Amazon Managed Streaming for Apache Kafka

AWS Black Belt Online Seminar

Takayuki Enomoto

Solutions Architect

2023/04

AWS Black Belt Online Seminar とは

- 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナーシリーズです
- AWSの技術担当者が、AWS の各サービスやソリューションについてテーマごとに動画を公開します
- 動画を一時停止・スキップすることで、興味がある分野・項目だけの聴講も可能、スキマ時間の学習にもお役立ていただけます
- 以下の URL より、過去のセミナー含めた資料などをダウンロードできます
  - https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/

内容についての注意点

- 本資料では 2023 年 04 月時点のサービス内容および価格について説明しています。最新の情報は AWS 公式ウェブサイト(https://aws.amazon.com/)よりご確認ください
- 資料作成には十分注意しておりますが、資料内の価格と AWS 公式ウェブサイト記載の価格に相違があった場合、AWS 公式ウェブサイトの価格を優先いたします
- 価格は税抜表記となっています。日本居住者のお客様については、別途消費税が発生いたします

自己紹介

名前：榎本 貴之 (Enomoto, Takayuki)

所属：アマゾンウェブサービスジャパン
　　データ事業本部
　　サービスソリューションアーキテクト本部
　　シニアソリューションアーキテクト

経歴：インフラエンジニア @システムインテグレーター
-> インフラエンジニア @ゲーム会社
-> Cloud Support Engineer @AWS
-> **Solutions Architect @AWS**

好きなAWSサービス: OpenSearch, AWS Support
　Amazon QuickSight, Amazon Neptune,
　Amazon EventBridge, AWS Config, Amazon CloudWatch,
　Amazon Kinesis, **Amazon MSK**

本セミナーの対象者

現在 Kafka を運用されており、Amazon MSK の利用を検討されている方

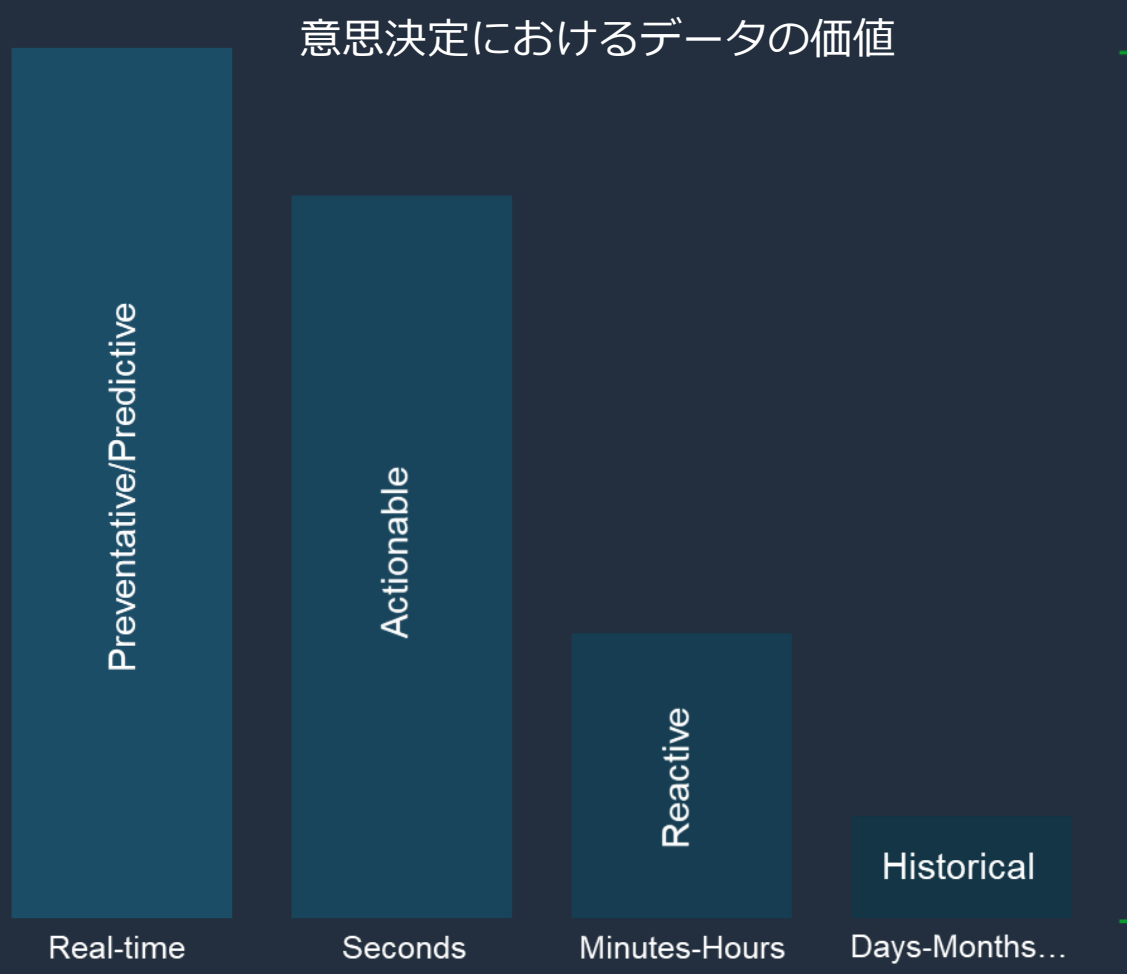
ストリーミングアプリケーション、ストリーミング ETL の構築を
検討されている方

アジェンダ

1. Apache Kafka の概要
2. Amazon MSK の概要
3. Amazon MSK の特徴

Apache Kafka 概要

データの価値は時間の経過とともに減少する



58% 顧客データの**リアルタイム分析**により、顧客維持率とロイヤルティが大幅に向上したと答えた回答者の割合

75% タイムリーでないデータがビジネスチャンスを妨げていると考えている調査対象企業の割合

https://www.forrester.com/report/Perishable+Insights+Stop+Wasting+Money+On+Unactionable+Analytics/-/E-RES135301#

https://hbr.org/resources/pdfs/comm/sas/Real.Time.Analytics.pdf

https://www.intersystems.com/news-events/news/news-item/survey-reveals-real-time-data-disconnect-leading-lost-business-opportunity/

データストリーミングは主流になりつつある

リアルタイム
Web 分析
ログ分析

トランザクション
イベントソーシング

メトリクス、
ログの集約

メッセージング

分離された
マイクロサービス

ストリーミング
ETL

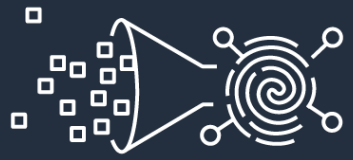
Apache Kafka

An open-source, distributed event streaming platform

- オープンソースの イベントストリーミング プラットフォーム
- リアルタイムアプリケーションのための 高スループット、低レイテンシーを提供
- 耐久性、分散処理、耐障害性を考慮して 設計されている
- 充実したエコシステム
- Fortune 100 企業の 80% 以上が信頼し、 Kafka を使用している

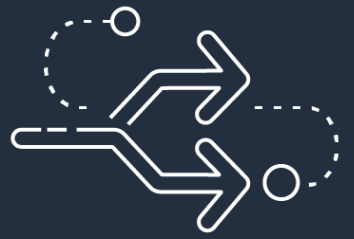
Apache Kafka のユースケース

メッセージング
(Pub-Sub)



ストリーミング処理

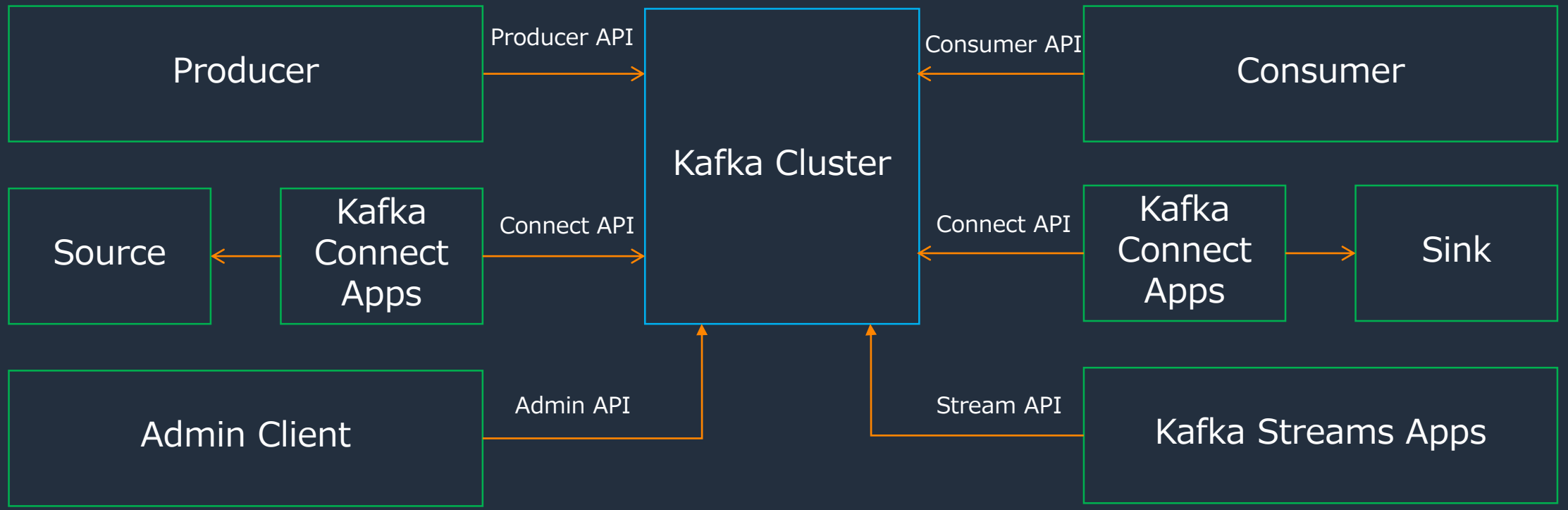
リアルタイム分析



トランザクション
イベントソーシング

Apache Kafka の概要

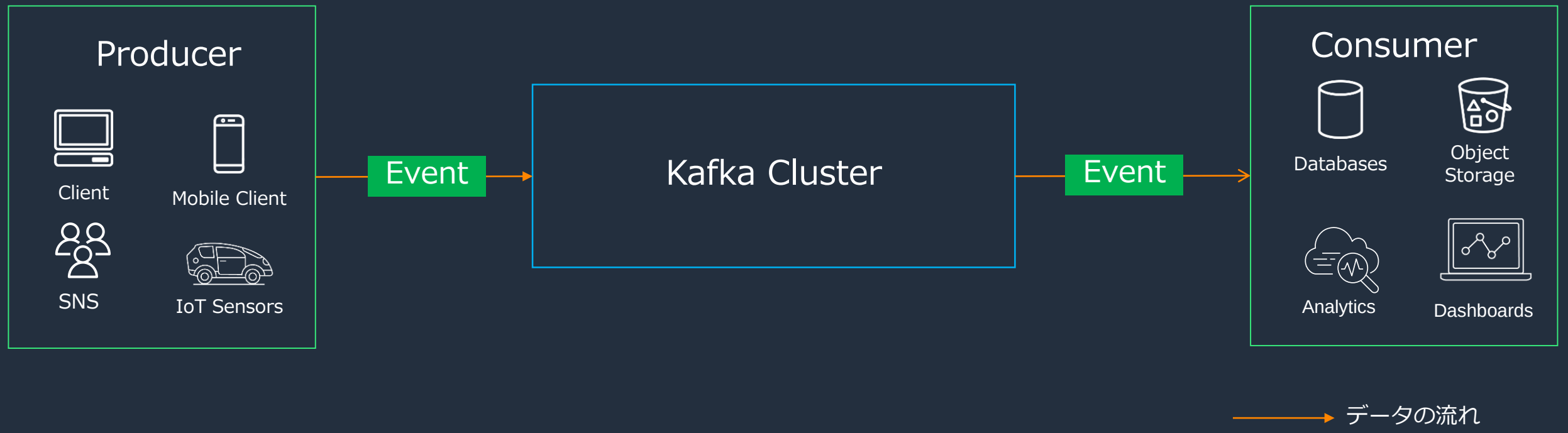
- Kafka を使用したストリームデータ処理環境は、ストリームデータを保持する Kafka Cluster と、用途に応じて異なる API を参照する複数のクライアントが存在する



データの流れ

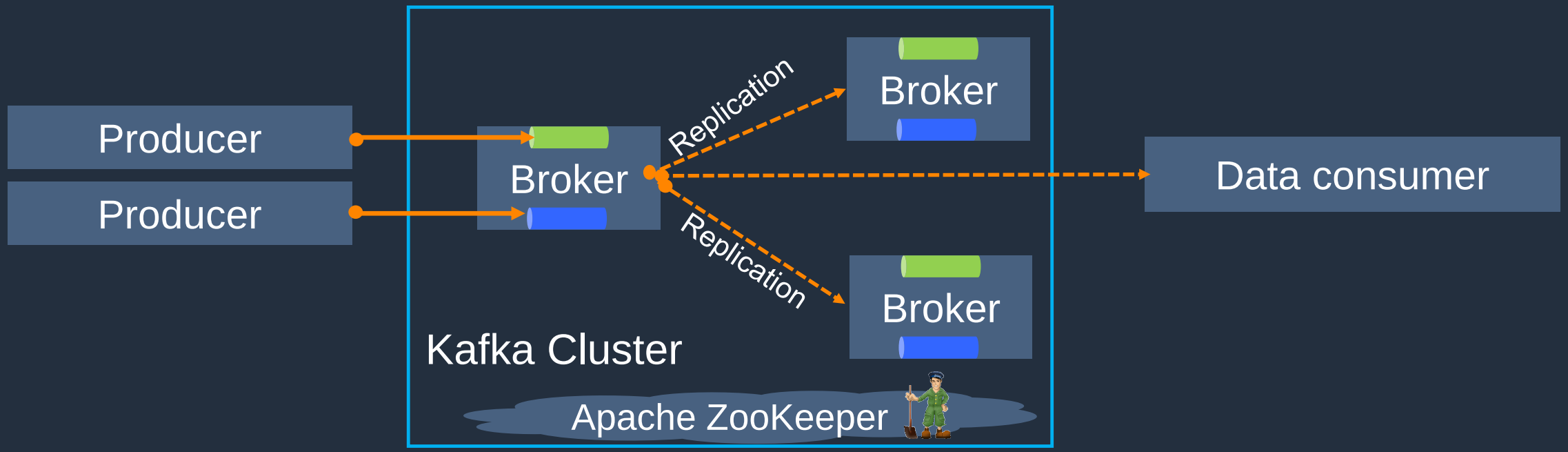
Apache Kafka によるイベントの送受信

イベントの Pub/Sub を行うクライアントとして、**プロデューサー**と**コンシューマー**を配置する。プロデューサーは**イベント**の発行（**パブリッシュ**）を行い、コンシューマーは**イベント**の購読（**サブスクライブ**）を行う



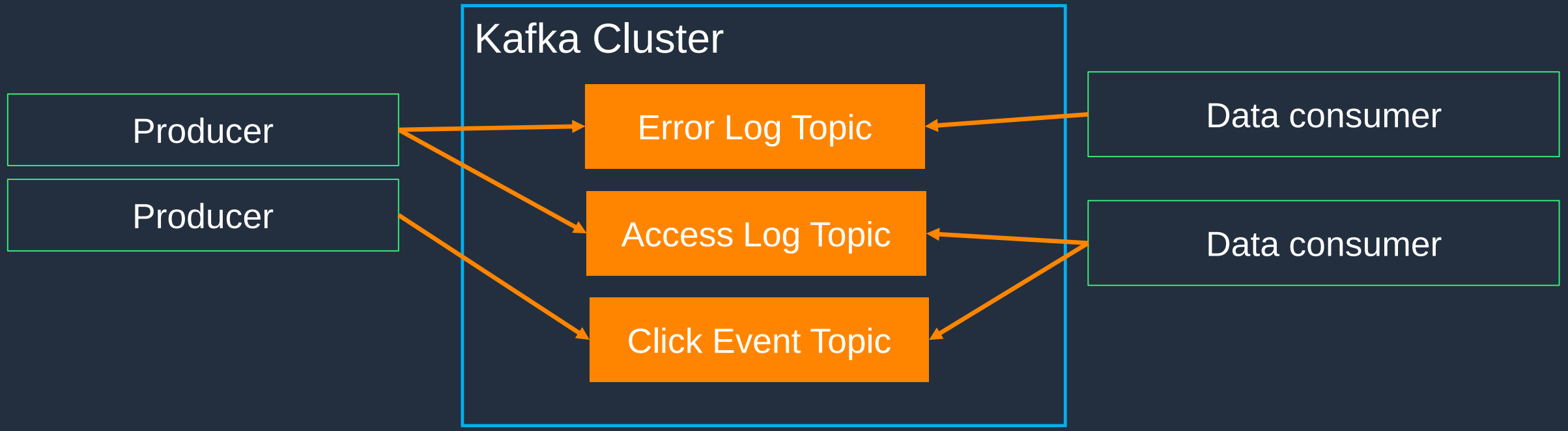
クラスター

- クラスター内に複数の Broker をクラスターに配置することで、拡張性と可用性を実現している
- メタデータやクォーラムは ZooKeeper 、Broker 内の Kafka Raft (Kafka 3.3.1 より Production Ready) のいずれかで管理される



トピック

- トピックは関連する**イベント**を収集する際に用いる一つの単位
- Kafka では**イベント**の種別に応じてクラスター内に複数のトピックを作成することができる
- トピックは複数の**パーティション**で構成されており、メッセージの発行・購読は
  トピック内のパーティションを指定して行う

