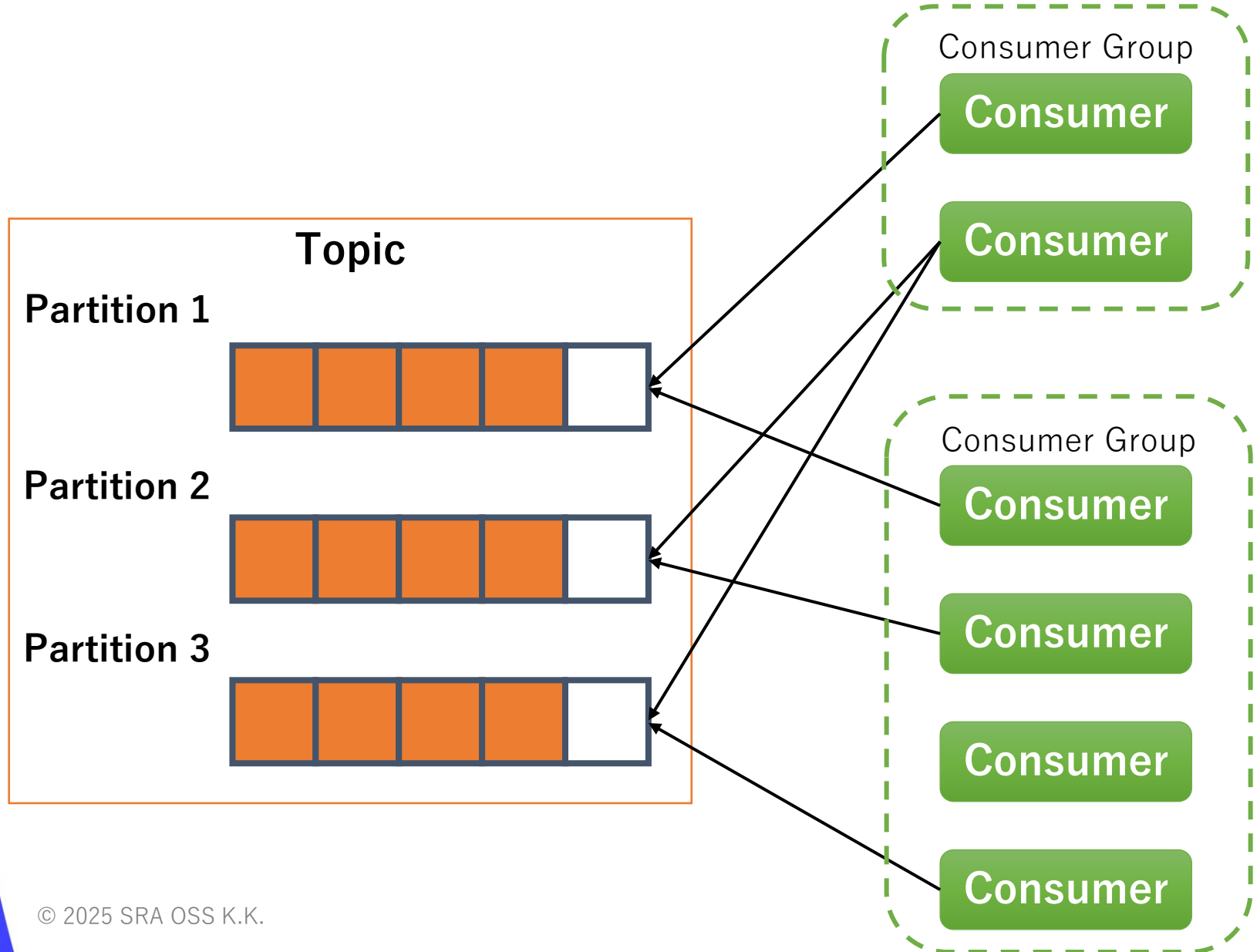


同じConsumer Groupでは同じPartition を複数のConsumer が担当することはない

Consumer が多ければ待機のConsumer も出てくる

© 2025 SRA OSS K.K.

受信 Partition の自動アサイン



Group が異なれば同じPartition もアサインされる

Consumer Group の概念は
耐障害性にも関わってくる
→後のスライドで説明

© 2025 SRA OSS K.K.

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

- Auto Offset Commit
  - 一定の間隔ごとに自動で Offset Commit を実行すること
  - On/Off の切り替え、間隔は設定値で変更可

- Manual Offset Commit
  - ユーザ側で Offset Commit のタイミングを実装する

Offset Commit タイミングは**送達保証**にかかわってくる
→後のスライドで説明

© 2025 SRA OSS K.K.

耐障害性とスケーラビリティ

© 2025 SRA OSS K.K.