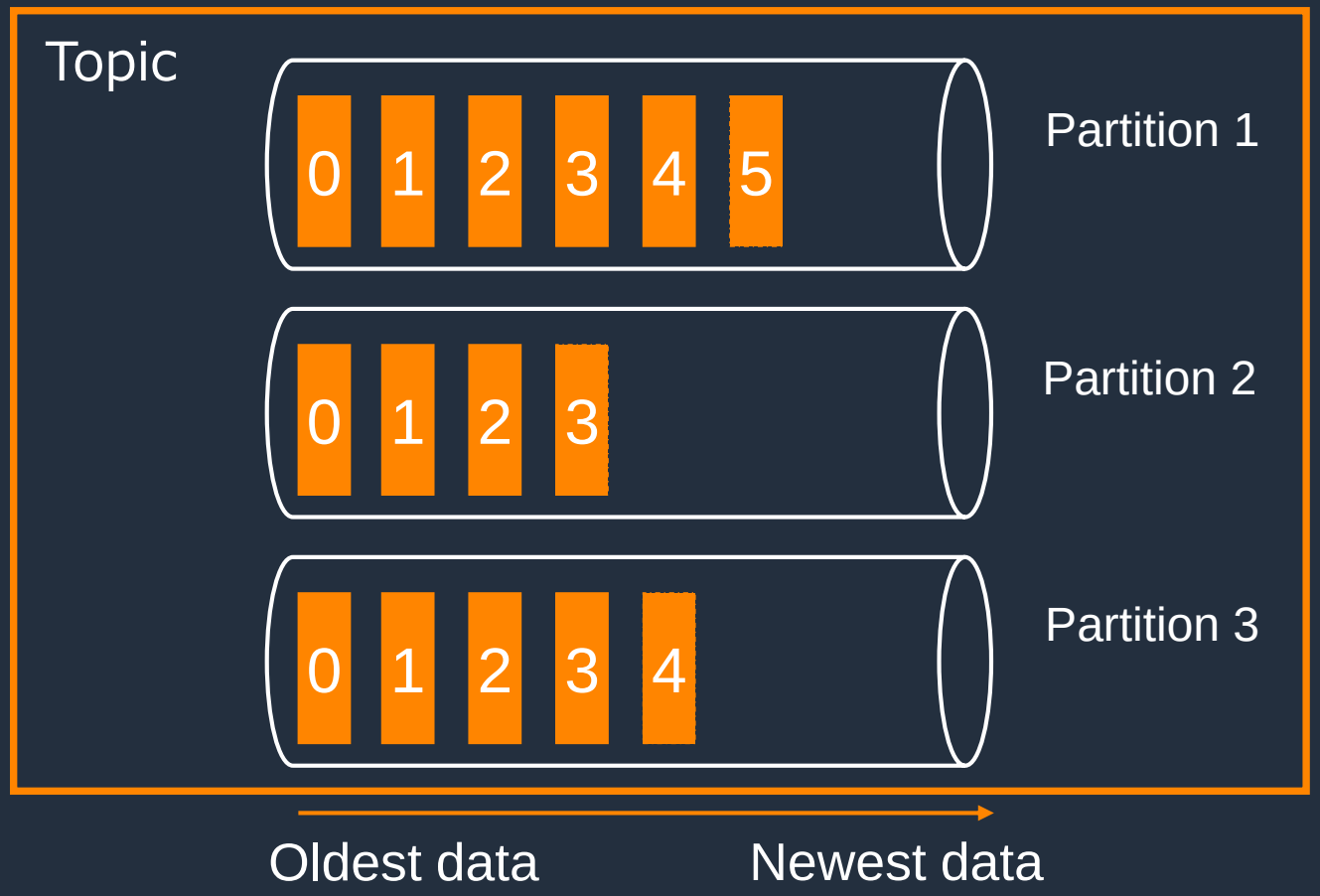


パーティション

- トピックを構成する一単位。トピック内で一意の ID が割り当てられている

パーティション

- トピックを構成する一単位。トピック内で一意の ID が割り当てられている
- パーティションはオンラインで追加することが可能。パーティションの削除は不可



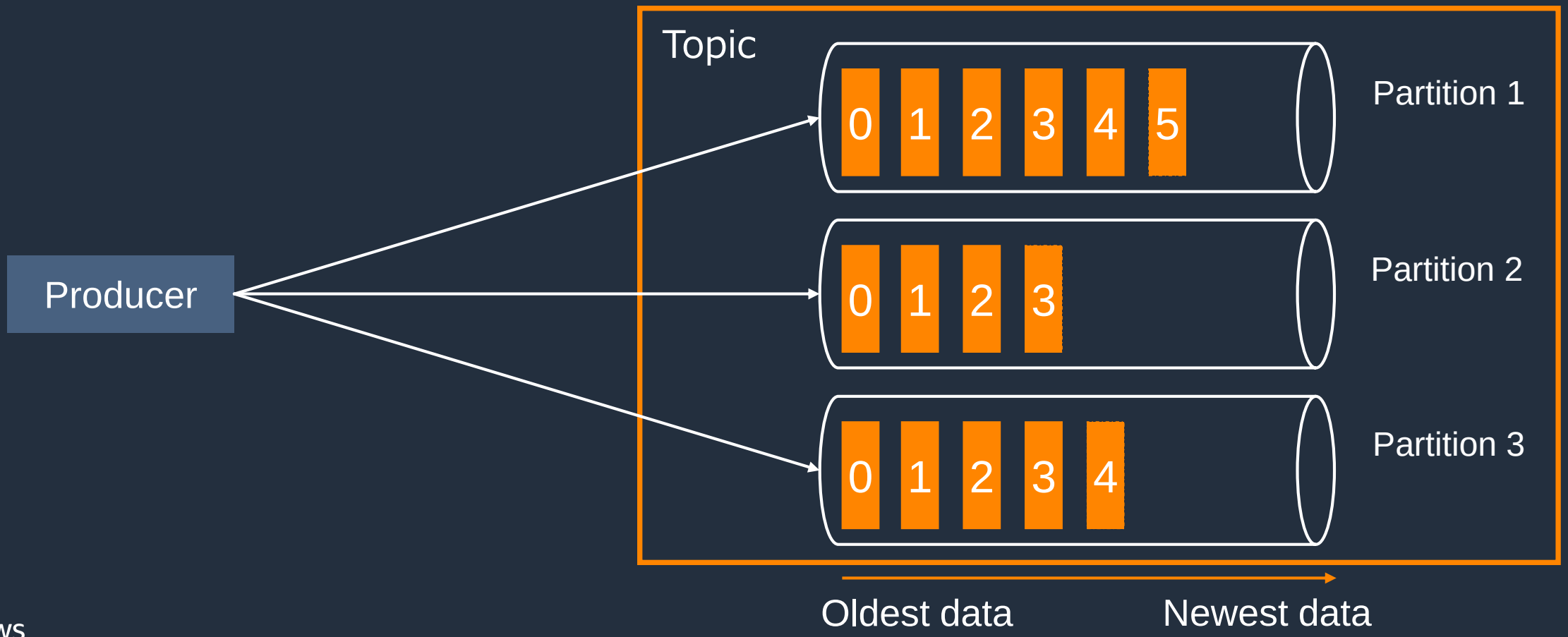
トピックの設定

- トピックごとに、メッセージ保管期間、保管期間に到達した場合の対応(削除or圧縮)、レプリカの数、メッセージサイズなどを変更可能
- トピック側で指定しない設定についてはブローカーにセットした設定が使われる

https://kafka.apache.org/documentation/#topicconfigs

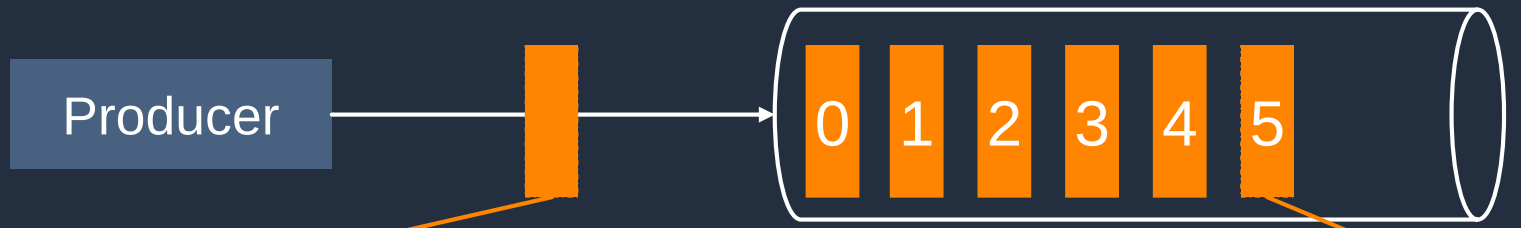
プロデューサー

- トピック内のパーティションに対して**イベント**を送信するコンポーネント
- 書き込み先のパーティションは、イベント内のメタデータをベースに決定される



イベント

- パーティションに書き込まれるデータの最小単位。メッセージ、レコードとも呼ばれる
- プロデューサー、トピック、パーティションに加えてキー情報などを含む
- 書き込まれたイベントに応じて、Kafka は**オフセット**やタイムスタンプなどのメタデータを追加で付与する
- メッセージサイズ上限はデフォルトで 1 MiB。ブローカーまたはトピックレベルで増減可能



イベントに含まれる情報

必須: **Topic**, **Value**(メッセージ本文)
オプション: Key, Partition, Timestamp, Headers(カスタムヘッダ)

イベントに含まれる情報:
Topic, **Value**, Key, **Partition***, **Timestamp***, Headers

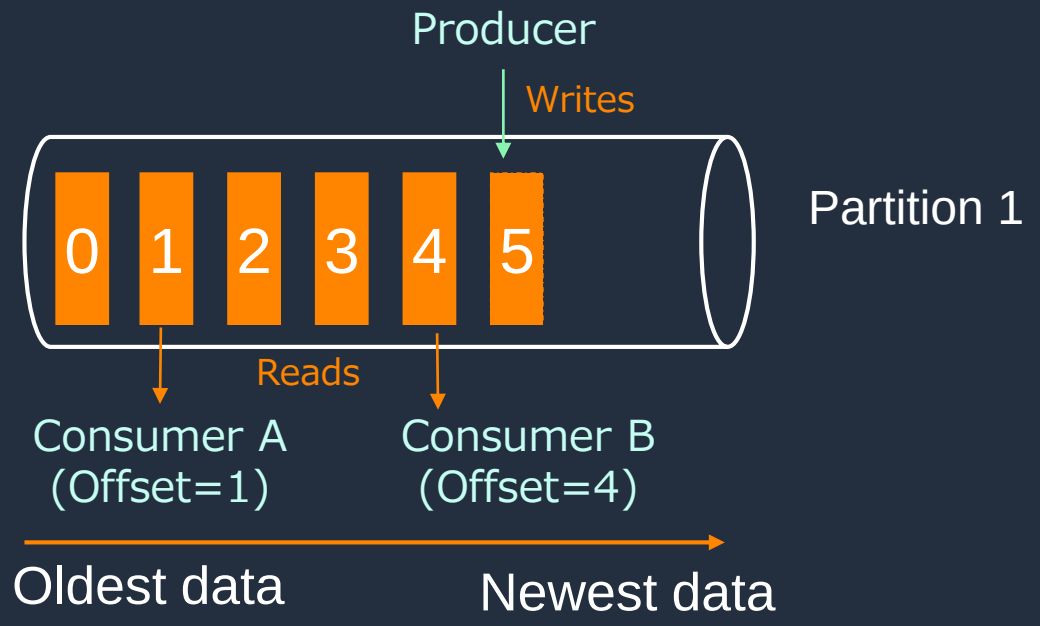
格納時に Kafka 側で付与するメタデータ: Offset

*イベントに含まれなかった場合も格納時に Kafka が付与する

https://kafka.apache.org/33/documentation.html#messageformat
https://kafka.apache.org/33/javadoc/org/apache/kafka/clients/producer/ProducerRecord.html

オフセット

- メッセージがパーティションに入れられた際に付与されるシーケンシャルな番号
- オフセットにより、コンシューマーは継続的に Kafka に保存されるメッセージのどこまで読み出したかを管理することが可能
- コンシューマーはイベントの処理状況=どのオフセットまで処理したかをオフセット管理用トピックに保存することで、何らかの理由で停止しても直前の状態から処理を再開できる



イベントの振り分け

- 格納先のパーティションは、イベントのメタデータに基づき決定される
  - 格納先のパーティション ID を直接指定
  - イベントに含まれる Key のハッシュ値を元にパーティションを決定
- キーやパーティション ID 未指定の場合はデフォルトの挙動が適用される。挙動はバージョンによって異なる。
  - ラウンドロビン (~2.3)
  - Sticky Partitioning (2.4~3.2): メッセージバッチをランダムに選出されたパーティションに配信
  - Strictly Uniform Sticky Partitioning(3.3~): メッセージバッチをランダムに選出されたパーティションに配信。キューにリクエストがたまっていたり、負荷で応答性が低下しているブローカーは対象から除外される
- この他に、ユーザーが独自に実装した割当ルールも利用可能

イベントのバッチ化と圧縮

以下のプロデューサー設定をチューニングすることで、メッセージの伝送効率が向上する可能性がある。レイテンシや追加リソースとのトレードオフとなるため、計測しながらチューニングを行うことを推奨

linger.ms

- イベントバッチを配信するまでの待機時間。デフォルトは 0 (使わない)

batch.size

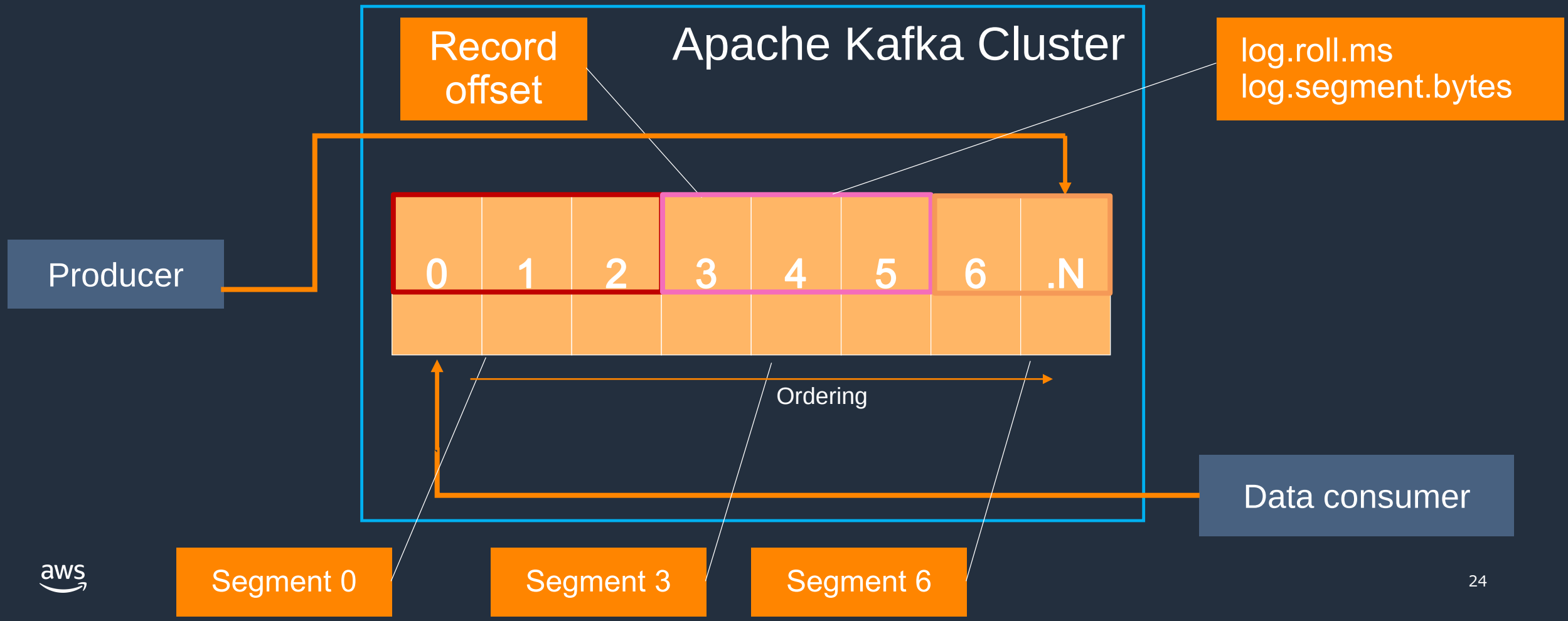
- イベントバッチを配信する際のバッチサイズの閾値。デフォルトは 16 KB

compression.type

- 圧縮無し、gzip、Snappy、LZ4 から選択可能
- トピックとプロデューサーで compression.type が異なる場合、ブローカーによる展開処理と再圧縮処理が実行されるため、スループットが低下する。双方の設定を揃える、もしくはトピックの compression.type を producer に設定することを推奨

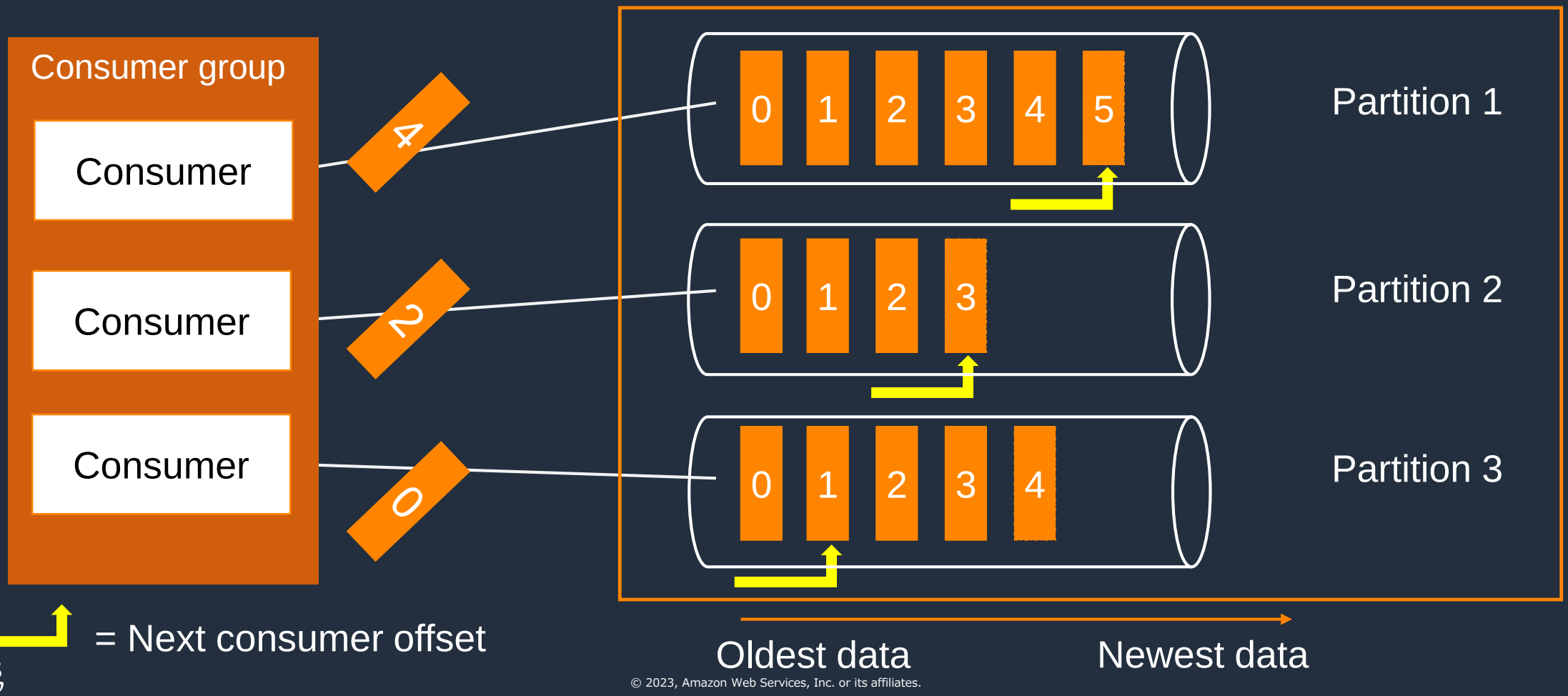
コンシューマー

- コンシューマーは開始位置(オフセット)を指定しイベントをサブスクライブする
- イベントはオフセットの番号順に取得されるため、パーティション単位で順序制御が可能



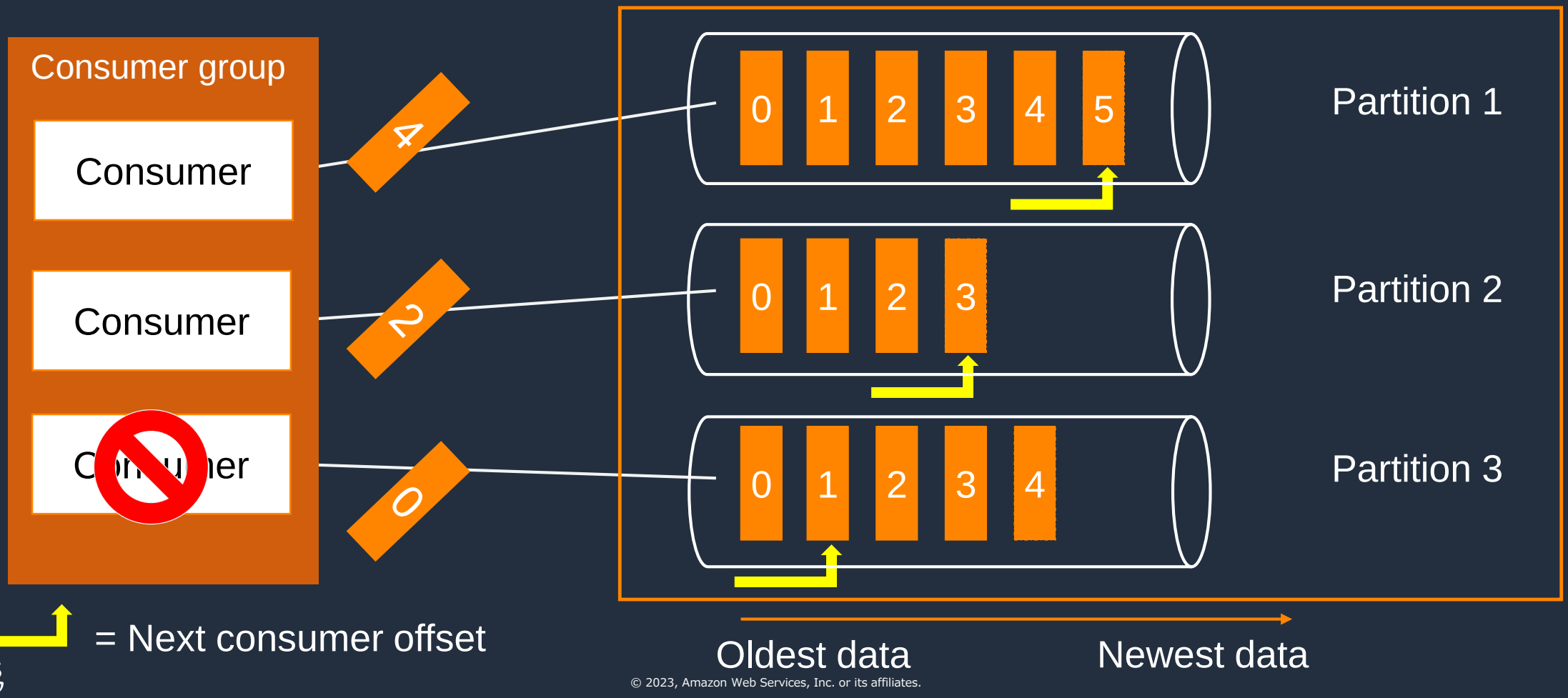
コンシューマーグループ

- Kafka の Consumer はグループを形成することで、データ取得処理をスケールさせている
- 特定の Consumer が特定の Partition を受け持つ形で処理を行う



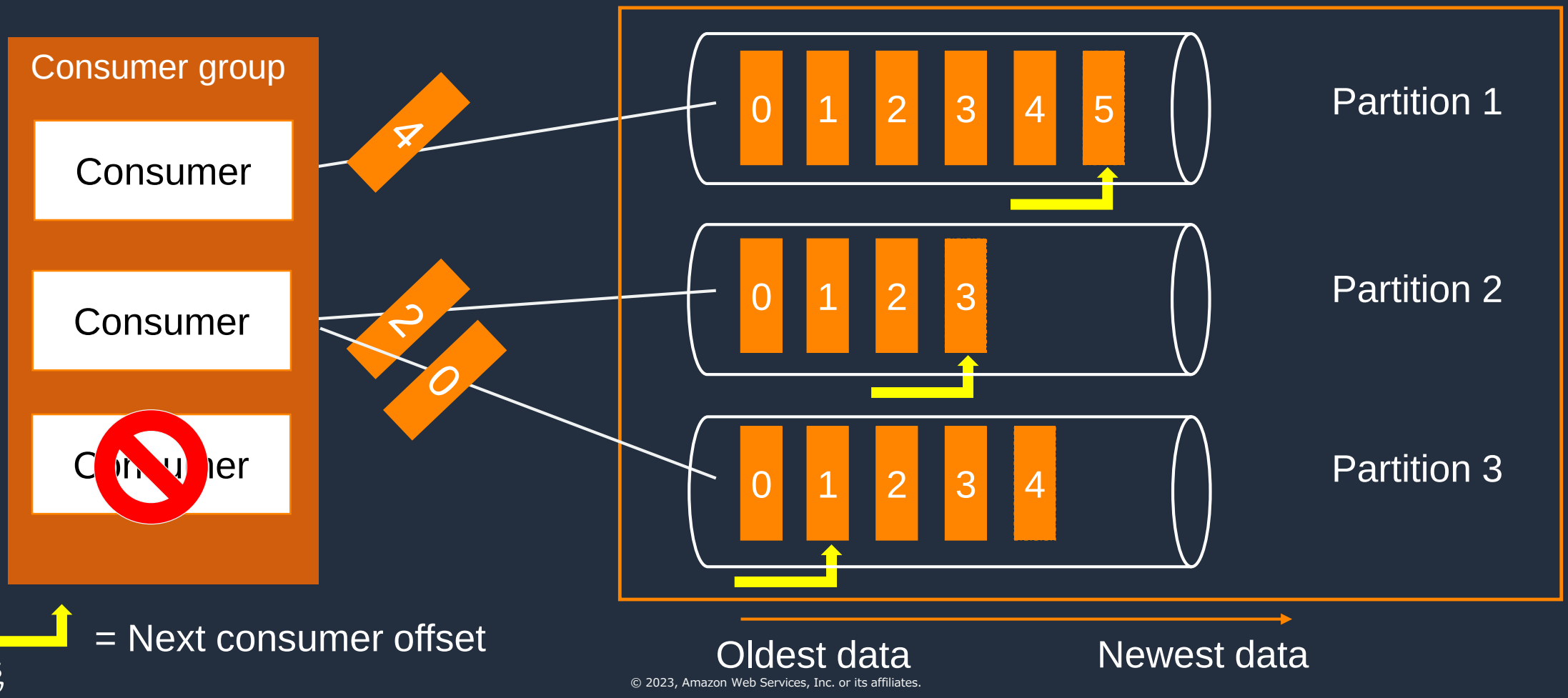
コンシューマーグループ

- 特定の Consumer で問題が発生した場合



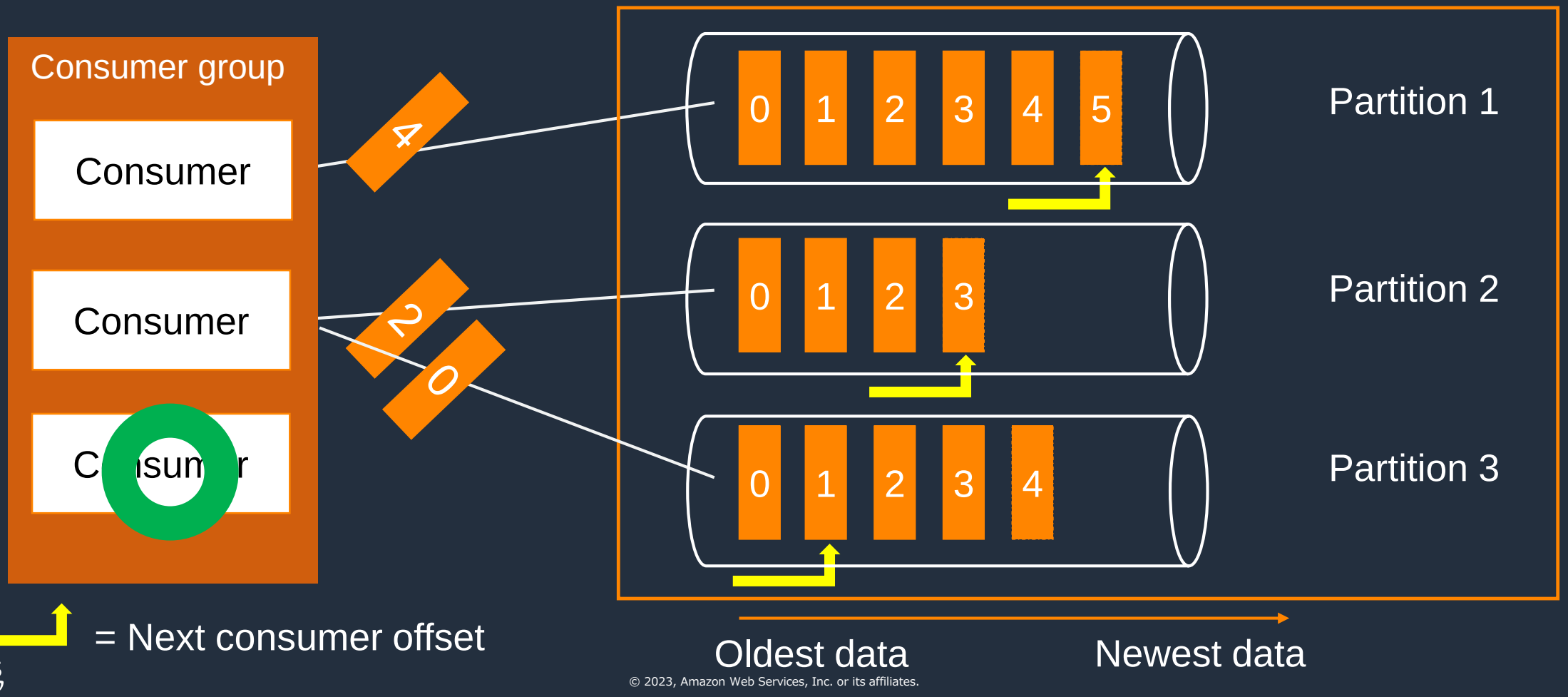
コンシューマーグループ

- 特定の Consumer で問題が発生した場合、生存している他の Consumer が処理を引き継ぐ



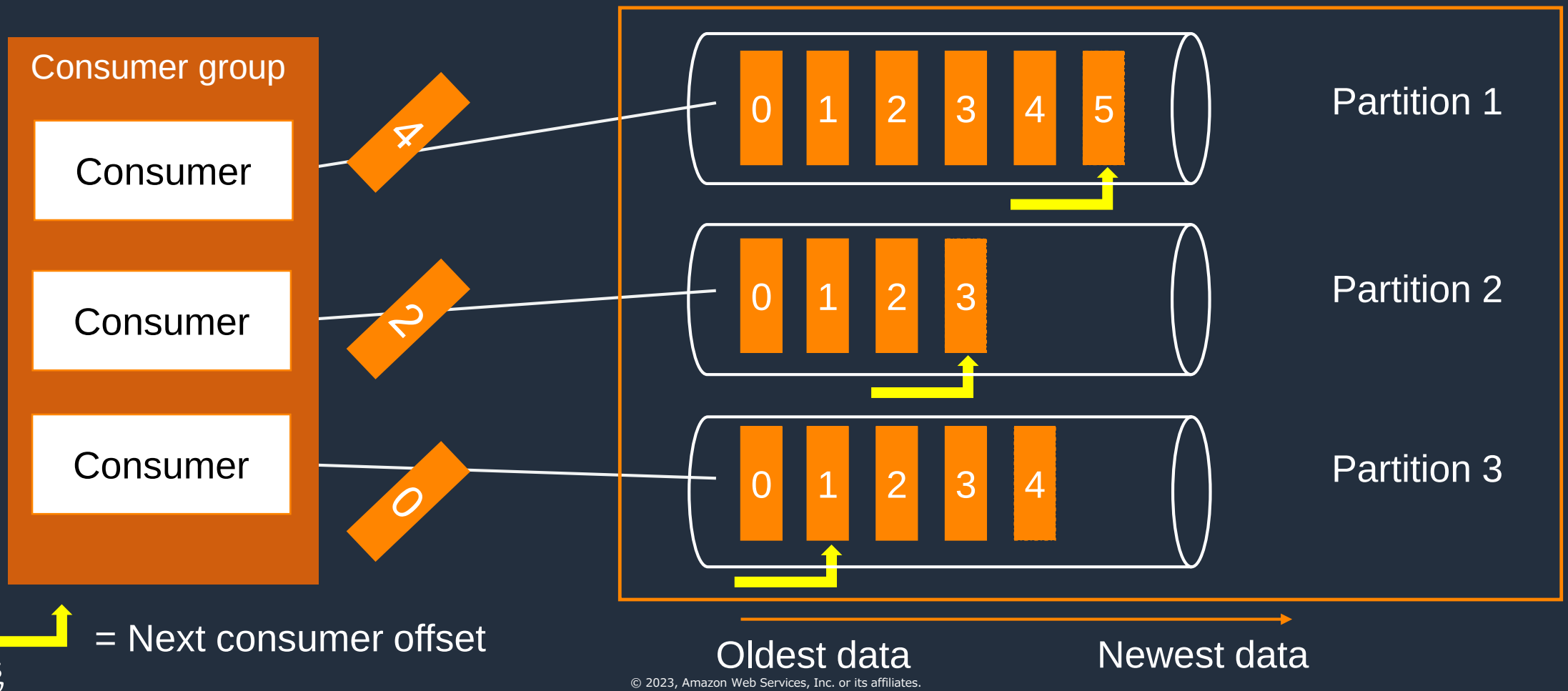
コンシューマーグループ

- 特定の Consumer で問題が発生した場合、生存している他の Consumer が処理を引き継ぐ
- Consumer が復旧すると、



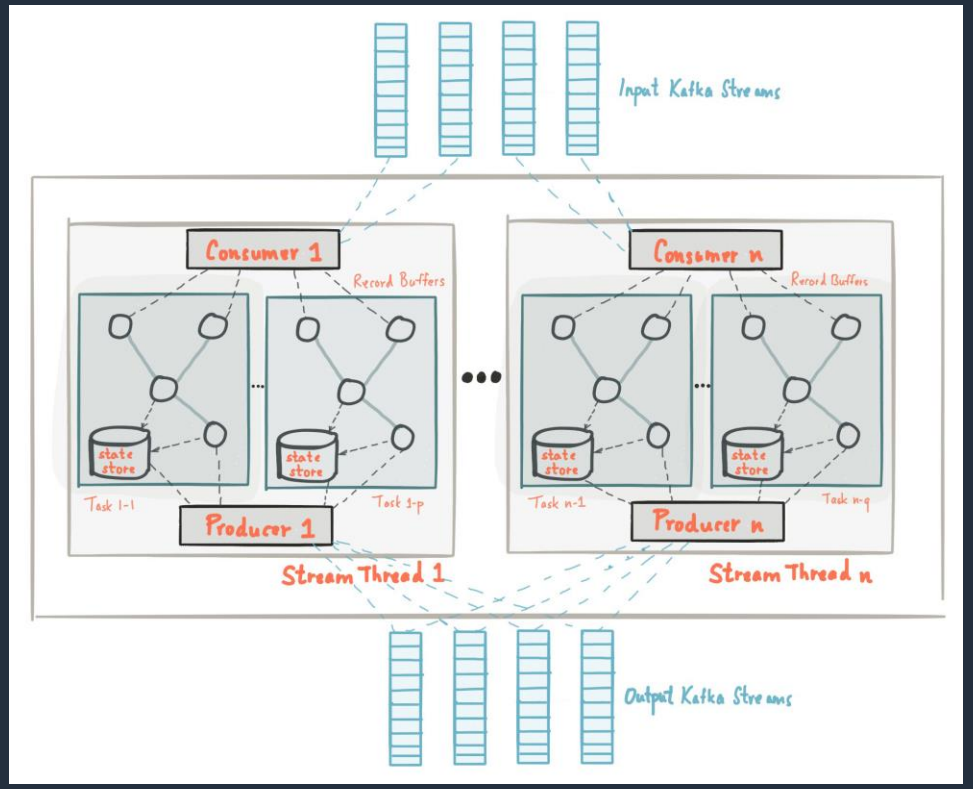
コンシューマーグループ

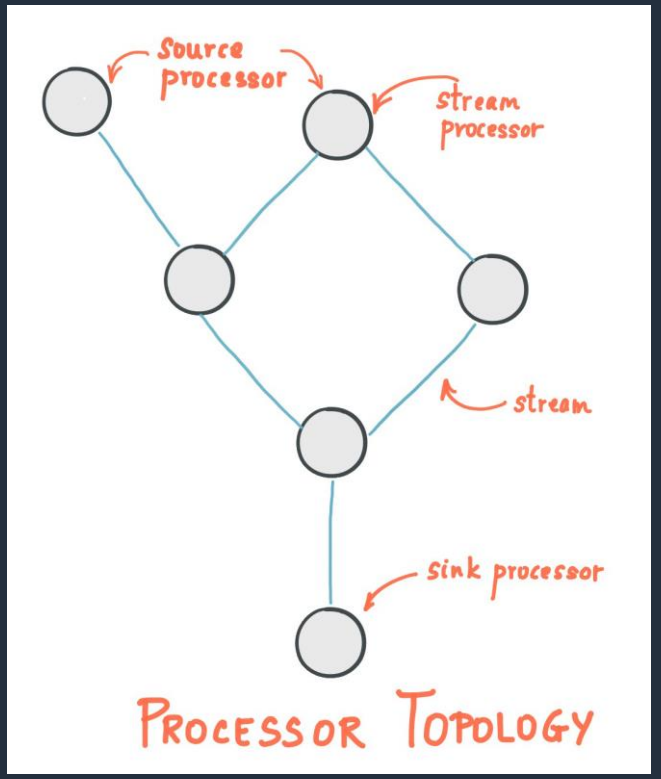
- 特定の Consumer で問題が発生した場合、生存している他の Consumer が処理を引き継ぐ
- Consumer が復旧すると、再度リバランスが行われる



Kafka Stream によるストリームデータ処理

- トピック内のデータに対する集計などのデータ加工処理を効率行うための仕組み
- Kafka が提供する Kafka Streams API と、ライブラリである Kafka Streams Client Library を組み合わせて実装するほか、Flink や Spark Streaming など他のストリーミング処理フレームワークとも連携可能





https://kafka.apache.org/33/documentation/streams/

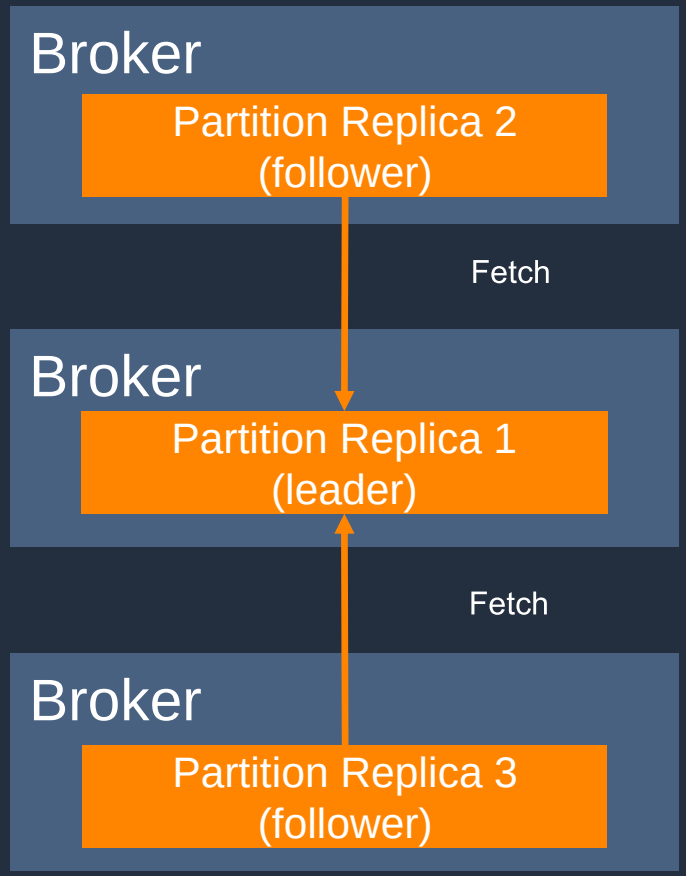
Kafka Connect によるデータストア間の連携

- Source、Sink の 2 つのデータストア間でデータを連携・同期する際に用いられる仕組み。
- MySQL や Redis、OpenSearch、Amazon S3 など様々なサービスやデータストア向けのコネクターを組み合わせることで、ローコードで異なるデータストア間の連携処理を実装することができる