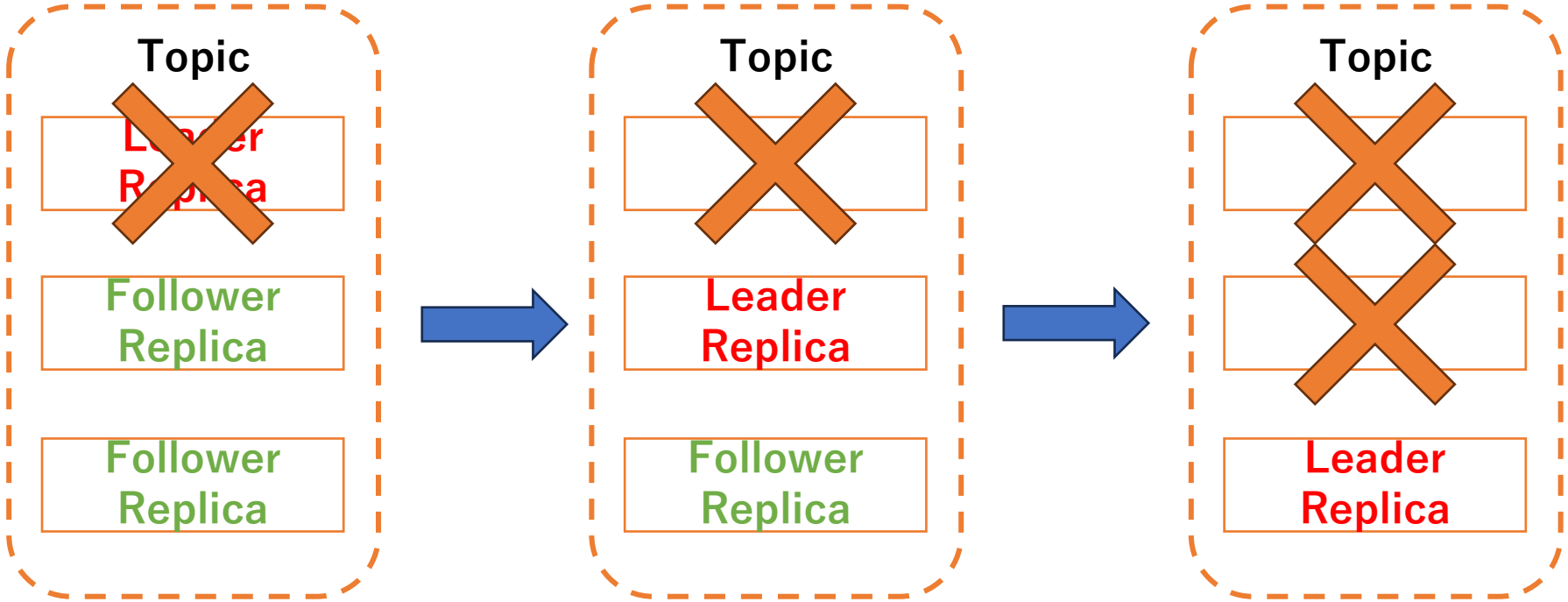
耐障害性とスケーラビリティの概要

- Broker:
  - 複数台、複数 Partition、Replication によって耐障害性を持たせられる
  - Broker を追加してクラスタを拡張できる
- Controller:
  - 複数台で管理情報を複製し耐障害性を持たせられる
- Consumer:
  - Consumer Group によってアサインされる Partition を分散できる
  - Consumer の障害時や、Consumer を増やして拡張する場合は Partition をアサインしなおして対応できる
- Producer:
  - Producer 障害時や追加時のための機能は Kafka では用意されていない

© 2025 SRA OSS K.K.

Broker 障害時のしくみ

- Leader Replica 障害時は ISR の Follower Replica が Leader に昇格



- ISR が少なくなってくると…

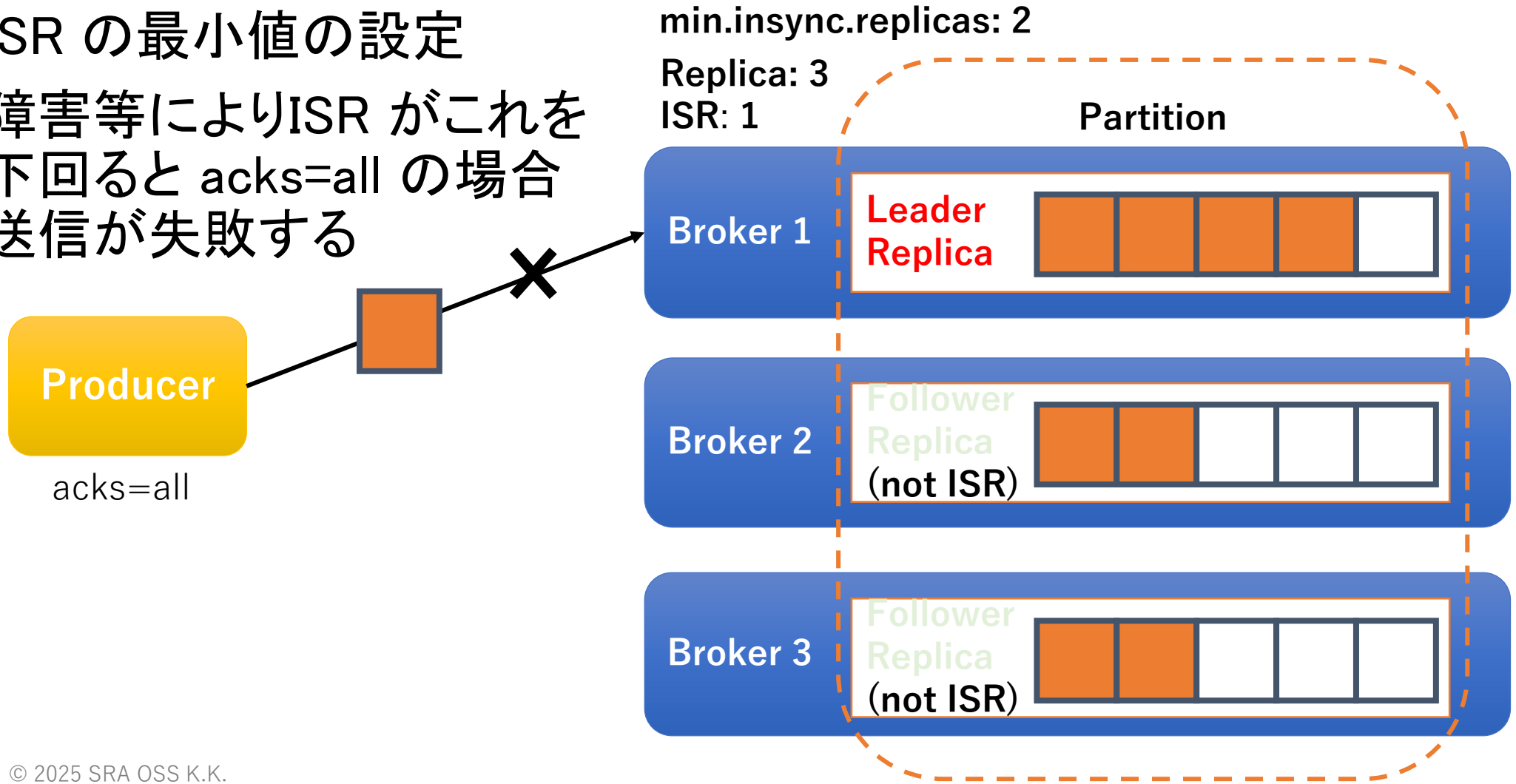
→送ったデータが複製できないという状態を防ぐために
送信を失敗させることができる

© 2025 SRA OSS K.K.