レプリカによる冗長化

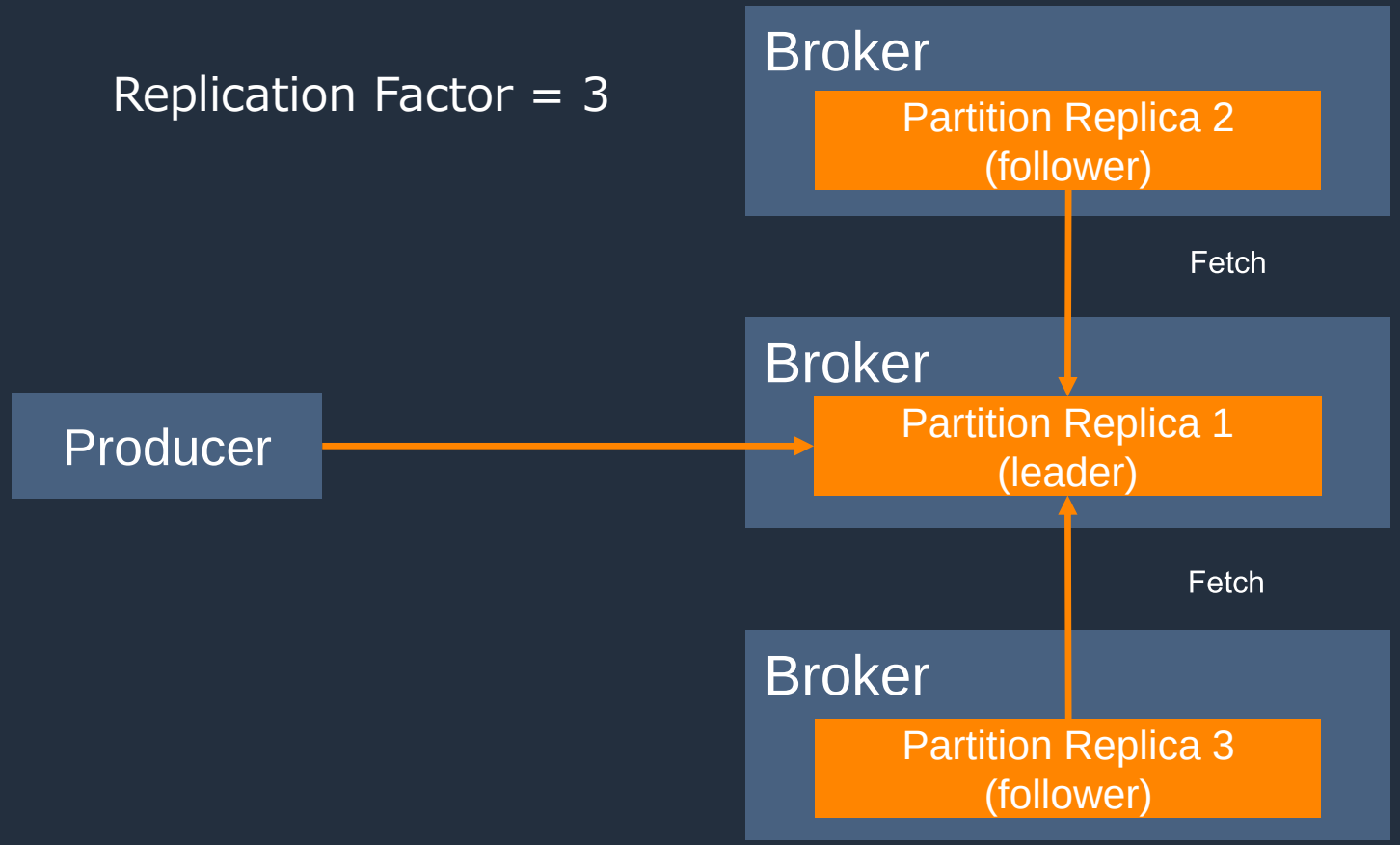
- トピックの可用性、メッセージの冗長性確保のために、パーティションのレプリカを設定
- Replication Factor と呼ばれる設定でレプリカ数を制御。3 以上を推奨

Replication Factor = 3



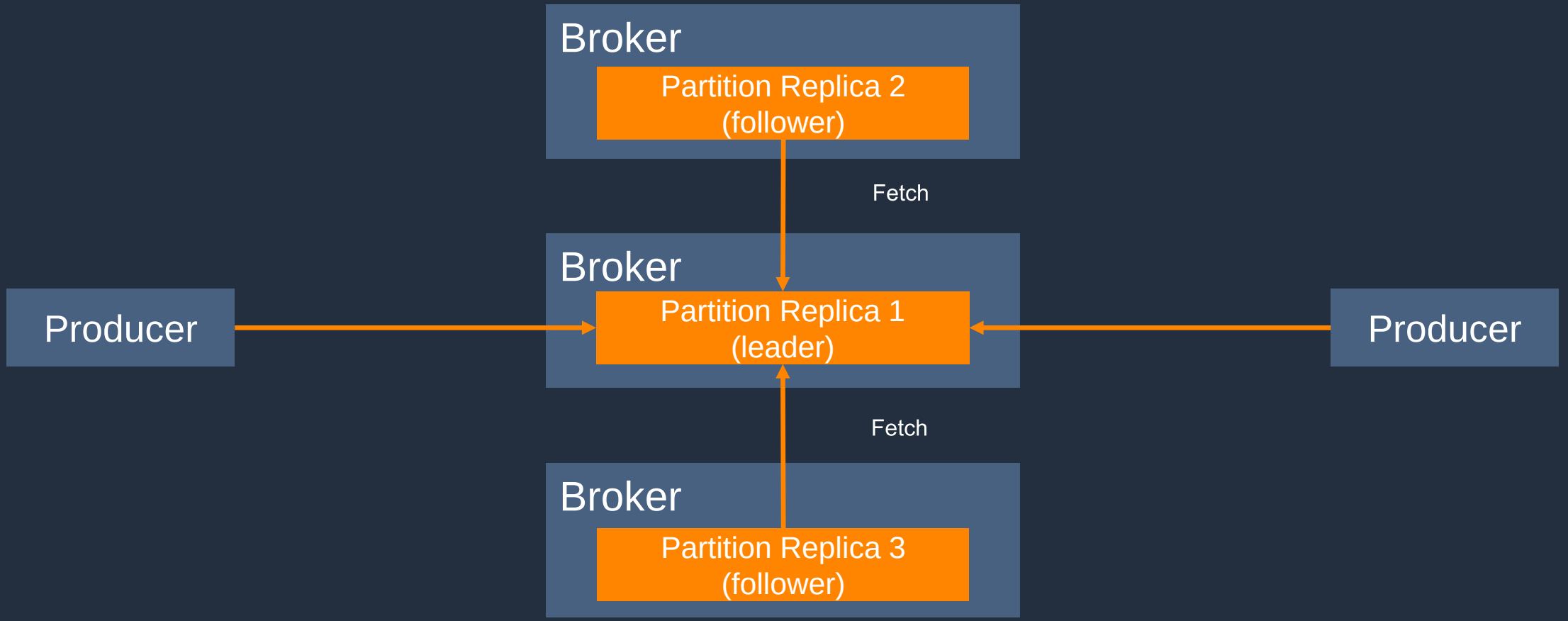
レプリカとプロデューサー

- Producer は必ず Leader パーティションに書き込む
- Follower パーティションは Leader パーティションからデータをコピーする



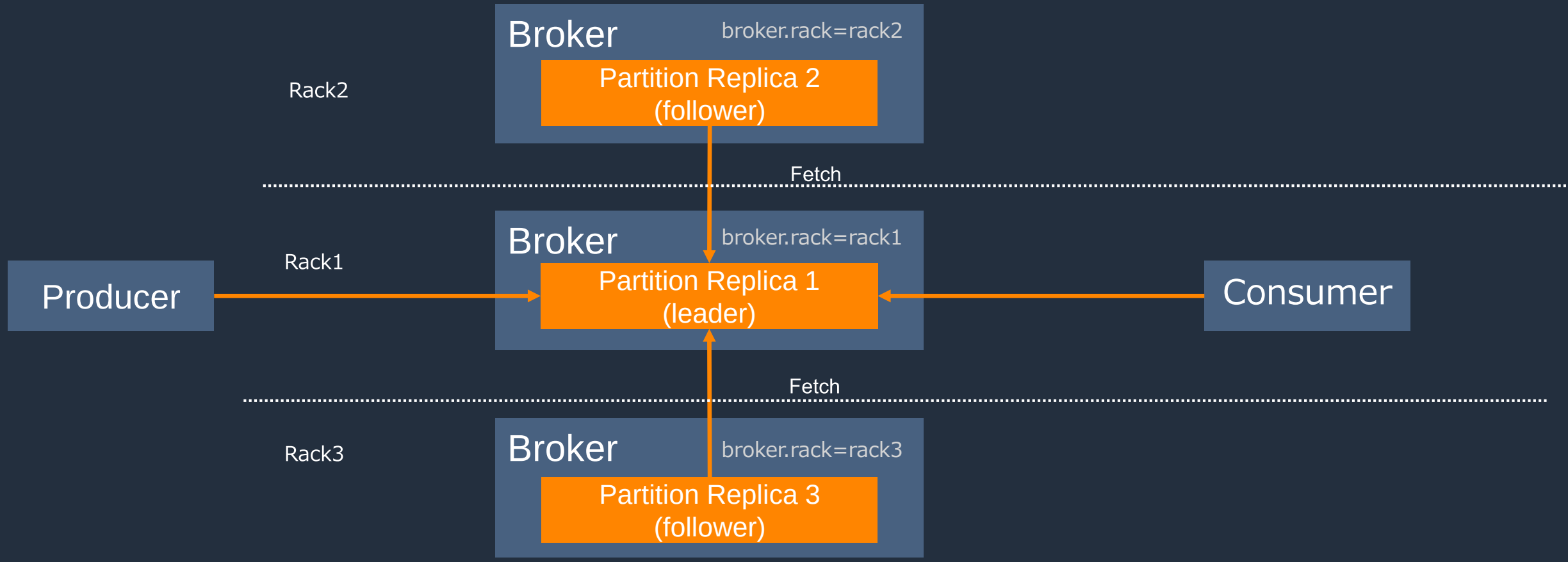
レプリカとコンシューマー

- デフォルトの挙動では、コンシューマーも Leader パーティションからデータを取得する



Rack Awareness によるパーティション配置の最適化

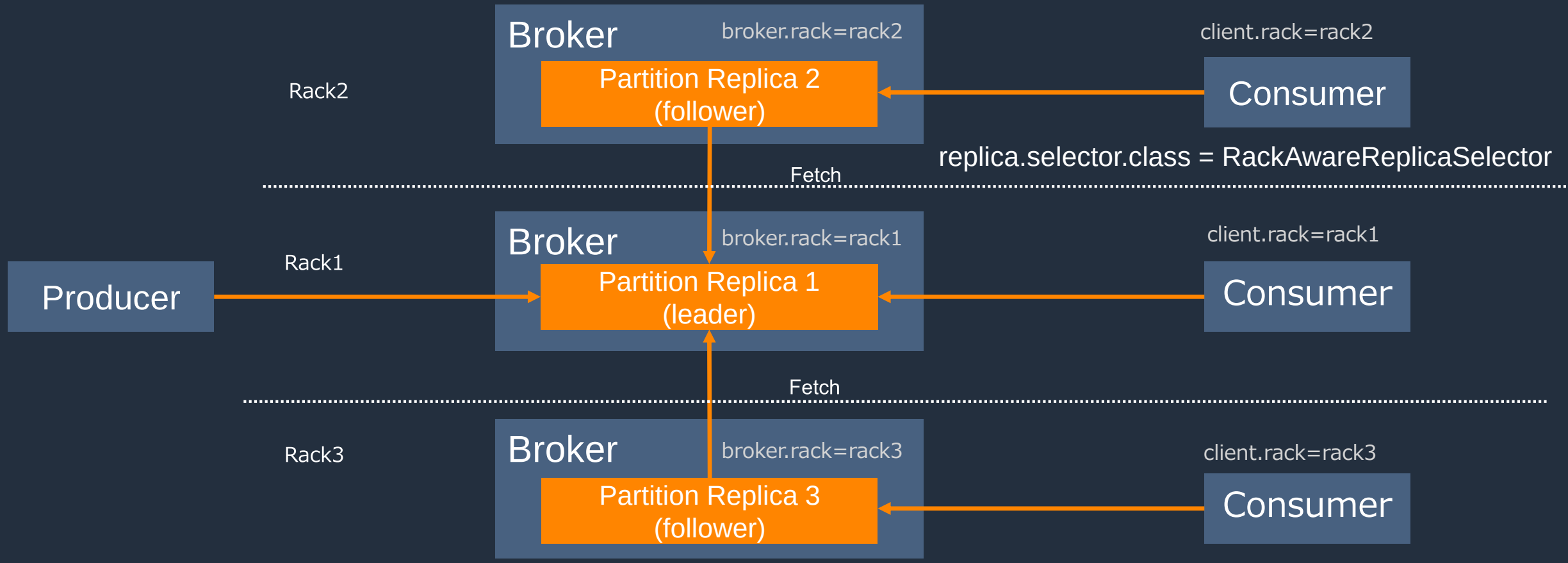
- Rack Awareness により、レプリカを任意に分散配置することが可能(バージョン 2.4.1 以降)
- broker.rack の設定により分散を制御する



https://aws.amazon.com/jp/blogs/big-data/reduce-network-traffic-costs-of-your-amazon-msk-consumers-with-rack-awareness/

Rack Awareness による通信の最適化

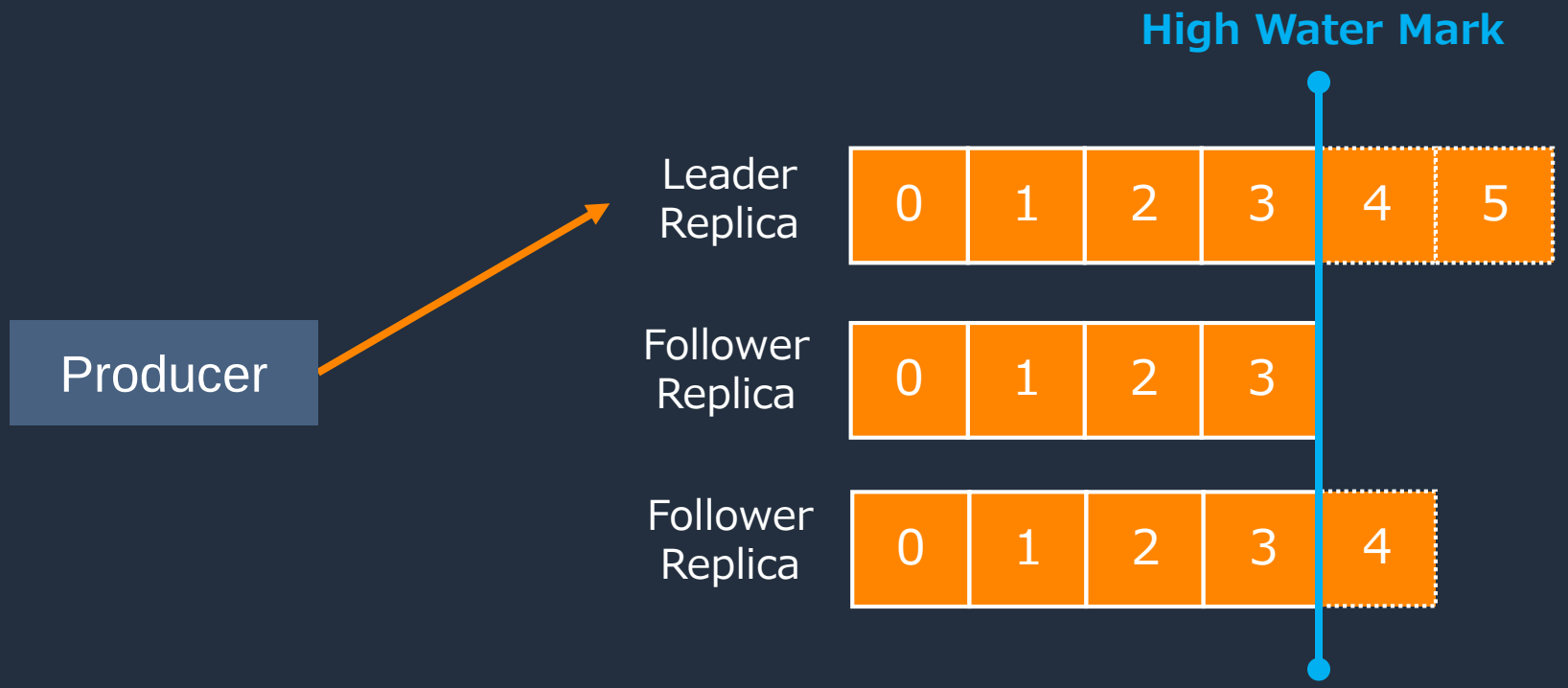
同一ラックのブローカーからデータを取得するようにコンシューマーを設定することで、
レイテンシの短縮を実現



https://aws.amazon.com/jp/blogs/big-data/reduce-network-traffic-costs-of-your-amazon-msk-consumers-with-rack-awareness/