Broker 障害時のしくみ

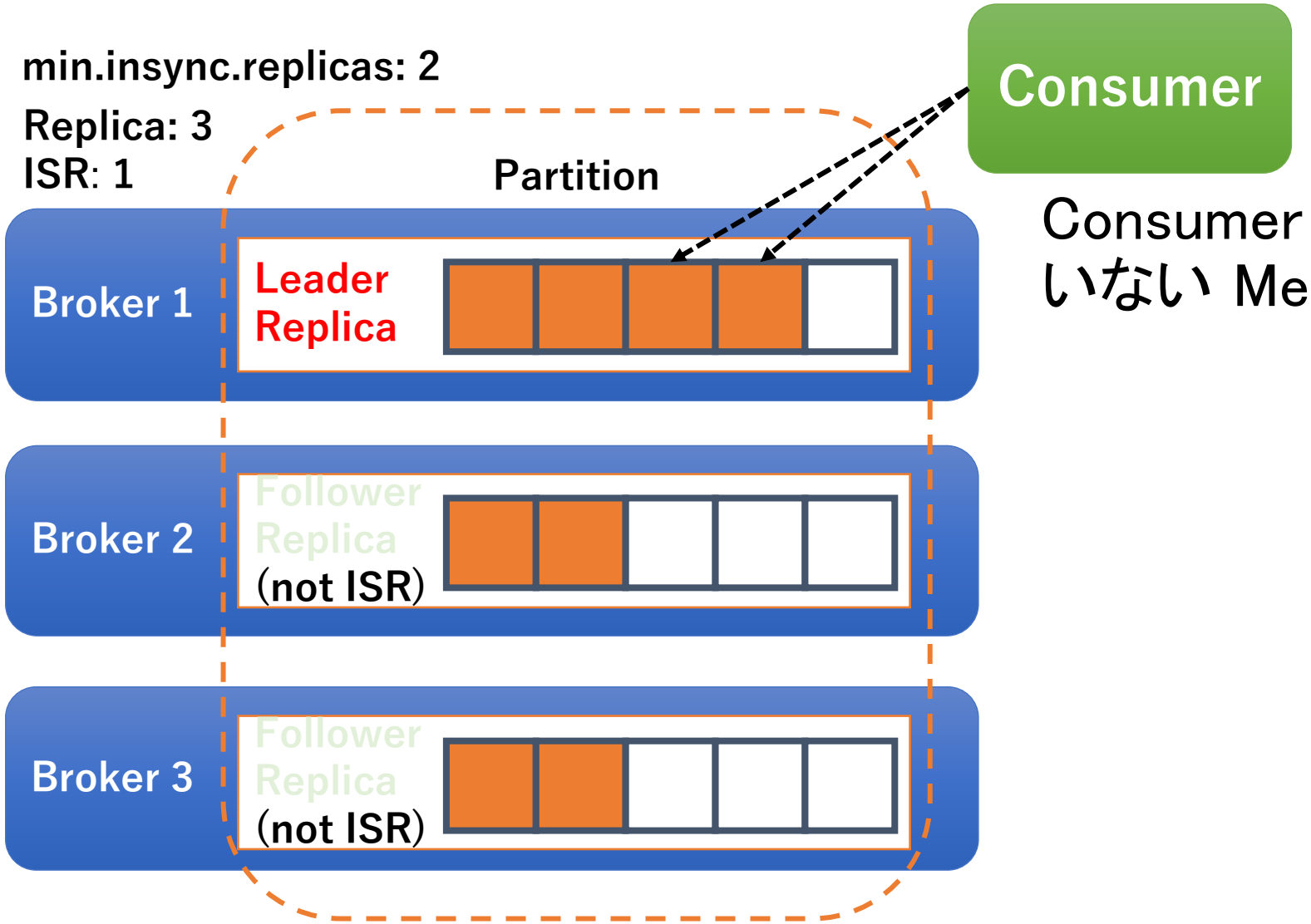
min.insync.replicas

- ISR の最小値の設定
- 障害等によりISR がこれを下回ると acks=all の場合送信が失敗する



© 2025 SRA OSS K.K.