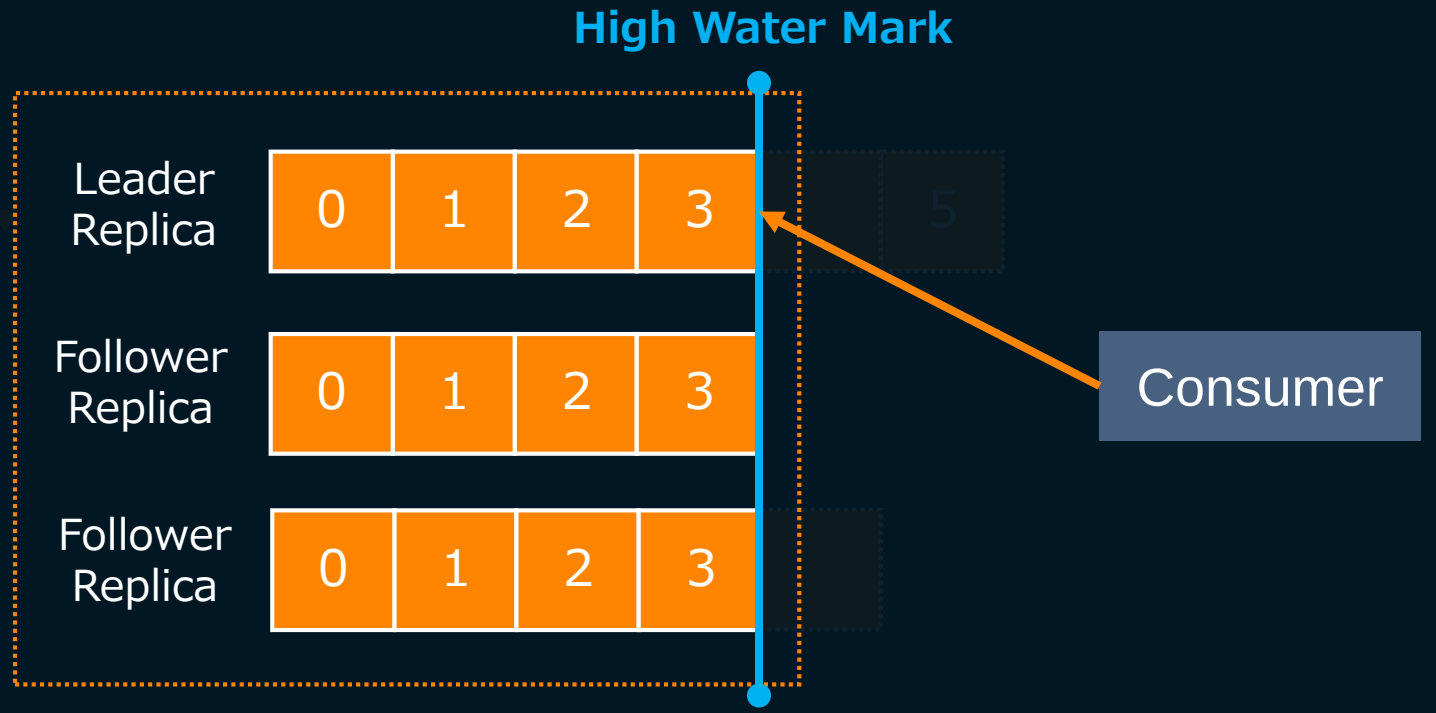
ハイウォーターマーク

- レプリカによる複製が完了済みのオフセット
- コンシューマーはハイウォーターマークのデータのみ取得可能



ハイウォーターマーク

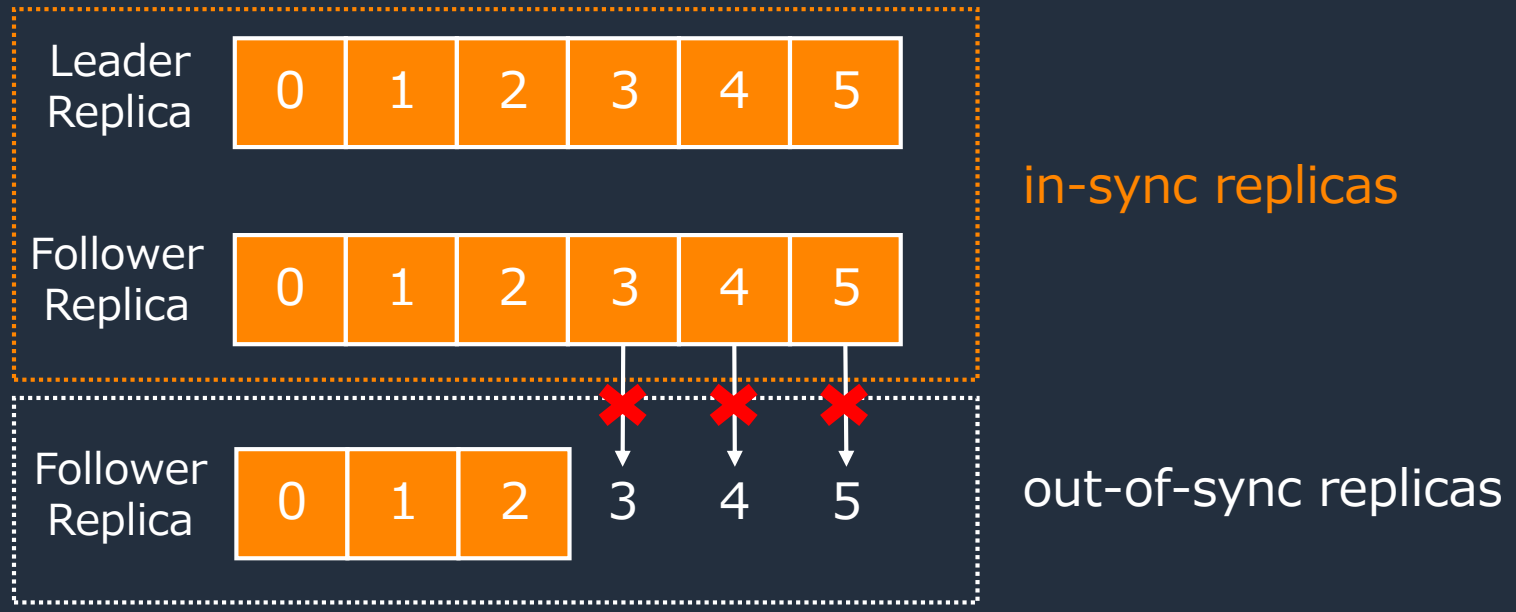
- レプリカによる複製が完了済みのオフセット
- コンシューマーはハイウォーターマークのデータのみ取得可能



コンシューマーは枠内のオフセットまでのメッセージを取得できる

レプリカの状態追跡

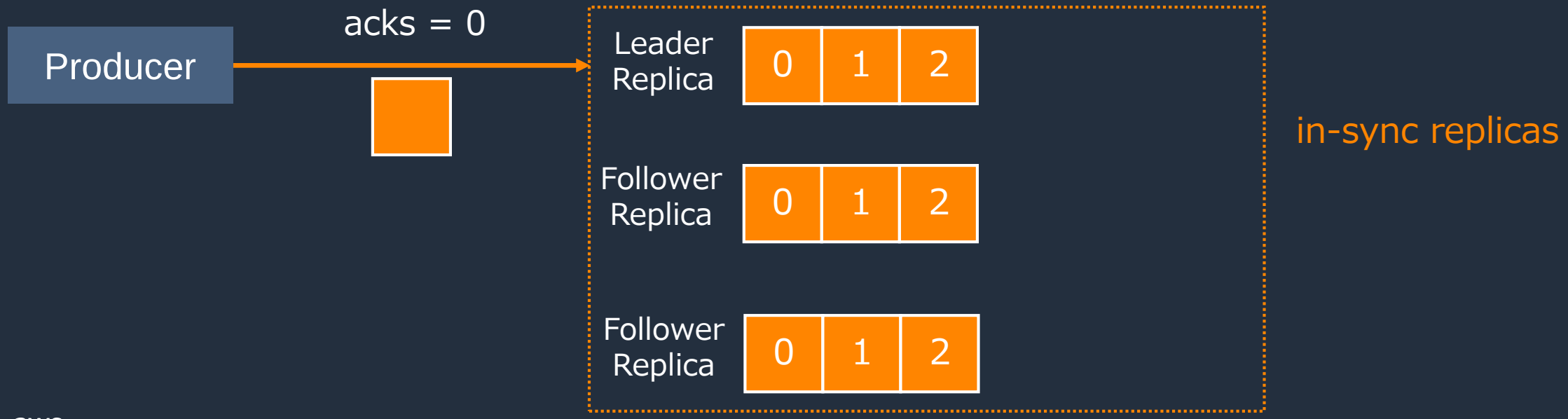
- Kafka はリーダーレプリカとフォロワーレプリカの状態を追跡する。同期が正常に行われているレプリカは in-sync replicas、ブローカーやネットワークの問題により、一定時以上で同期されていないレプリカは out-of-sync replicas と呼ばれる



最小同期レプリカ数と acks

Producer は acks と呼ばれるパラメーターによって、書き込みに成功したと判定するタイミングを制御可能

acks = 0: broker にメッセージが到達すれば、broker の返答が無くても成功とみなす



最小同期レプリカ数と acks

Producer は acks と呼ばれるパラメーターによって、書き込みに成功したと判定するタイミングを制御可能

acks = 0: broker にメッセージが到達すれば、broker の返答が無くても成功とみなす

acks = 1: leader パーティションから応答が有った時点で成功とみなす

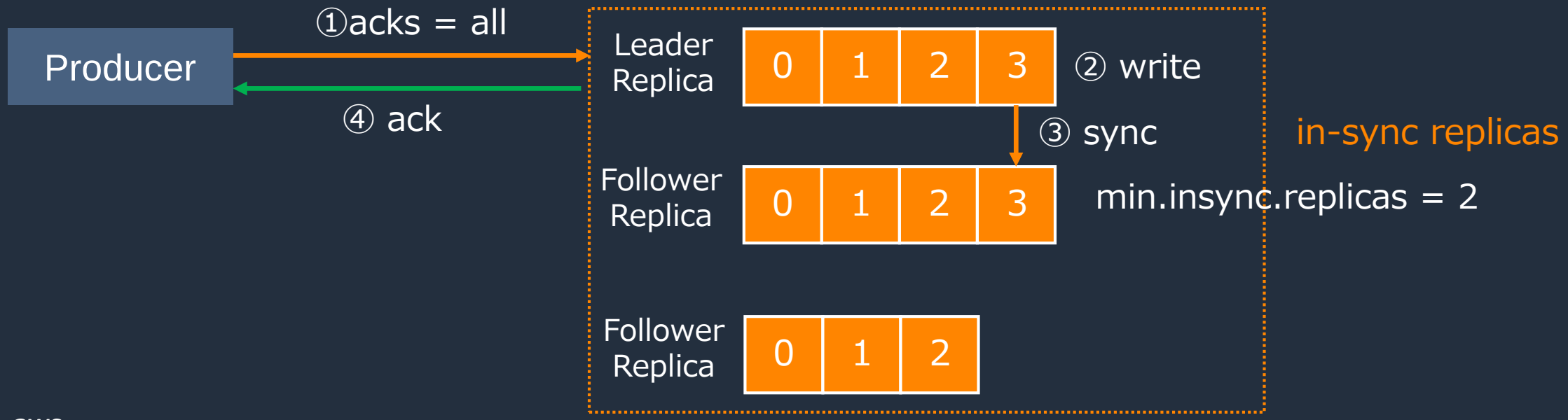
最小同期レプリカ数と acks

Producer は acks と呼ばれるパラメーターによって、書き込みに成功したと判定するタイミングを制御可能

acks = 0: broker にメッセージが到達すれば、broker の返答が無くても成功とみなす

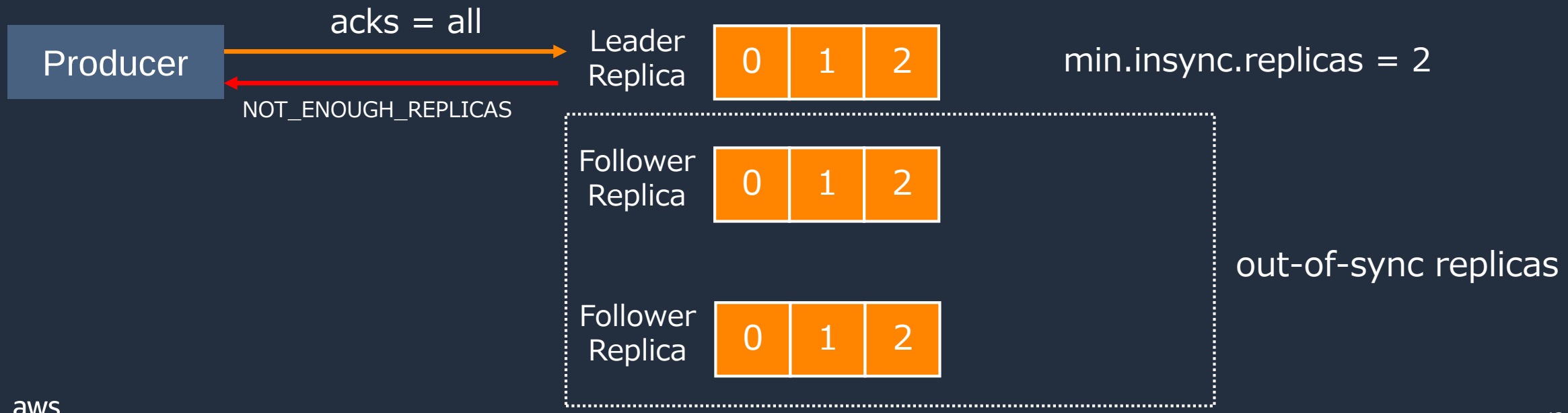
acks = 1: leader パーティションから応答が有った時点で成功とみなす

acks = all: トピックの最小レプリカ数(min.insync.replicas) に書き込まれて初めて成功とみなす



In-sync-replicas が最小同期レプリカ数を下回ると

NOT_ENOUGH_REPLICAS エラーにより書き込みが拒否される



Kafka におけるメッセージの送達保証

プロデューサー、コンシューマーの設定の組み合わせによって、
適用される送達保証のレベルが変わる

At most once
メッセージは再送されないため、失われる可能性がある

At least once
再送処理によってメッセージは失われることは回避できるが、
同一メッセージが重複して配信されることを考慮する必要が有る

Exactly once
各メッセージは一度だけ配信される

https://kafka.apache.org/documentation/#semantics

exactly once 実現上の考慮事項

プロデューサー

- 単一プロデューサーにおける冪等を有効化 (enable.idempotence = true)
- プロデューサーにおけるトランザクションを有効化 (transaction.id)
- 最小同期レプリカへ書き込みが成功するまで待機 (acks=all)
- リトライ回数はデフォルトのままとする (retries=2147483647)
- データの一意性を判断するためのユニークキーの挿入 (要アプリケーション実装)
- 処理取り消しのための補償トランザクションの実装 (要アプリケーション実装)

コンシューマー

- 自動コミットの無効化、手動によるコミット(enable.auto.commit = false)
- 非トランザクションメッセージ、コミット済みメッセージのみを受信 (isolation.level=read_commited)
- ユニークキーを元にした重複メッセージのチェック(要アプリケーション実装)

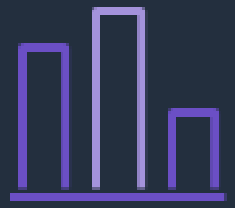
https://kafka.apache.org/documentation/#semantics

Amazon MSK 概要

Apache Kafka 運用における課題

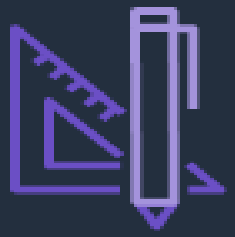
セットアップ

スケーリング



高可用性の達成

統合のための開発



エラー対応、運用管理

メンテナンスコスト

Amazon Managed Streaming for Apache Kafka

Apache Kafka および
Kafka Connect の
フルマネージドサービス

Amazon MSK の特徴

セキュリティ

VPC によるネットワークの分離・保護、保管時・転送中のデータ暗号化、
IAM によるアクセス制御など、複数のセキュリティオプションを利用可能

高可用性

複数のアベイラビリティゾーンに分散してレプリカを配置可能。
ノード障害の監視、自動復旧機能を提供

プライベート接続

データを AWS ネットワーク内にとどめておくことができる。
パブリックインターネットには公開されないよう構成可能

完全な互換性

ソースコードを変更せずに、AWS で既存の Apache Kafka アプリケーションを実行可能。
Cruise Control や Kafka Connect などの周辺ツールにも対応

Amazon MSK の特徴

AWS サービスとの統合

データ ソースとしての AWS IoT、データ コンシューマーとしての AWS Lambda、AWS Glue Schema Registry によるスキーマ管理、Amazon Kinesis Data Analytics によるストリーム処理

スケーラビリティ

ブローカーの追加、ブローカーのサイズの変更、ストレージの追加

モニタリング・オブザーバビリティ

Amazon CloudWatch を介したログとメトリクスの監視、Open Monitoring for Prometheus による JMX メトリクスの抽出などをサポート

ローリングバージョンアップグレード

クラスターのダウンタイムなしで Kafka バージョンをアップグレードする

2 つのクラスタータイプを提供

Amazon MSK Provisioned

- クラスターのプロビジョニング、設定変更、メンテナンスを自動化
- オープンソースの Apache Kafka との完全な互換性
- 細かなパラメーターチューニング
- 高いスケーラビリティ
- 割り当てたリソースに対する時間単位の課金

Amazon MSK Serverless

- 個別のブローカーノードのモニタリング、スケーリング、パーティションの再割り当てといった作業は不要
- インフラストラクチャのアップデートなどのメンテナンス作業が不要
- オートスケールにより、事前サイジングやピークに合わせた余剰リソースの確保が不要
- スループットに応じた課金

2 つのクラスタータイプを提供

本 Black Belt の対象範囲

Amazon MSK Provisioned

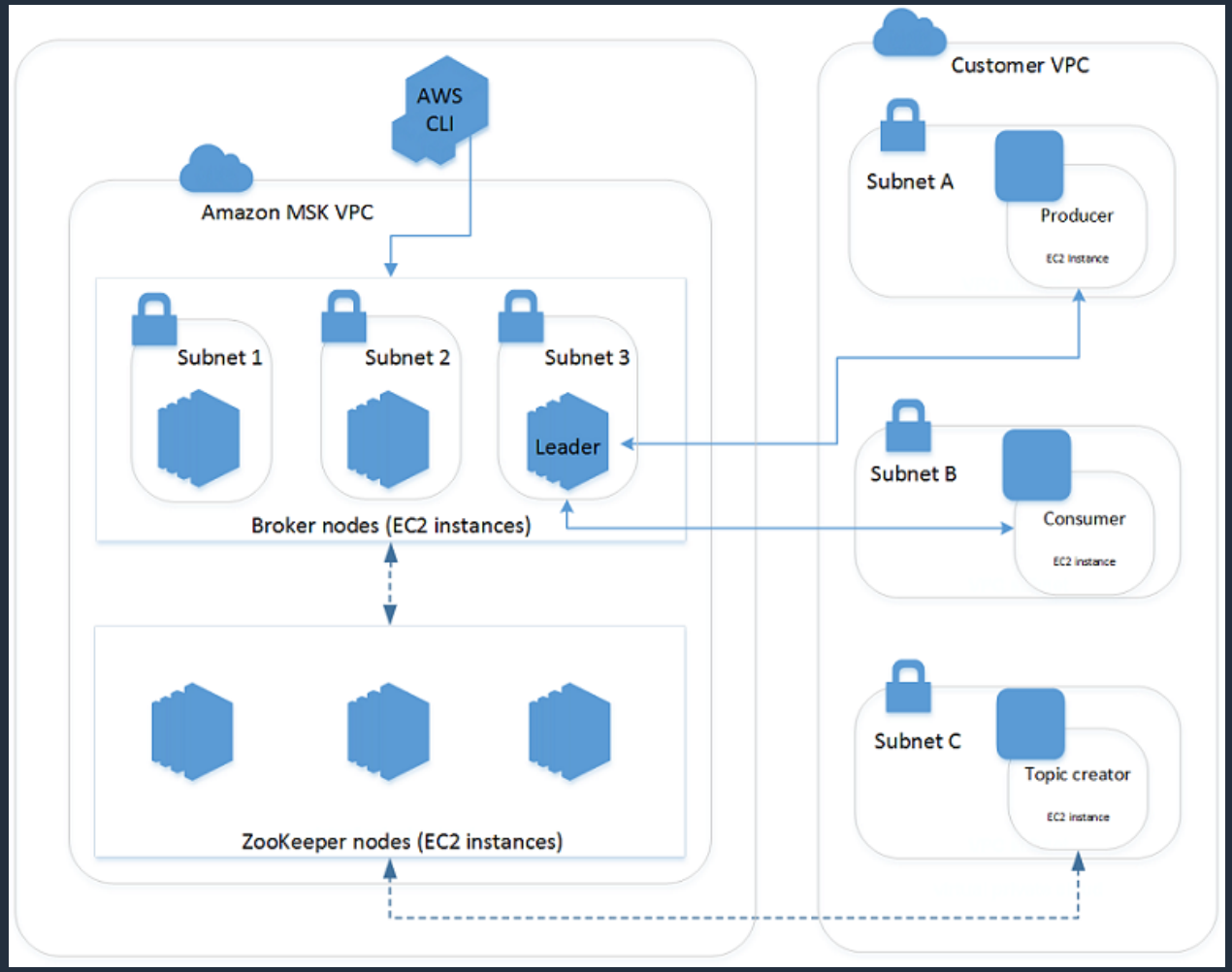
- クラスターのプロビジョニング、設定変更、メンテナンスを自動化
- オープンソースの Apache Kafka との完全な互換性
- 細かなパラメーターチューニング
- 高いスケーラビリティ
- 割り当てたリソースに対する時間単位の課金

Amazon MSK Serverless

- 個別のブローカーノードのモニタリング、スケーリング、パーティションの再割り当てといった作業は不要
- インフラストラクチャのアップデートなどのメンテナンス作業が不要
- オートスケールにより、事前サイジングやピークに合わせた余剰リソースの確保が不要
- スループットに応じた課金

Amazon MSK Provisioned Cluster のアーキテクチャー

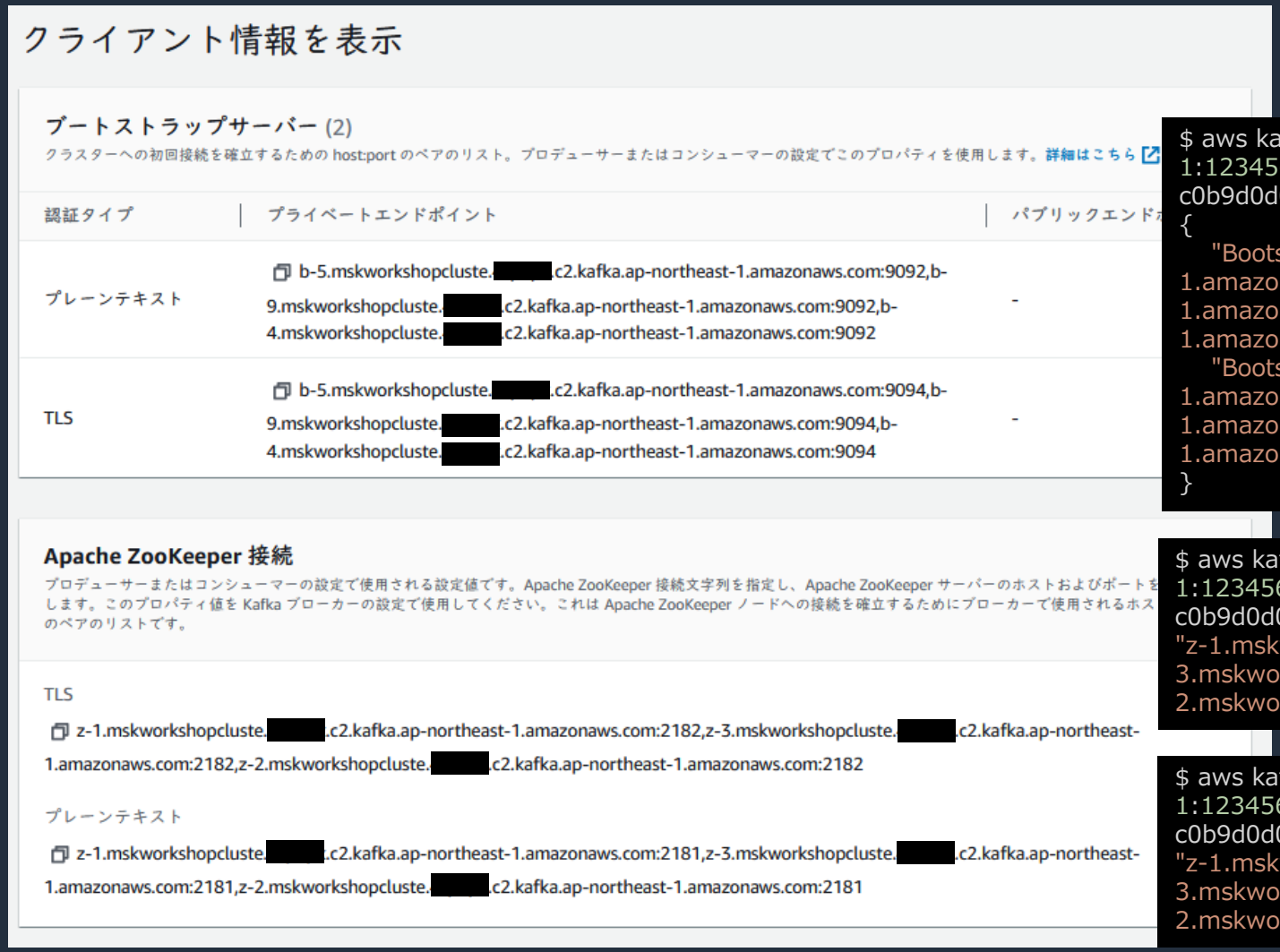
- ユーザーはブローカーノードのスペック、台数を指定してクラスターを作成する
- ZooKeeper はサービスにより自動的にプロビジョン、管理される
- クラスターは Amazon MSK が所有する VPC に作成される。
- ユーザー VPC にはブローカーおよび ZooKeeper にアクセスするための ENI が作成される
- KRaft には未対応



Provisioned クラスターの作成

- コンソール、API から作成可能。コンソール上の作業は数ステップで完了
- クラスターのサイズや VPC 、セキュリティ設定などを行う

接続情報の取得




```
$ aws kafka get-bootstrap-brokers --cluster-arn arn:aws:kafka:ap-northeast-1:123456789012:cluster/MSKWorkshopCluster/11ae4b6e-9b2d-411c-8871-c0b9d0d070c0-2
{
   "BootstrapBrokerString": "b-7.mskworkshopcluste.******.c2.kafka.ap-northeast-1.amazonaws.com:9092,b-3.mskworkshopcluste.******.c2.kafka.ap-northeast-1.amazonaws.com:9092,b-4.mskworkshopcluste.******.c2.kafka.ap-northeast-1.amazonaws.com:9092",
   "BootstrapBrokerStringTls": "b-7.mskworkshopcluste.******.c2.kafka.ap-northeast-1.amazonaws.com:9094,b-3.mskworkshopcluste.******.c2.kafka.ap-northeast-1.amazonaws.com:9094,b-4.mskworkshopcluste.******.c2.kafka.ap-northeast-1.amazonaws.com:9094"
}
```

```
$ aws kafka describe-cluster-v2 --cluster-arn arn:aws:kafka:ap-northeast-1:123456789012:cluster/MSKWorkshopCluster/11ae4b6e-9b2d-411c-8871-c0b9d0d070c0-2 --query ClusterInfo.Provisioned.ZookeeperConnectString
"z-1.mskworkshopcluste.******.c2.kafka.ap-northeast-1.amazonaws.com:2181,z-3.mskworkshopcluste.******.c2.kafka.ap-northeast-1.amazonaws.com:2181,z-2.mskworkshopcluste.******.c2.kafka.ap-northeast-1.amazonaws.com:2181"
```

```
$ aws kafka describe-cluster-v2 --cluster-arn arn:aws:kafka:ap-northeast-1:123456789012:cluster/MSKWorkshopCluster/11ae4b6e-9b2d-411c-8871-c0b9d0d070c0-2 --query ClusterInfo.Provisioned.ZookeeperConnectStringTls
"z-1.mskworkshopcluste.******.c2.kafka.ap-northeast-1.amazonaws.com:2182,z-3.mskworkshopcluste.******.c2.kafka.ap-northeast-1.amazonaws.com:2182,z-2.mskworkshopcluste.******.c2.kafka.ap-northeast-1.amazonaws.com:2182"
```

トピックの作成

```
$ export BS=$( aws kafka get-bootstrap-brokers --cluster-arn arn:aws:kafka:ap-northeast-1:123456789012:cluster/MSKWorkshopCluster/11ae4b6e-9b2d-411c-8871-c0b9d0d070c0-2 --query BootstrapBrokerString --output text)

$ ./kafka-topics.sh --bootstrap-server $BS --create --topic ExampleTopic10 --partitions 10 --replication-factor 3
Created topic "ExampleTopic10".

$ ./kafka-topics.sh --bootstrap-server $BS --describe --topic ExampleTopic10
Topic: test10   PartitionCount: 10      ReplicationFactor: 3    Configs:
min.insync.replicas=2,message.format.version=2.7-IV2,unclean.leader.election.enable=true
        Topic: test10   Partition: 0    Leader: 1       Replicas: 1,3,2 Isr: 1,2,3
        Topic: test10   Partition: 1    Leader: 2       Replicas: 2,1,3 Isr: 1,2,3
        Topic: test10   Partition: 2    Leader: 3       Replicas: 3,2,1 Isr: 1,2,3
        Topic: test10   Partition: 3    Leader: 1       Replicas: 1,2,3 Isr: 1,2,3
        Topic: test10   Partition: 4    Leader: 2       Replicas: 2,3,1 Isr: 1,2,3
        Topic: test10   Partition: 5    Leader: 3       Replicas: 3,1,2 Isr: 1,2,3
        Topic: test10   Partition: 6    Leader: 1       Replicas: 1,3,2 Isr: 1,2,3
        Topic: test10   Partition: 7    Leader: 2       Replicas: 2,1,3 Isr: 1,2,3
        Topic: test10   Partition: 8    Leader: 3       Replicas: 3,2,1 Isr: 1,2,3
        Topic: test10   Partition: 9    Leader: 1       Replicas: 1,2,3 Isr: 1,2,3
```


https://catalog.workshops.aws/msk-labs/en-US/commontasks/kafkacrud

メッセージ送受信

```
# Amazon MSK の Get Bootstrap Brokers API で クラスターの接続先ブローカー (BootstrapBrokerString) を取得
$ aws kafka get-bootstrap-brokers --cluster-arn "ClusterArn”
{
    "BootstrapBrokerStringTls": "b-1.awskafkatutorialc.6xjbqy.c4.kafka.ap-northeast-1.amazonaws.com:9094,b-2.awskafkatutorialc.6xjbqy.c4.kafka.ap-northeast-1.amazonaws.com:9094,b-3.awskafkatutorialc.6xjbqy.c4.kafka.ap-northeast-1.amazonaws.com:9094"
}

# Kafka のプロデューサーユーティリティでブローカーに接続し、イベントを送信
$ bin/kafka-console-producer.sh --broker-list BootstrapBrokerString ¥
            --producer.config client.properties --topic AWSKafkaTutorialTopic
> Hello Kafka!
> Hello MSK!
```

```
# Kafka のコンシューマーユーティリティでブローカーに接続し、イベントを受信
$ bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString ¥
            --consumer.config client.properties --topic AWSKafkaTutorialTopic --from-beginning
Hello Kafka!
Hello MSK!
```

最も古いオフセットから
イベントを順番に取得

https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/produce-consume.html

クラスターの設定管理

- カスタム設定を作成し、MSK クラスターの新規作成時、更新時に指定することが可能

Kafkaの設定ファイルを作成 ⇨ MSKのConfigurationを作成 ⇨ MSKのクラスターを作成・更新

config-file

```
auto.create.topics.enable = true

zookeeper.connection.timeout.ms = 1000

log.roll.ms = 604800000
```

```
$ aws kafka create-configuration ¥
--name "CustomConfiguration" ¥
--description ”MSK BlackBelt." ¥
--kafka-versions "1.1.1" ¥
--server-properties file://config-file-path
{
  "Arn": "arn:aws:kafka:XXX",
  …,
  "LatestRevision": {
    …,
    "Description": "MSK BlackBelt.",
    "Revision": 1
  },
  "Name": "CustomConfiguration"
}
```

configuration-info.json

```
{
  "Arn": "arn:aws:kafka:XXX",
  "Revision": 1
}
```

```
$ aws kafka update-cluster-configuration ¥
--cluster-arn "arn:aws:kafka:YYY" ¥
--configuration-info file://configuration-info.json ¥
--current-version ”K21V3IB1VIZYYH”
{
  "ClusterArn": "arn:aws:kafka:YYY",
  "ClusterOperationArn": "arn:aws:kafka:ZZZ”
}
```

https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/msk-configuration.html

Amazon MSK の特徴

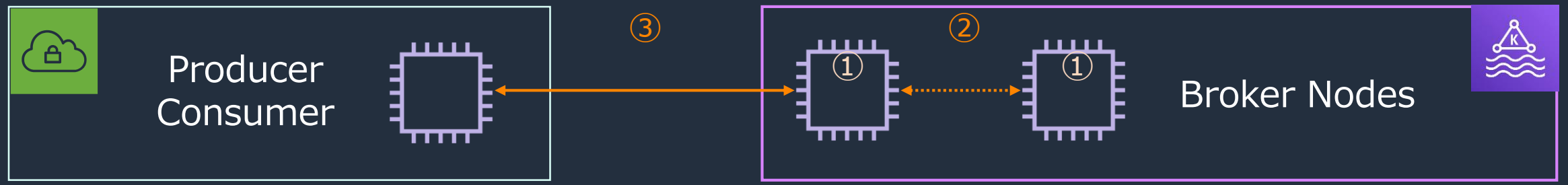
Amazon MSK の特徴 > セキュリティ

セキュリティ

VPC によるネットワークの分離・保護、保管時・転送中のデータ暗号化、
IAM によるアクセス制御など、複数のセキュリティオプションを利用可能

保管時のデータ暗号化

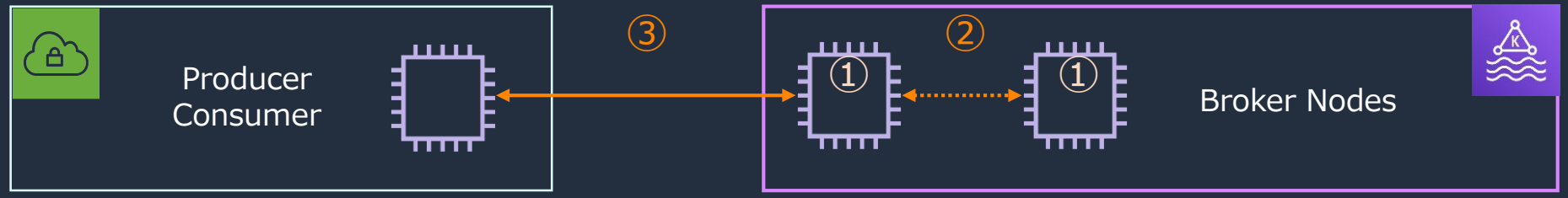
- MSK では、サーバー側の暗号化 ① がデフォルトで有効
- クラスター作成時に指定した Customer Managed Key (CMK) を暗号化に使用する。特に指定が無い場合は、MSK が作成する AWS Managed key が使用される。



https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/msk-encryption.html

通信の暗号化

- ブローカー同士の通信②はデフォルトで暗号化されている。無効化も可能
- クライアントとブローカー間の通信③は以下の3つから選択が可能
  1. TLS 1.2 で暗号化された通信のみを許可 (デフォルト)
  2. プレーンテキストとTLSで暗号化された通信の両方を許可
  3. プレーンテキストの通信のみを許可
- 有効時は追加の CPU オーバーヘッド、数ミリ秒のレイテンシ増加あり
- AWS Certificate Manager より発行された証明書を使用。有効期限は 13 ヵ月で、サービス側のメンテナンス処理により自動更新される



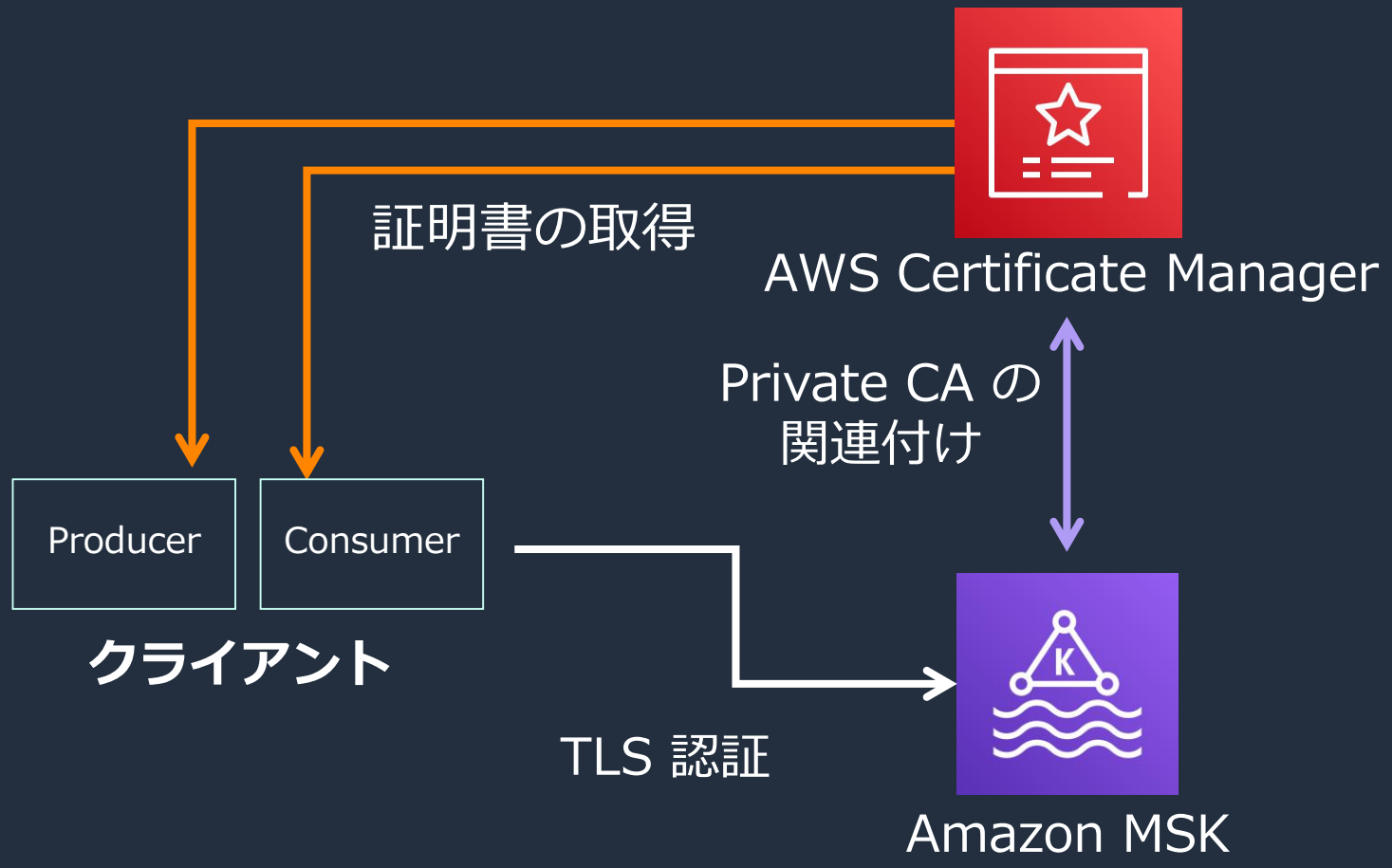
https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/msk-encryption.html

認証

複数の認証方式を選択可能。利用可能な方式は以下の通り

1. 認証されていないアクセス
2. TLS クライアント認証
3. IAM 認証
4. SALS/SCRAM 認証

AWS Certificate Manager を使用した TLS 認証

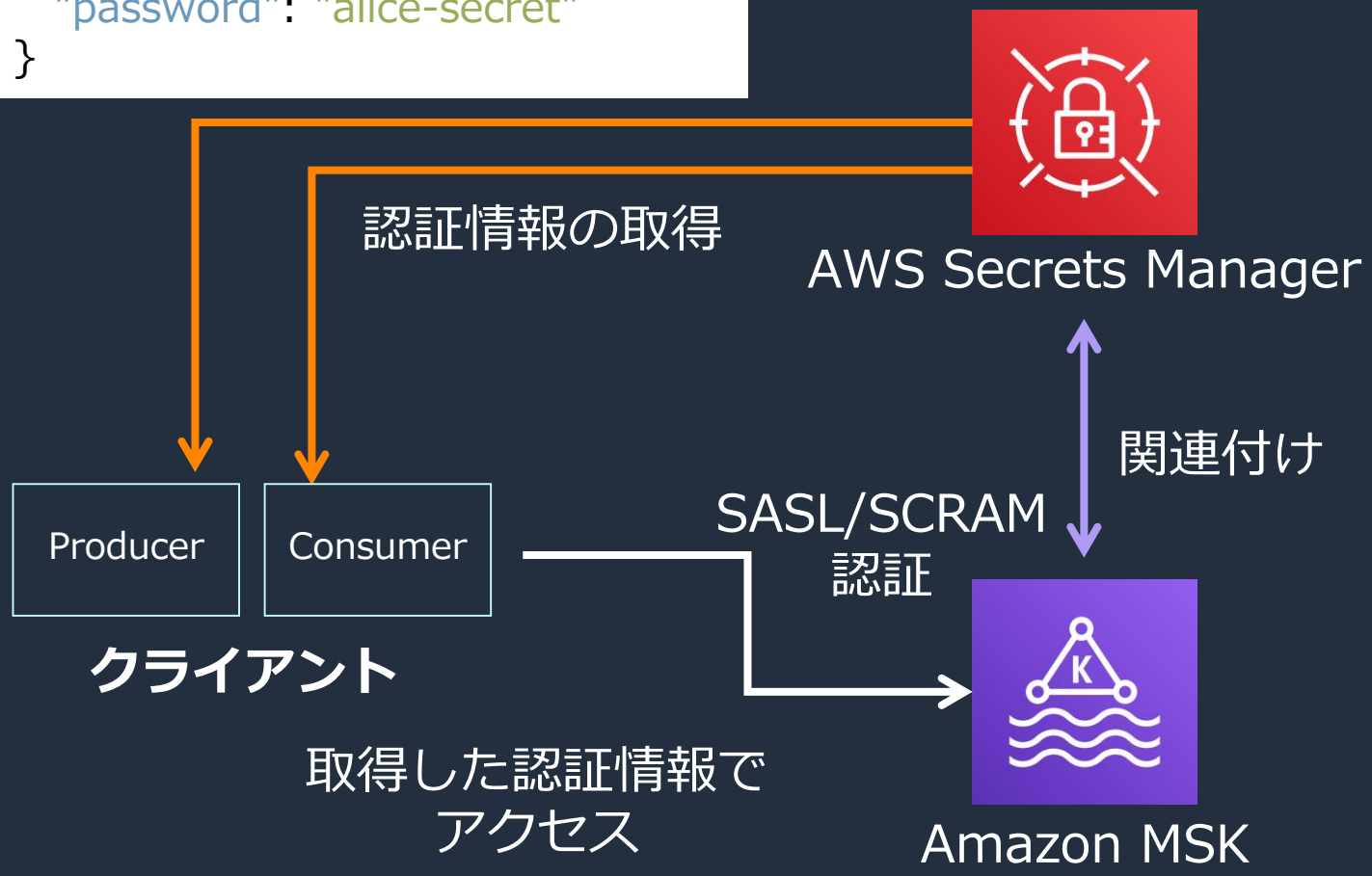


- Private CA から取得した証明書をクライアントマシンにセットすることで、TLS クライアント認証を実現
- 認可については、別途 Apache Kafka ACLs で設定を行う
- MSK クラスターごとに独立した Private CA を用意することを推奨

https://docs.aws.amazon.com/msk/latest/developerguide/msk-authentication.html

AWS Secrets Manager を利用した SASL/SCRAM認証

```
{
  "username": "alice",
  "password": "alice-secret"
}
```