Broker 障害時のしくみ

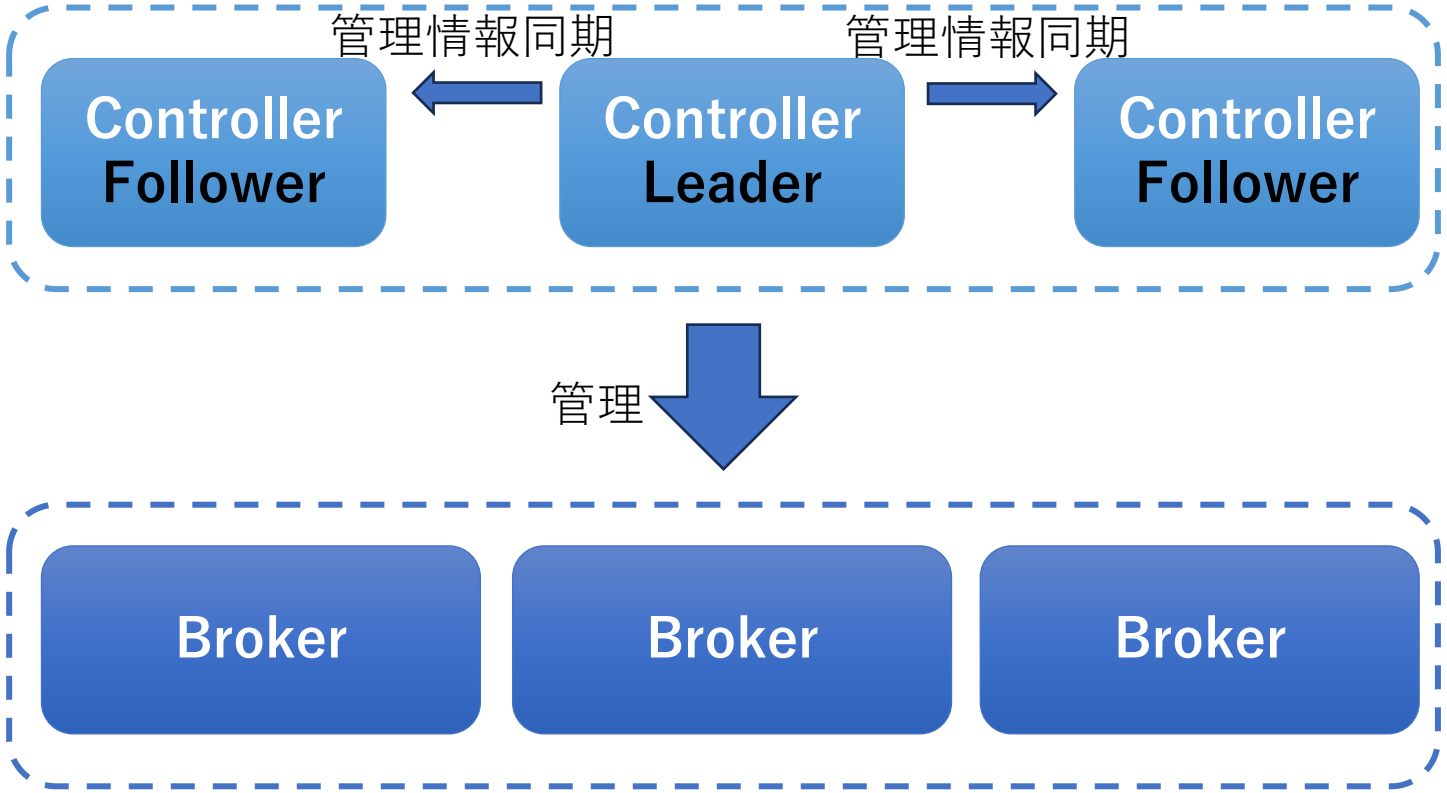


Consumer 側では複製が追い付いていない Message が取得できない

© 2025 SRA OSS K.K.

クラスタの管理機構

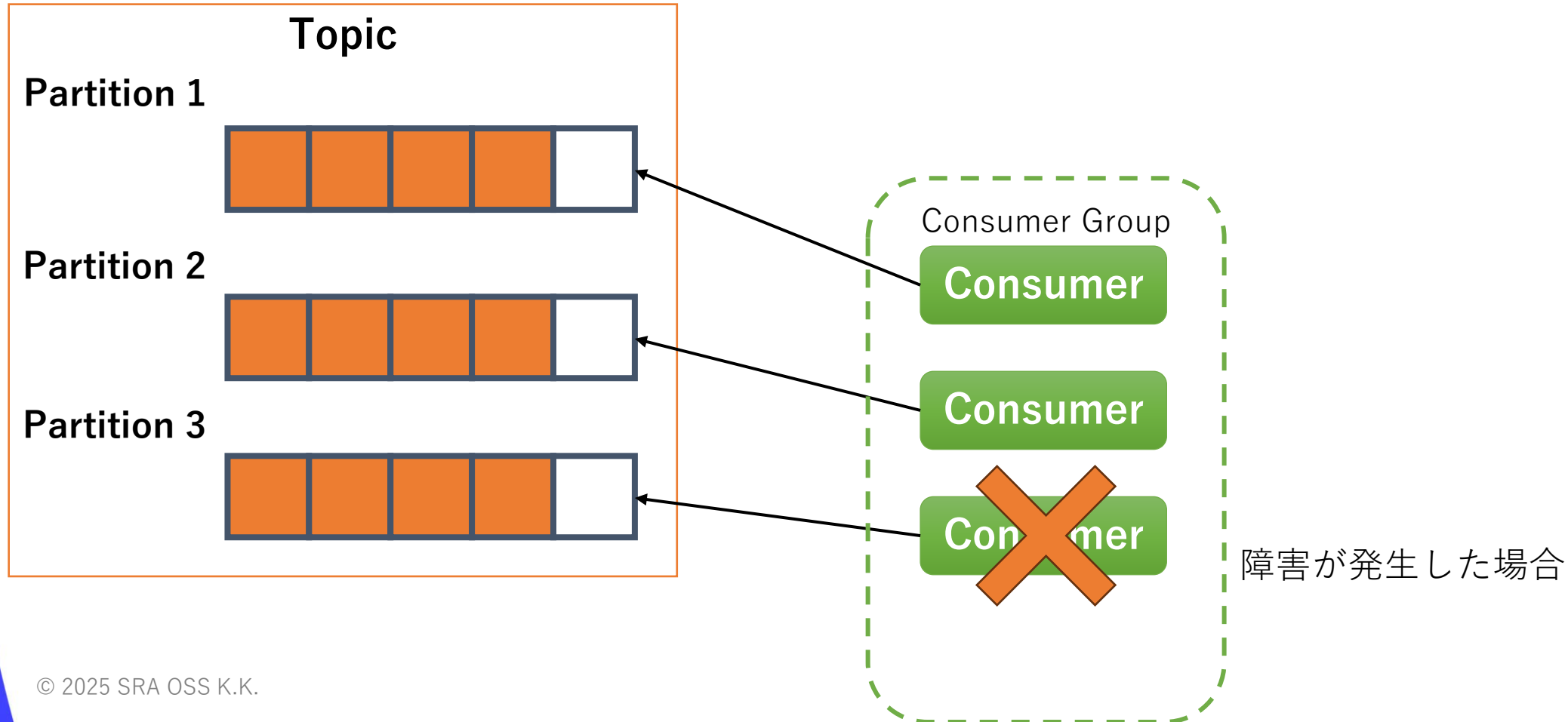
- Controller が Kafka クラスタを管理する
  - Topic の作成・削除
  - Partition 数の変更
  - Leader の選定
  - Replica の配置
- 管理には Kraft という
  プロトコルが使われる
  （かつては zookeeper という別の
  ソフトウェアが使われていた）
- Controller は複数で
  分散協調して動作する
  - Broker とのやり取りは
    Leader が行う
  - Follower はLeader の障害に備える



© 2025 SRA OSS K.K.