- SASL = **S**imple **A**uthentication and **S**ecurity **L**ayer
- SCRAM = **S**alted **C**hallenge **R**esponse **A**uthentication Mechanism

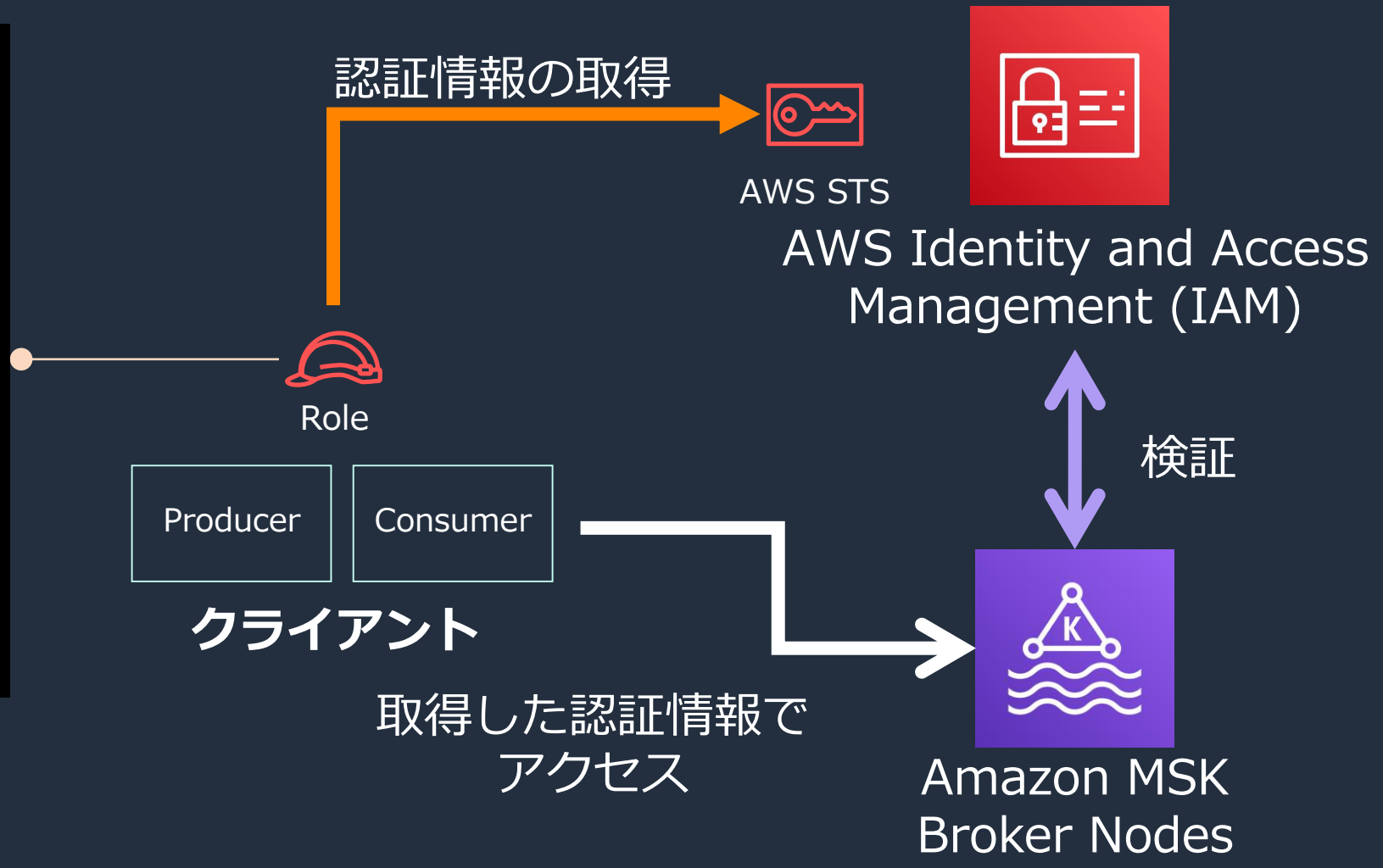
- ユーザー名とパスワードによる認証
- AWS Secrets Manager に認証情報を保管し、クラスターと関連付けることで、複数クラスターで同じ認証情報を共用可能
- クライアントのプロパティにユーザー名とパスワードを含めることで接続が可能になる
- 認可は Apache Kafka ACL で制御

https://docs.aws.amazon.com/msk/latest/developerguide/msk-password.html

IAM 認証

IAM ポリシーによるブローカーノードへの接続制御、認可制御が可能

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kafka-cluster:Connect",
        "kafka-cluster:CreateTopic",
        "kafka-cluster:DescribeTopic",
        "kafka-cluster:WriteData"
      ],
      "Resource": [
        "arn:aws:kafka:*:Account-ID:cluster/msk-tutorial/*",
        "arn:aws:kafka:*:Account-ID:topic/msk-tutorial/*"
      ]
    }
  ]
}
```



https://docs.aws.amazon.com/msk/latest/developerguide/iam-access-control.html

IAM 認証の特徴

- 認証用クライアントライブラリ として、AWS より提供される aws-msk-iam-auth が利用可能
- ZooKeeper へのアクセス制御には未対応
- CloudTrail による監査に対応。トピックレベルの操作(作成、削除など)が記録される。イベントの書き込みなどデータレベルの操作は記録されない。
- Zookeeper ノードへのアクセス制限は対象外
- インスタンスタイプに応じた TCP 新規接続数のレート制限あり
  - 20 TCP connections per broker per second (t3.small 以外)
  - 4 TCP connections per broker per second (t3.small)

https://github.com/aws/aws-msk-iam-auth
https://docs.aws.amazon.com/msk/latest/developerguide/limits.html

Apache Kafka ACL

- ACL をセットすることで TLS、SCRAM 認証時のユーザーごとの細かなアクセス制御が可能。ACL はトピック単位の設定となる。
- MSK ではデフォルトの設定が “allow.everyone.if.no.acl.found = true” のため、ACL がセットされていない場合は、フルオープンの状態になる。同設定は変更可能

```
[ec2-user@ip-10-0-1-194 kafka]$ bin/kafka-acls.sh --authorizer-properties zookeeper.connect=$zoo --add --allow-principal User:dummy
--operation Read --group=* --topic testaclfail
Adding ACLs for resource `ResourcePattern(resourceType=TOPIC, name=testaclfail, patternType=LITERAL)`:
        (principal=User:dummy, host=*, operation=READ, permissionType=ALLOW)

Adding ACLs for resource `ResourcePattern(resourceType=GROUP, name=*, patternType=LITERAL)`:
        (principal=User:dummy, host=*, operation=READ, permissionType=ALLOW)

Current ACLs for resource `Topic:LITERAL:testaclfail`:
        User:dummy has Allow permission for operations: Read from hosts: *

Current ACLs for resource `Group:LITERAL:*`:
        User:dummy has Allow permission for operations: Read from hosts: *
        User:CN=ip-10-0-1-194.ec2.internal has Allow permission for operations: Read from hosts: *
        User: has Allow permission for operations: Read from hosts: *

[ec2-user@ip-10-0-1-194 kafka]$ bin/kafka-acls.sh --authorizer-properties zookeeper.connect=$zoo --add --allow-principal User:dummy
--operation Write --topic testaclfail
Adding ACLs for resource `ResourcePattern(resourceType=TOPIC, name=testaclfail, patternType=LITERAL)`:
        (principal=User:dummy, host=*, operation=WRITE, permissionType=ALLOW)

Current ACLs for resource `Topic:LITERAL:testaclfail`:
        User:dummy has Allow permission for operations: Read from hosts: *
        User:dummy has Allow permission for operations: Write from hosts: *
```


https://docs.aws.amazon.com/msk/latest/developerguide/msk-acls.html

Amazon MSK compliance

以下をはじめとしたコンプライアンス基準を満たしている

- System and Organization Controls (SOC) 1, 2, 3
- Payment Card Industry Data Security Standard (PCI DSS)
- ISO および CSA STAR
- FedRAMP Moderate / High
- HIPAA

最新情報は AWS クラウドセキュリティのページを参照

https://aws.amazon.com/compliance/services-in-scope/?nc1=h_ls

Amazon MSK の特徴 > 高可用性

セキュリティ

VPC によるネットワークの分離・保護、保管時・転送中のデータ暗号化、
IAM によるアクセス制御など、複数のセキュリティオプションを利用可能

高可用性

複数のアベイラビリティゾーンに分散してレプリカを配置可能。
ノード障害の監視、自動復旧機能を提供

Amazon MSK クラスターの耐障害性

Amazon MSKのクラスターは、マルチAZクラスターの以下の一般的な障害シナリオを検出して自動的に回復

- ブローカーへのネットワーク接続の損失
- ブローカーノードの障害

正常でない (到達不能な) ブローカーは新しいブローカーに置き換えられる

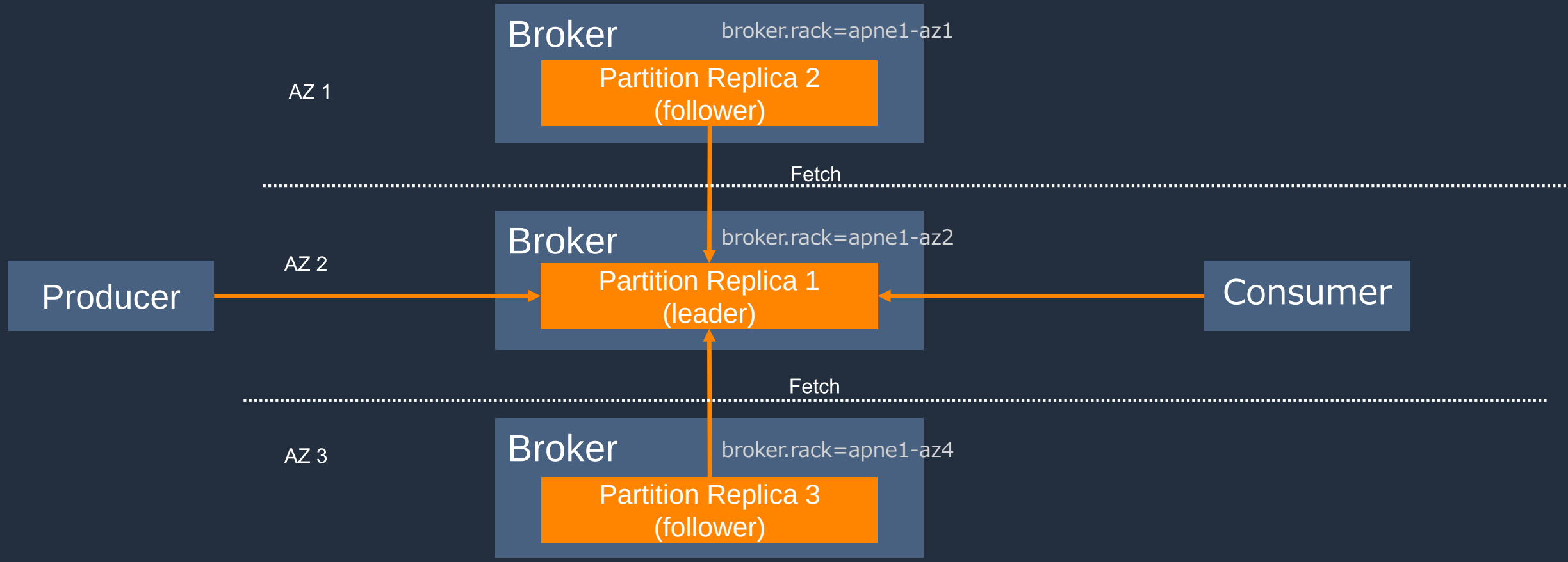
- 可能な限り古いブローカーのストレージを再利用して、Kafkaが複製する必要があるデータ量を削減する
- 可用性への影響は、MSK による問題検出から回復を完了するまでの所要時間に制限される
- 復旧後、プロデューサーアプリケーションとコンシューマーアプリケーションは、障害が発生する前に使用していたものと同じIPアドレスでブローカーとの通信を継続可能

3 つのアベイラビリティゾーンの利用、3 つ以上の**レプリカ**の利用を推奨

https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/what-is-msk.html

MSK におけるパーティション配置の最適化

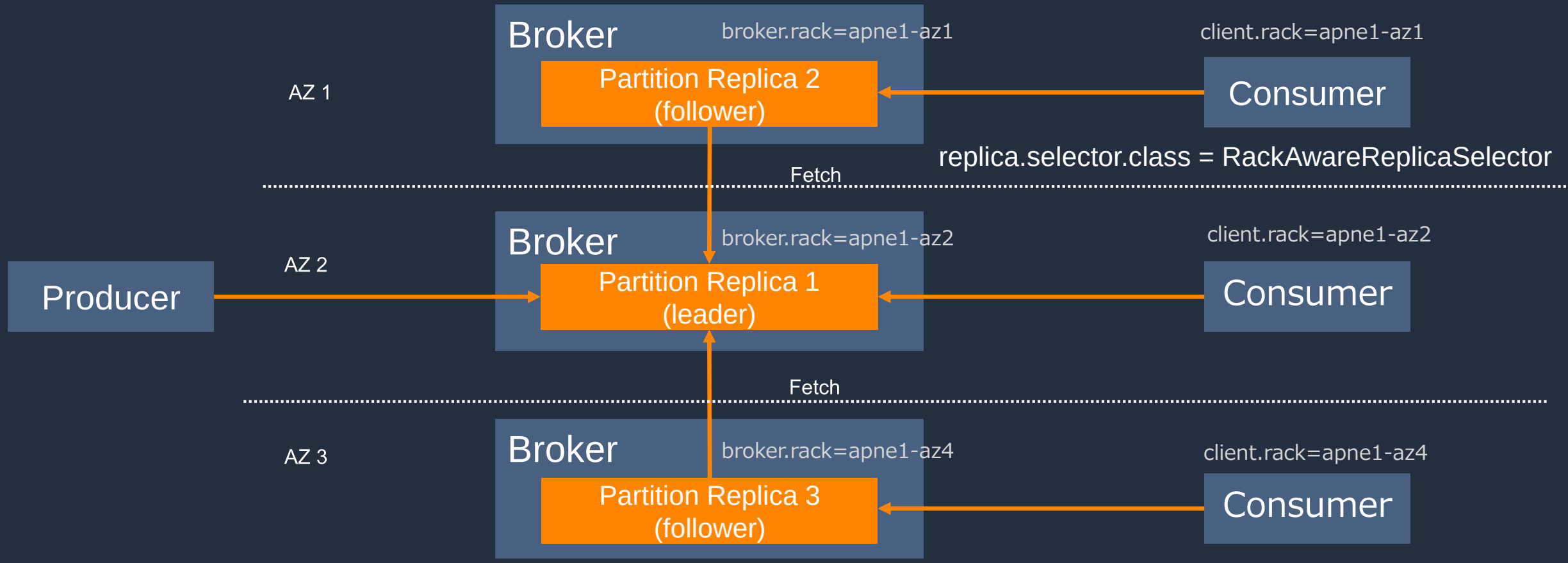
- Kafka が提供する Rack Awareness の仕組みを利用 (バージョン 2.4.1 以降)
- MSK がブローカーの broker.rack に アベイラビリティゾーンの ID をセットしている



https://aws.amazon.com/jp/blogs/big-data/reduce-network-traffic-costs-of-your-amazon-msk-consumers-with-rack-awareness/

Rack Awareness による通信の最適化

- コンシューマーの client.rack を指定することで、同一ゾーンのレプリカからデータを取得可能。ゾーン間転送料金の削減、レイテンシの短縮を実現



https://aws.amazon.com/jp/blogs/big-data/reduce-network-traffic-costs-of-your-amazon-msk-consumers-with-rack-awareness/

broker.rack の取得方法

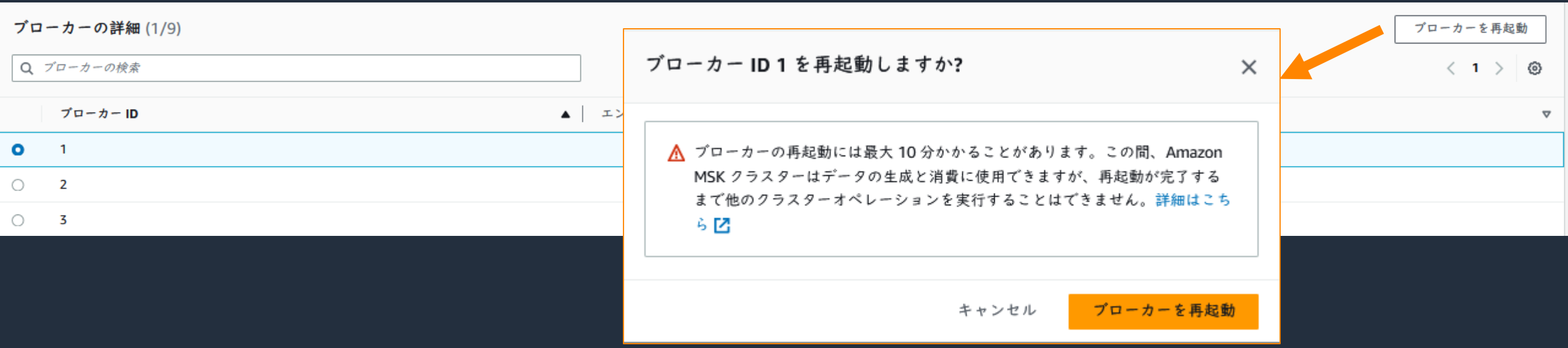
kafka-config.sh 等から取得可能。以下は 9 ノードクラスターにおける実行例