Consumer Group リバランス

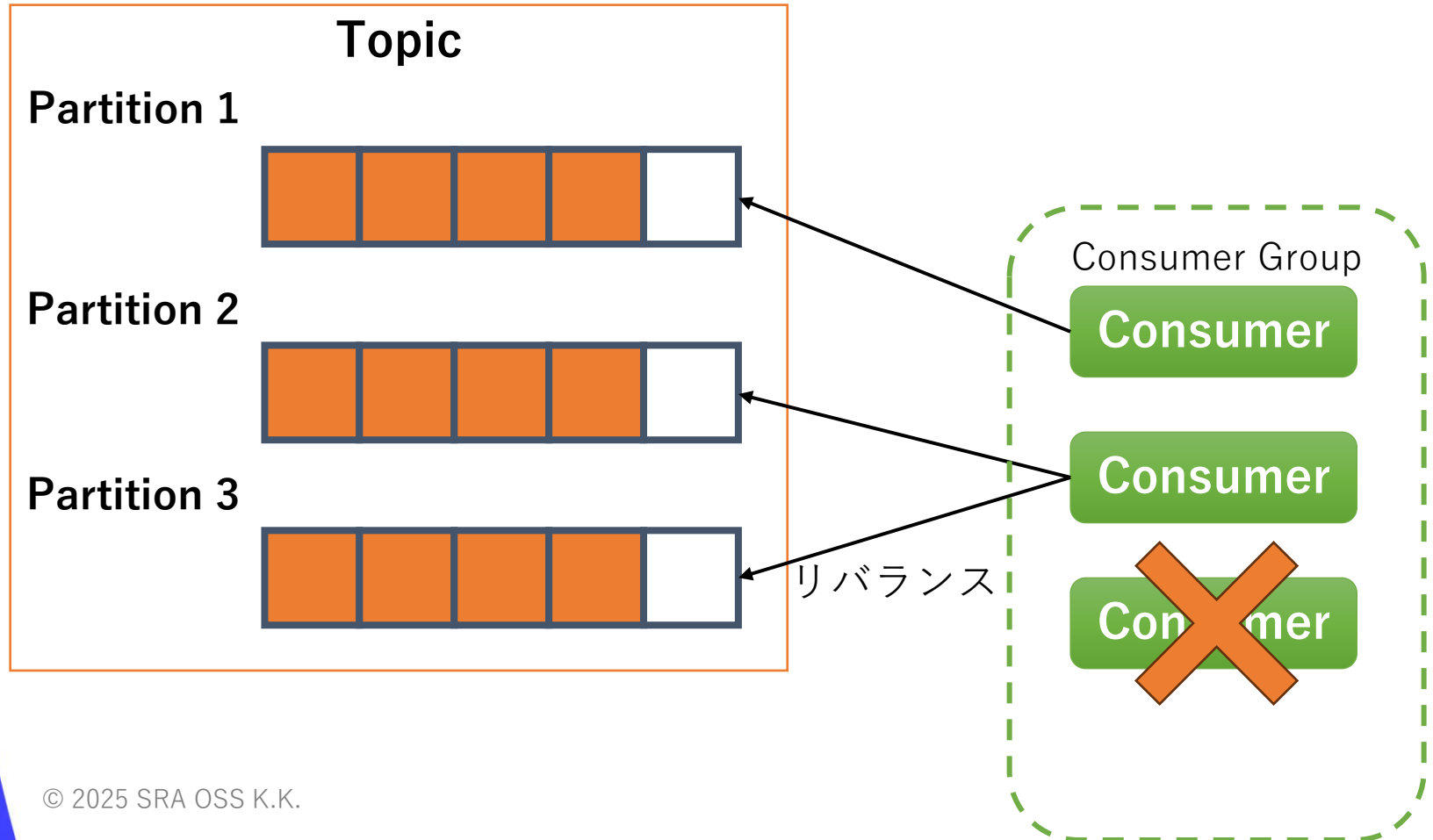
- Consumer の増減があった時に Partition のアサインをし直す機能



© 2025 SRA OSS K.K.

Consumer Group リバランス

- Consumer の増減があった時に Partition のアサインをし直す機能

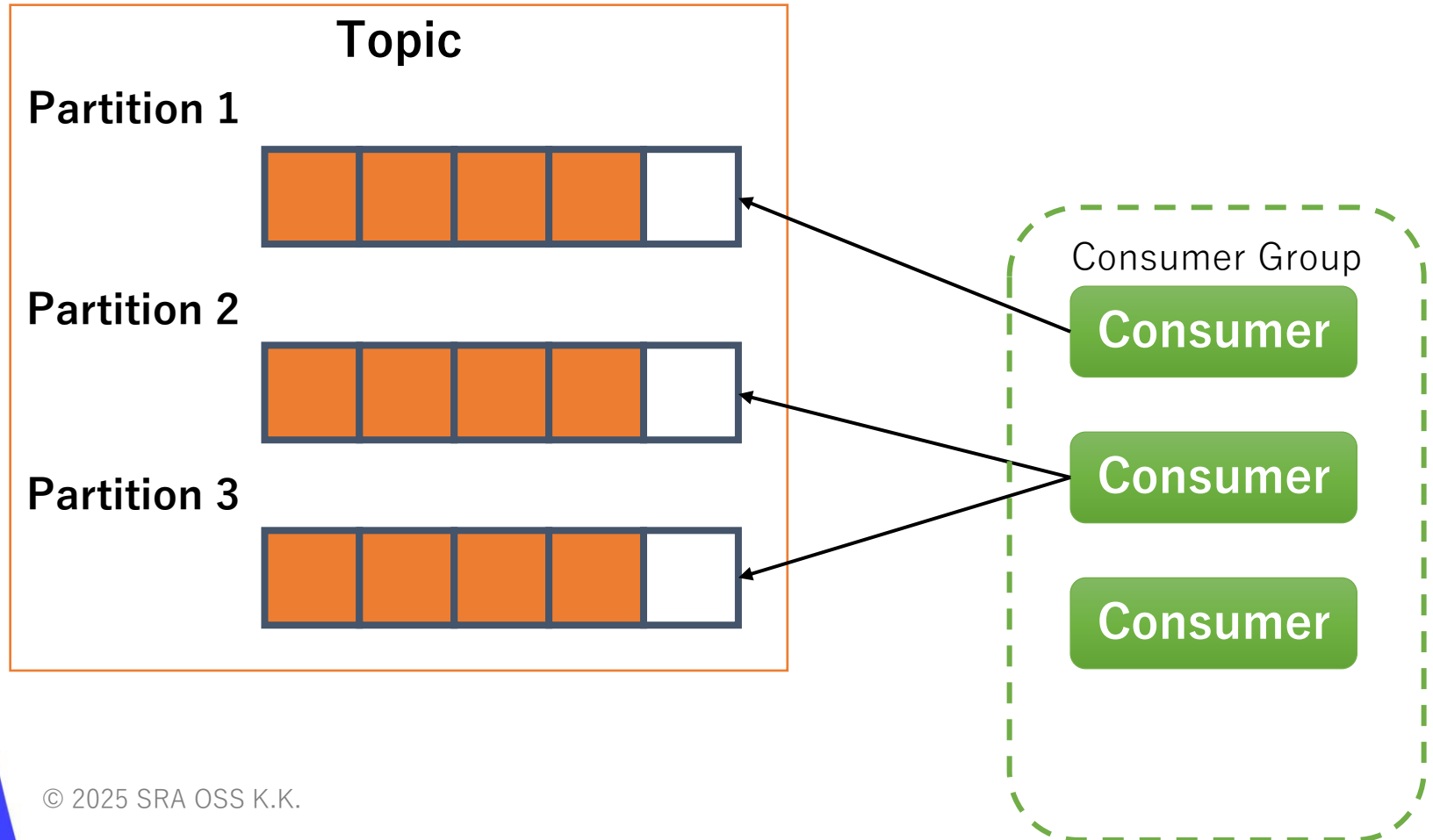


障害が発生した場合
→リバランスによって処理継続できる

© 2025 SRA OSS K.K.

Consumer Group リバランス

- Consumer の増減があった時に Partition のアサインをし直す機能

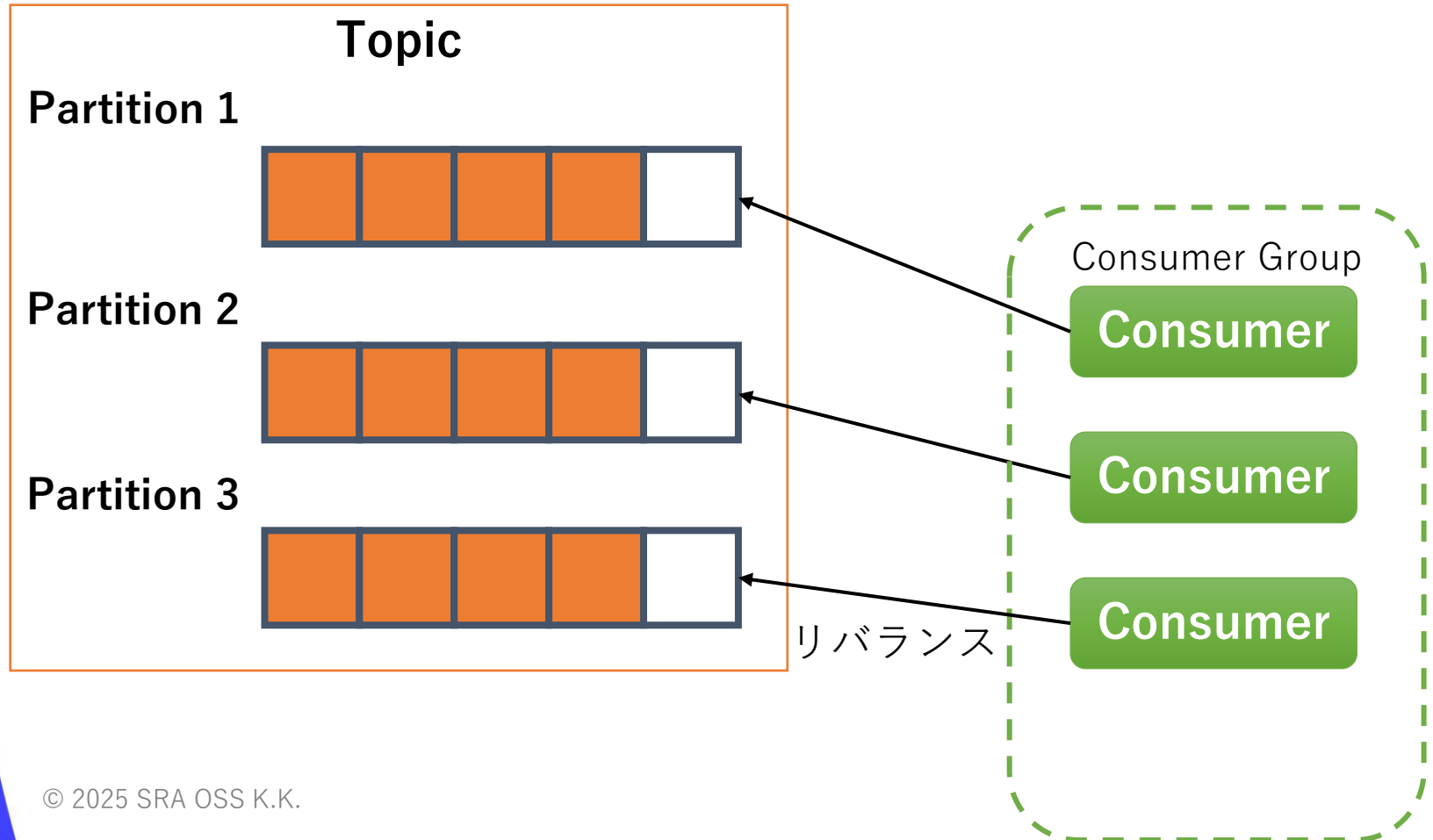


追加（or 復帰）があった場合

© 2025 SRA OSS K.K.

Consumer Group リバランス

- Consumer の増減があった時に Partition のアサインをし直す機能



追加（or 復帰）があった場合
→リバランスによってスケールアウト

© 2025 SRA OSS K.K.

送達保証

© 2025 SRA OSS K.K.

セマンティクス

今回はこちら二つに関する
Kafka の機能を紹介します

- Message の配信保証
  - At least once：Message は失われないが、再送される可能性がある
  - At most once：Message は失われる可能性があるが、再送はされない
  - Exactly once：Message は抜けや重複なく1度だけ配送される

- Kafka では設定および実装によってどのセマンティクスも実現可能

- どのセマンティクス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

© 2025 SRA OSS K.K.

Acks の利用

At least once を実現する際、再送すべきかの判断は Ack でできる

- 成功の場合: 再送しなくて問題なし
- 失敗 or 返ってこない場合: 再送

acks 設定と送達保証

- acks=0:
  まったく Ack を返さないためデータロストしうる(at most once)
- acks=1:
  Leader への到達時点で返すため
  Leader 障害がなければデータロストしない(at least once)
- acks=all:
  全ての ISR への到達時点で返すため設計限度内の障害であれば
  データロストしない(at least once)

※これらは再送の処理など、実装を適切に行う場合。
Acks の設定だけで実現できるわけではない。

2025 © SRA OSS K.K.

Offset Commit の利用

- Consumer 障害時、
  Offset Commit が適切に行われていれば復旧時どこから
  読み込みを再開すればよいか分かる
  →データロストを防げる

- Message を受け取ってすぐ実施: at most once
- Message 受信後処理が終了した時点で実施: at least once

（Auto Offset Commit では Offset Commit タイミングが制御できないため
データロストも重複も発生しうる）

© 2025 SRA OSS K.K.

Kafka クラスタ監視概要

© 2025 SRA OSS K.K.

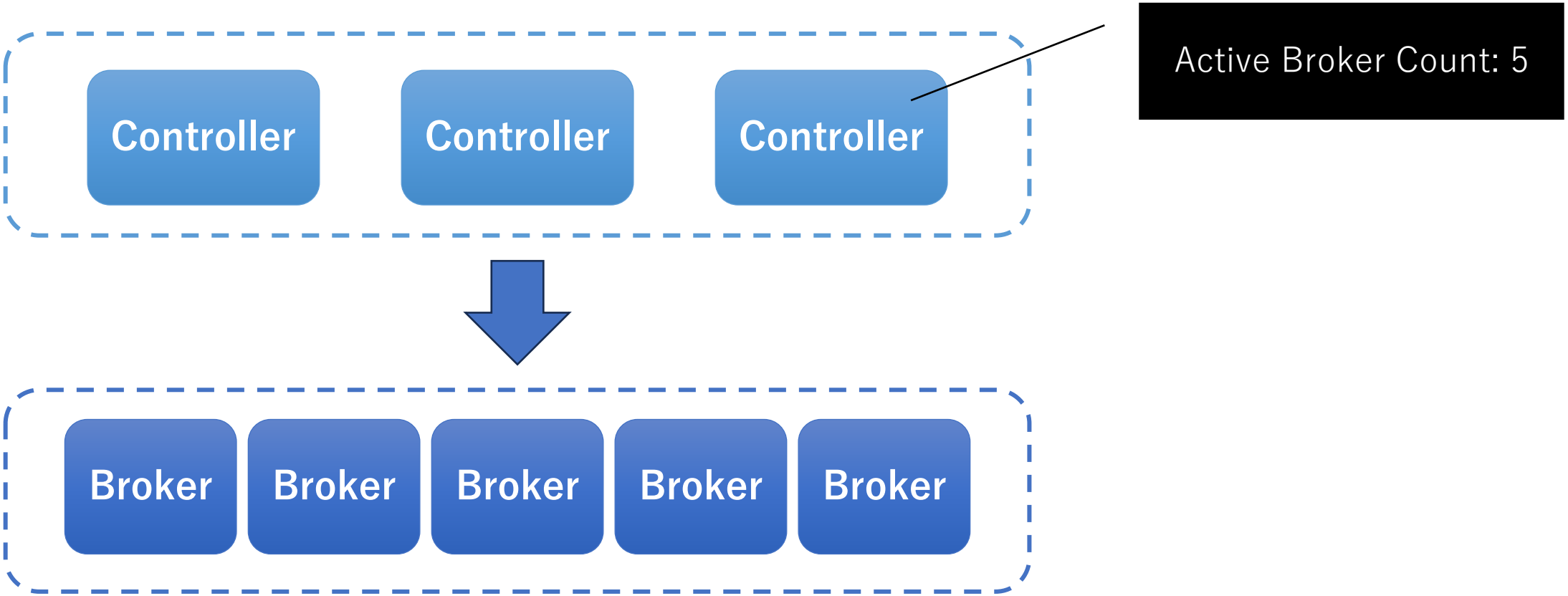
JMX について

- リアルタイムに処理をし続けるために、
障害発生時即座に対応できるよう監視を行うことは重要
→Kafka ではこうした監視のために JMX を利用する
- JMX
  - Java アプリケーションを監視するための機能
  - ヒープなど、JVM の情報も取れるし、
アプリケーション側で実装すれば任意の情報を外部から取れるようにできる
  - Zabbix や Prometheus といった監視ツールも対応しているので、
取得した値をグラフにしたり、アラートを上げるトリガーを設定して監視できる
- Kafka では Broker、Controller、Producer、Consumer などの情報を取得できるようになっている
  - デフォルトでは取れないので簡単な設定は必要
  - 利用可能な値（メトリクス）は Kafka 公式ドキュメントで確認可能

© 2025 SRA OSS K.K.

メトリクス例: Active Broker Count

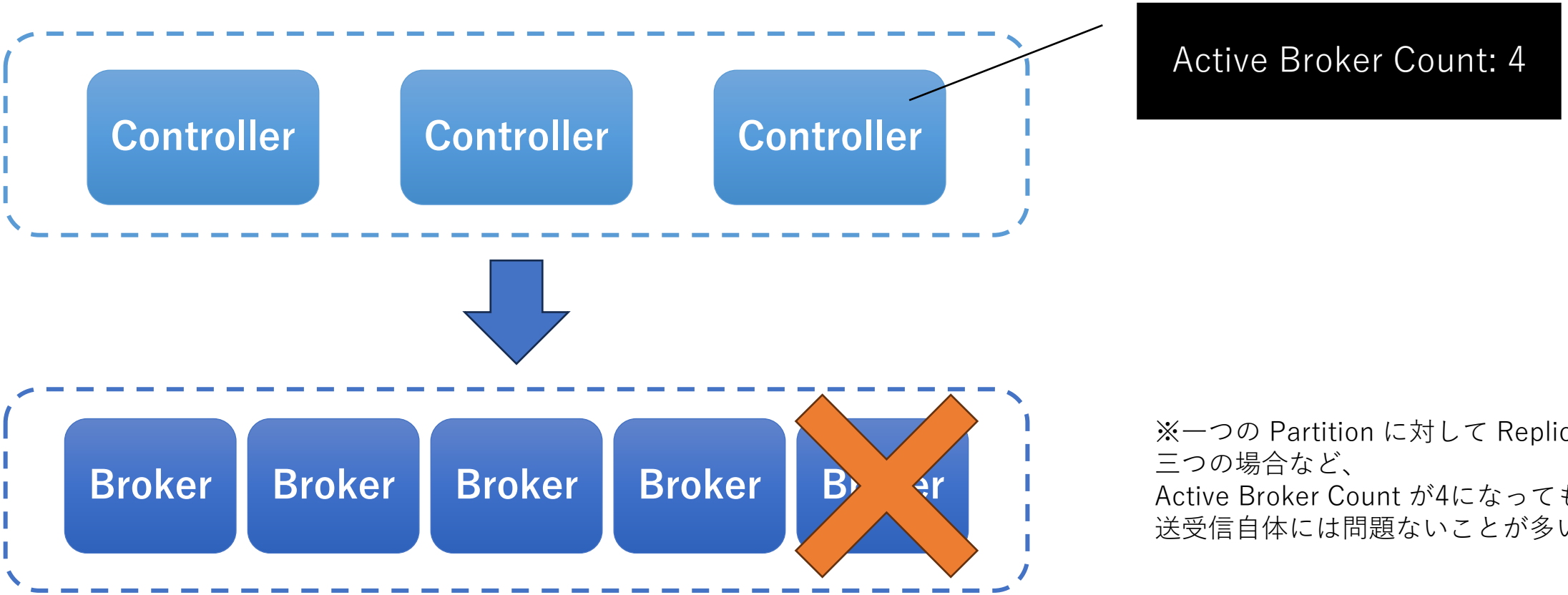
- Controller から見たアクティブなBroker の数
- 例えば Broker 5台でクラスタを組んでいるなら、
  5未満でアラートを上げるとBroker の異常を検知できる



© 2025 SRA OSS K.K.

メトリクス例: Active Broker Count

- Controller から見たアクティブなBroker の数
- 例えば Broker 5台でクラスタを組んでいるなら、
  5未満でアラートを上げるとBroker の異常を検知できる



※一つの Partition に対して Replica が
三つの場合など、
Active Broker Count が4になっても
送受信自体には問題ないことが多い

© 2025 SRA OSS K.K.

Zabbix による Kafka 監視

- Zabbix テンプレート:
  - 監視する値やアラートを上げるトリガーなどの設定をまとめたもの
  - Zabbix 公式が公開しているものもあり、自作も可能
- Apache Kafka by JMX
  - Zabbix の Kafka 監視用テンプレート
  - Apache Kafka 2.6.0 で動作確認されており、最新版では取得できない値がある

Apache Kafka by JMX を適用したときの Zabbix 上表示

例えば zookeeper（古いバージョンの Kafka で使われていた）関連の値は取得できない

kafka2-controller	ZooKeeper client request latency	component: zookeeper	グラフ
kafka2-controller	ZooKeeper connection status	Object or attribute not found: kafka.server:type=ZooKeeperClientMetrics,name=ZooKeeperRequestLatencyMs	
kafka2-controller	ZooKeeper disconnect rate	component: zookeeper	グラフ
kafka2-controller	ZooKeeper readonly rate	component: zookeeper	グラフ
kafka2-controller	ZooKeeper session expiration rate	component: zookeeper	グラフ
kafka2-controller	ZooKeeper sync rate	component: zookeeper	グラフ

→4系での値を取得できるようにしてみる

© 2025 SRA OSS K.K.

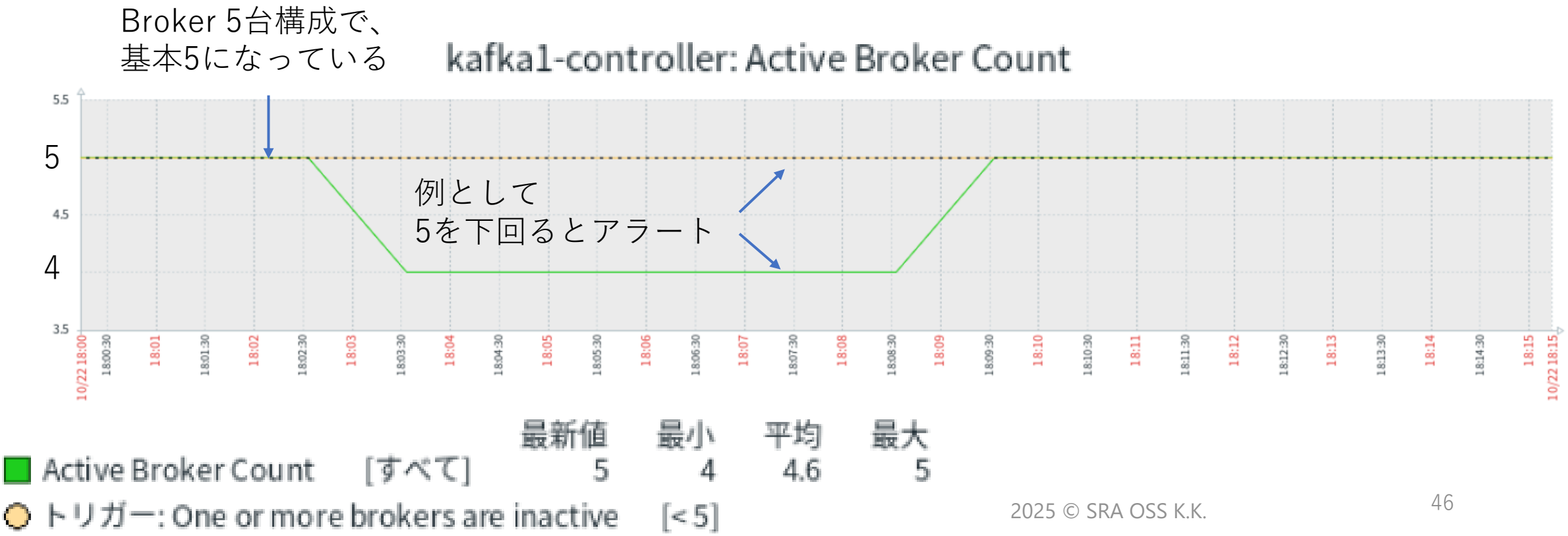
追加するメトリクス例

- Active Broker Count: アクティブな Broker の数
- Active Controller Count: アクティブな(=リーダーの)Controller の数
- Average Election Latency: Controller リーダー選挙の平均所要時間
- Maximum Election Latency: Controller リーダー選挙の最大所要時間
- Number Of New Controller Elections: Controller のリーダーに選ばれた回数

© 2025 SRA OSS K.K.

Active Broker Count 監視追加

- テンプレートで取っていなかった値も設定すれば取得できることを確認



2025 © SRA OSS K.K.

まとめ

© 2025 SRA OSS K.K.

Kafka 入門まとめ

- ストリーム処理を行うための分散メッセージングシステム
  - 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
- 耐障害性、スケーラビリティ
  - Partition を複数の Broker で複製し、耐障害性を持たせられる
  - Consumer Group を形成、Consumer 増減に対応できる
- 送達保証のための機能
  - Acks
  - Offset Commit
- JMX による監視
  - Zabbix の Kafka 用テンプレートはバージョンが古く、調整が必要

© 2025 SRA OSS K.K.

– 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。

2025 © SRA OSS K.K.