```
$ seq 1 9 | xargs -n1 -I% ./kafka-configs.sh --broker % --all --describe --bootstrap-server $BS | grep -e
"STATIC_BROKER_CONFIG:broker.id=" -e rack
  broker.rack=apne1-az4 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.rack=apne1-az4}
  broker.id=1 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.id=1, DEFAULT_CONFIG:broker.id=-1}
  broker.rack=apne1-az1 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.rack=apne1-az1}
  broker.id=2 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.id=2, DEFAULT_CONFIG:broker.id=-1}
  broker.rack=apne1-az2 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.rack=apne1-az2}
  broker.id=3 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.id=3, DEFAULT_CONFIG:broker.id=-1}
  broker.rack=apne1-az1 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.rack=apne1-az1}
  broker.id=4 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.id=4, DEFAULT_CONFIG:broker.id=-1}
  broker.rack=apne1-az4 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.rack=apne1-az4}
  broker.id=5 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.id=5, DEFAULT_CONFIG:broker.id=-1}
  broker.rack=apne1-az2 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.rack=apne1-az2}
  broker.id=6 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.id=6, DEFAULT_CONFIG:broker.id=-1}
  broker.rack=apne1-az4 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.rack=apne1-az4}
  broker.id=7 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.id=7, DEFAULT_CONFIG:broker.id=-1}
  broker.rack=apne1-az1 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.rack=apne1-az1}
  broker.id=8 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.id=8, DEFAULT_CONFIG:broker.id=-1}
  broker.rack=apne1-az2 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.rack=apne1-az2}
  broker.id=9 sensitive=false synonyms={STATIC_BROKER_CONFIG:broker.id=9, DEFAULT_CONFIG:broker.id=-1}
```


https://aws.amazon.com/jp/blogs/big-data/reduce-network-traffic-costs-of-your-amazon-msk-consumers-with-rack-awareness/

Amazon MSK に対する障害試験

- コンソールもしくは RebootBroker API を用いて任意のブローカーを再起動可能
- 再起動に伴う Kafka クライアントの挙動確認を通じて、障害時のシステムの回復性を確認
- クラスターの状態が ACTIVE の場合に実行でき、再起動中は REBOOT_IN_PROGRESS、再起動完了後は REBOOT_COMPLETE の状態になる
- ZooKeeper の再起動は不可



https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/msk-reboot-broker.html
https://docs.aws.amazon.com/msk/1.0/apireference/clusters-clusterarn-reboot-broker.html

可用性のベストプラクティス

3 AZ にクラスターを構築し、Replication Factor は 3 に、min.insync.replicas を 2 にセットするのが典型的な設定例

- 3 AZ にクラスターを構築する
- トピックの Replication Factor は 3 以上にセットする
- 最小同期レプリカ数 (min.insync.replicas) は (Replication Factor – 1) を最大値としてセットする
- Rack Awareness により AZ 間でレプリカが分散されるようにする。Consumer に client.rack をセットする

https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/bestpractices.html#ensure-high-availability

Amazon MSK の特徴

セキュリティ

VPC によるネットワークの分離・保護、保管時・転送中のデータ暗号化、
IAM によるアクセス制御など、複数のセキュリティオプションを利用可能

高可用性

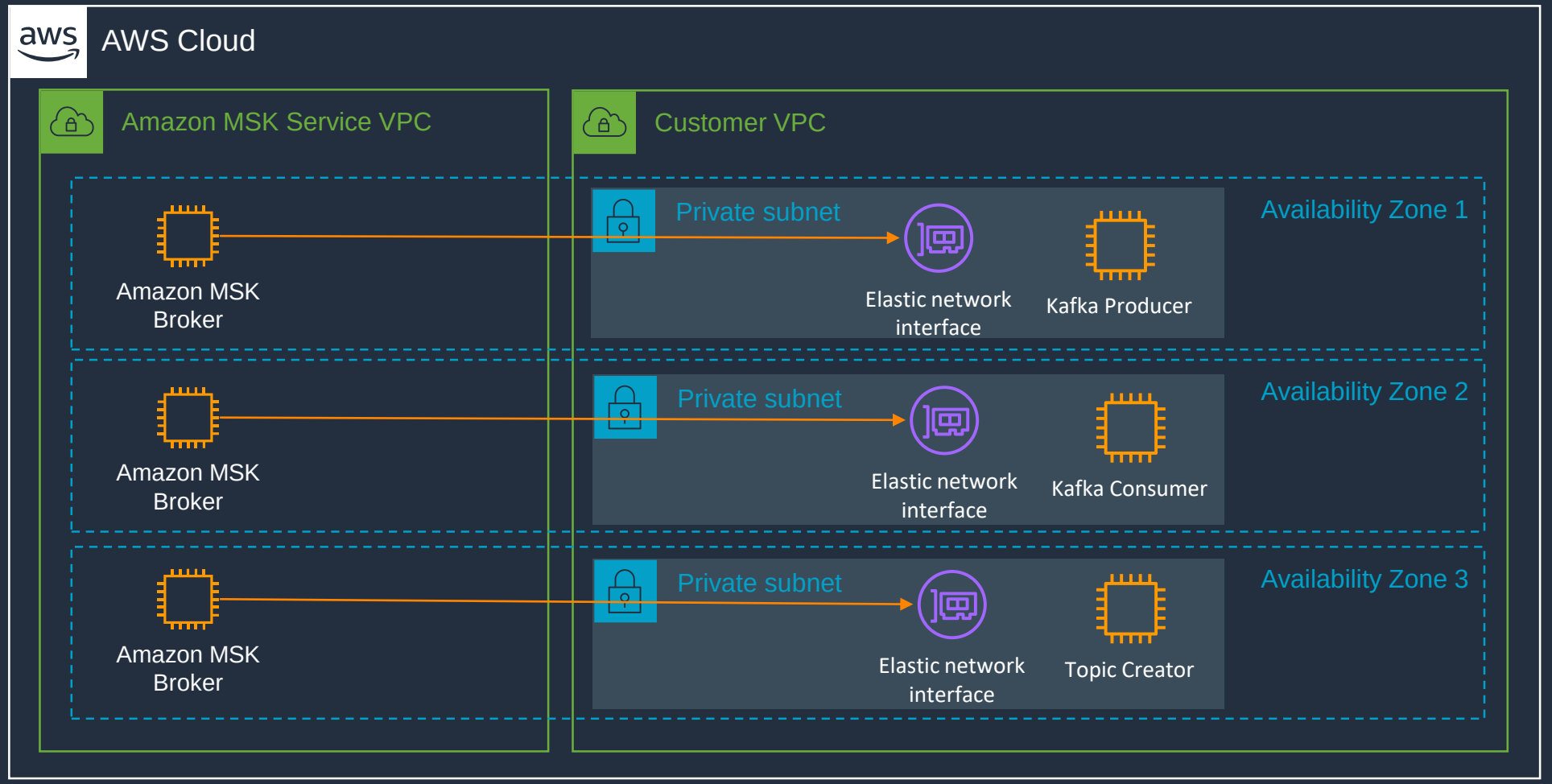
複数のアベイラビリティゾーンに分散してレプリカを配置可能。
ノード障害の監視、自動復旧機能を提供

プライベート接続

データを AWS ネットワーク内にとどめておくことができる。
パブリックインターネットには公開されないよう構成可能

MSK クラスターへのプライベート接続

デフォルトでは VPC および Direct Connect などの拠点からのみ接続可能。
セキュリティグループにも対応



パブリック接続の有効化

追加でパブリックエンドポイントを割り当てることでインターネット経由の接続が可能。
暗号化などクラスターのセキュリティ設定を有効化することが前提条件

Amazon MSK の特徴

セキュリティ

VPC によるネットワークの分離・保護、保管時・転送中のデータ暗号化、
IAM によるアクセス制御など、複数のセキュリティオプションを利用可能

高可用性

複数のアベイラビリティゾーンに分散してレプリカを配置可能。
ノード障害の監視、自動復旧機能を提供

プライベート接続

データを AWS ネットワーク内にとどめておくことができる。
パブリックインターネットには公開されないよう構成可能

完全な互換性

ソースコードを変更せずに、AWS で既存の Apache Kafka アプリケーションを実行可能。
Cruise Control や Kafka Connect などの周辺ツールにも対応

Amazon MSK における Kafka のエコシステムとの互換性

- **オープンソースの Apache Kafka に含まれるフレームワーク(使用可能)**
  - ✓ Kafka Mirror Maker
  - ✓ Kafka Connect
  - ✓ Kafka Streams

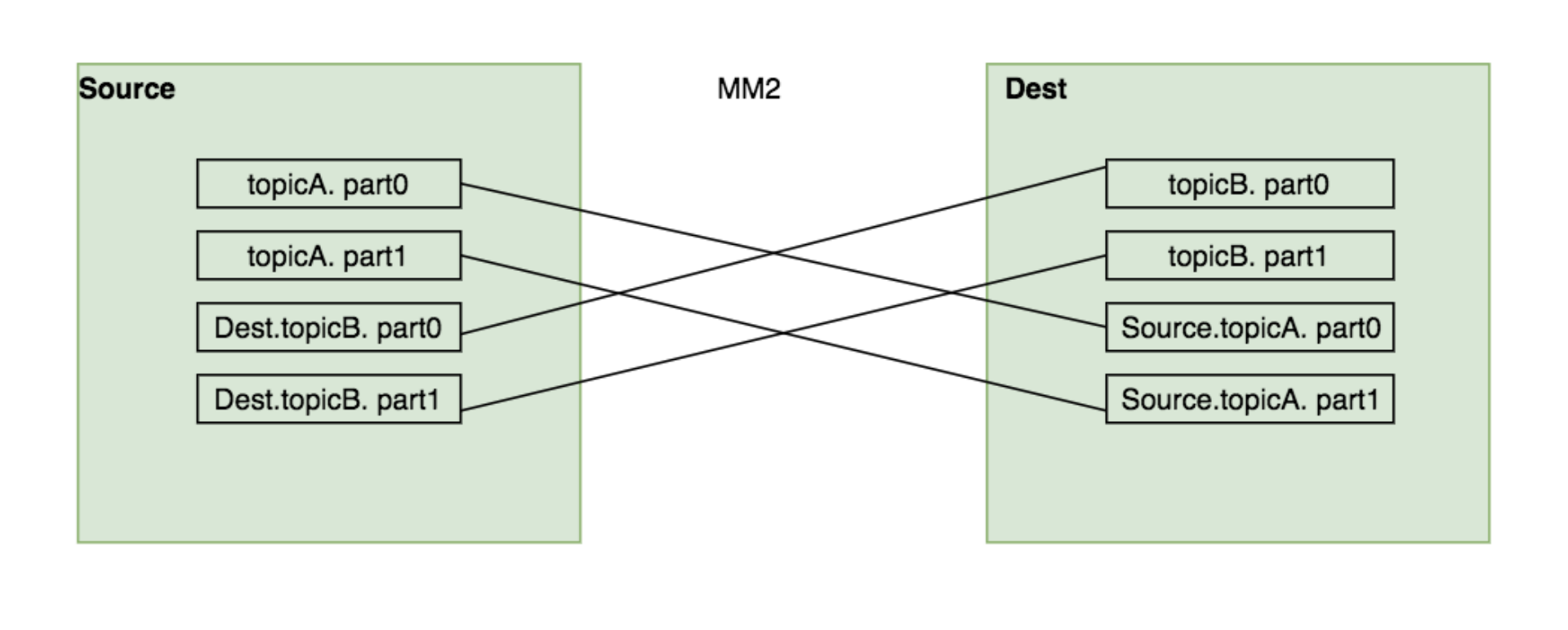
- **Apache Kafka の周辺ツール・フレームワーク(使用可能)**
  - ✓ AWS Glue Schema Registry or 3rd party schema registries
  - ✓ REST proxies
  - ✓ Cruise Control
  - ✓ その他のサードパーティーツール: Burrow, Kafdrop, CMAK, etc.

- **ブローカーに Jar ファイルをロードする必要があるツール(使用不可)**
  - Confluent Control Center
  - Auto Data Balancer (Confluent)
  - uReplicator (Uber)

Mirror Maker 2.0 (MM2)

Kafka クラスター間でトピックのレプリケーションを行うツール

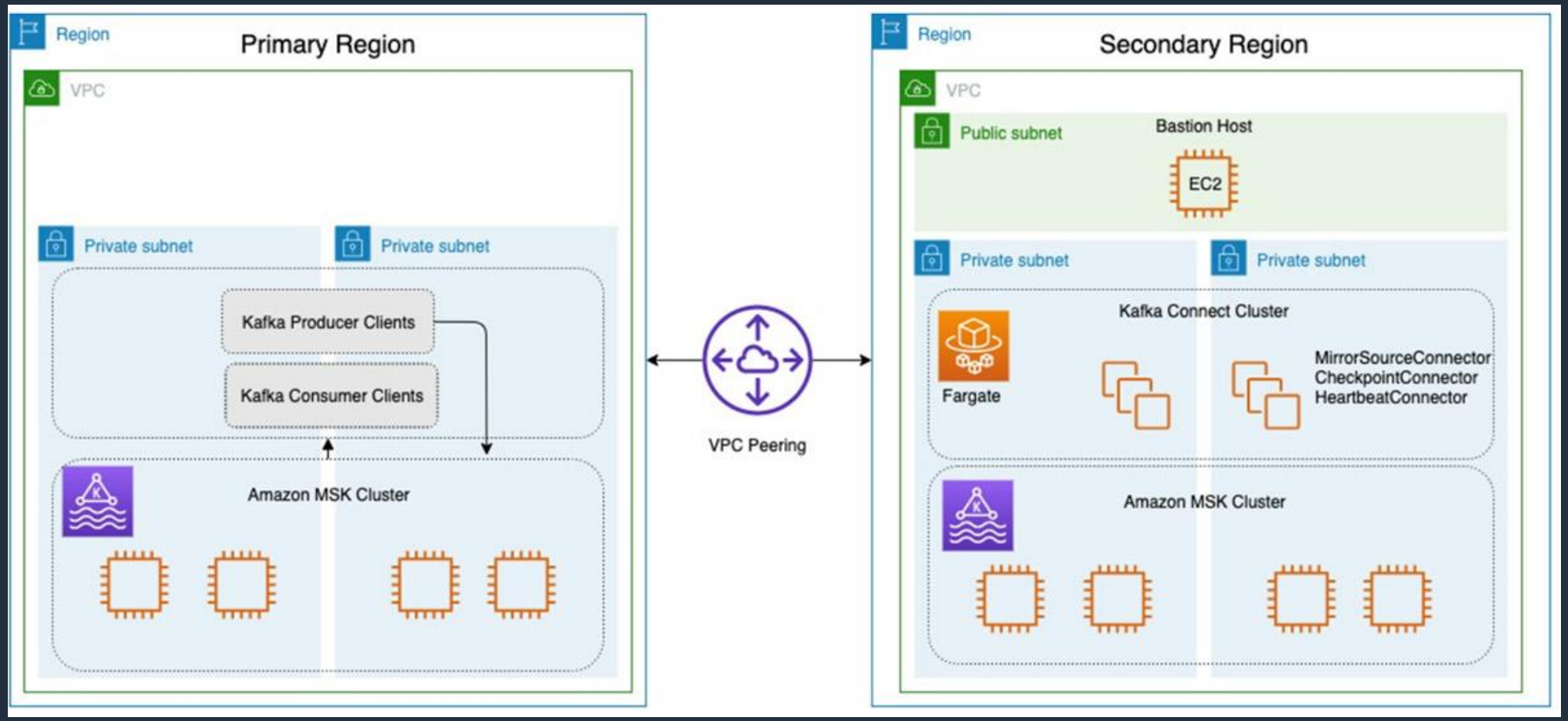
Mirror Maker 2.0 以降は Kafka Connect 上で実行される



https://docs.aws.amazon.com/whitepapers/latest/amazon-msk-migration-guide/mirrormaker-2.0-mm2.html
https://aws.amazon.com/jp/blogs/big-data/increase-apache-kafkas-resiliency-with-a-multi-region-deployment-and-mirrormaker-2/

Mirror Maker 2.0 の実行環境

Mirror Maker 2.0 は Scala で実装されたツールであるため、
EC2、ECS など様々な環境で実行可能

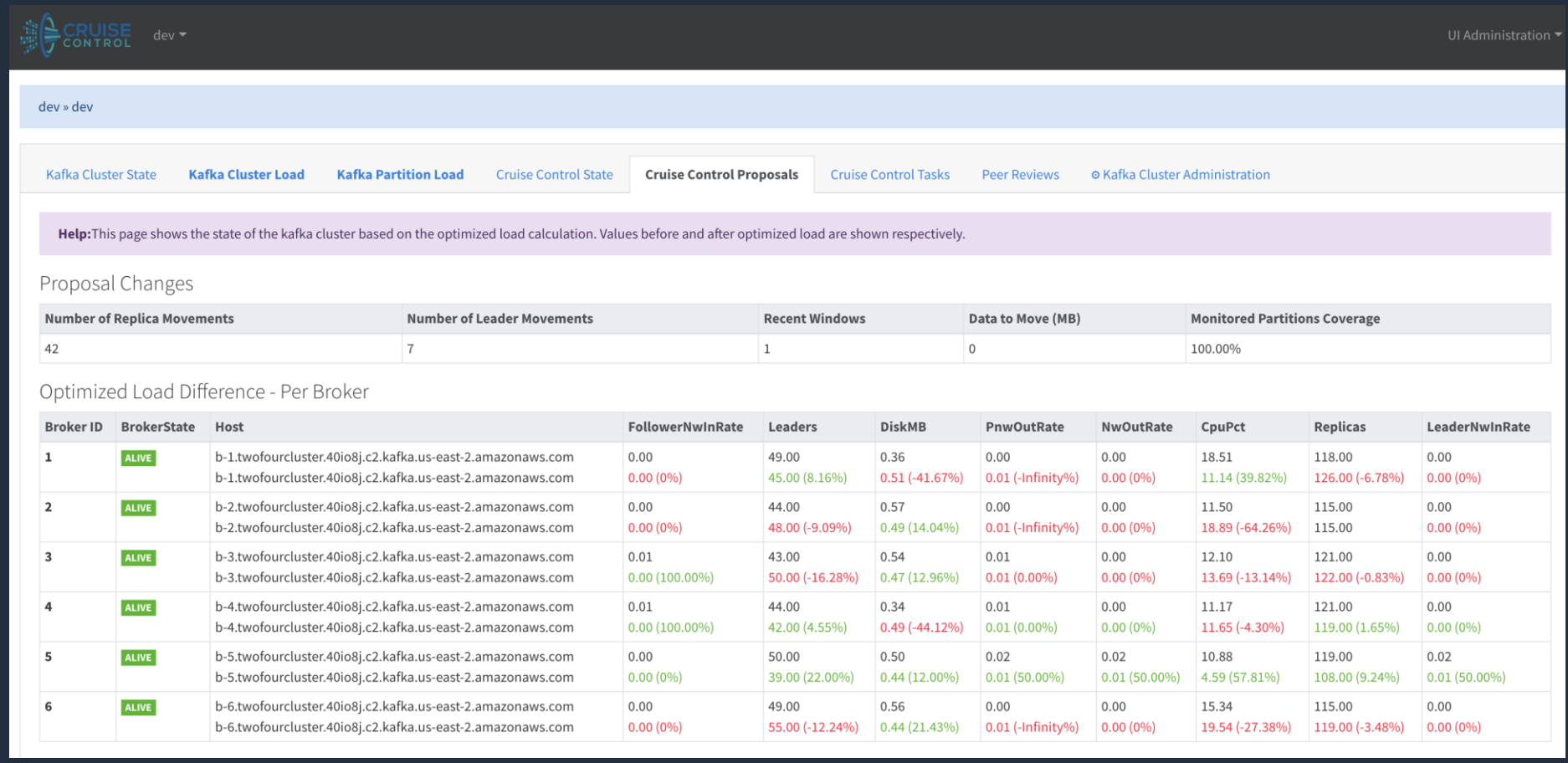


https://docs.aws.amazon.com/whitepapers/latest/amazon-msk-migration-guide/mirrormaker-2.0-mm2.html
https://aws.amazon.com/jp/blogs/big-data/increase-apache-kafkas-resiliency-with-a-multi-region-deployment-and-mirrormaker-2/

Cruise Control for Apache Kafka

LinkedIn によって開発された、大規模なクラスター運用を支えるソフトウェア

モニタリング、異常検知と復旧、負荷状況に応じたパーティションのリバランス機能などを提供



https://catalog.workshops.aws/msk-labs/en-US/cruisecontrol

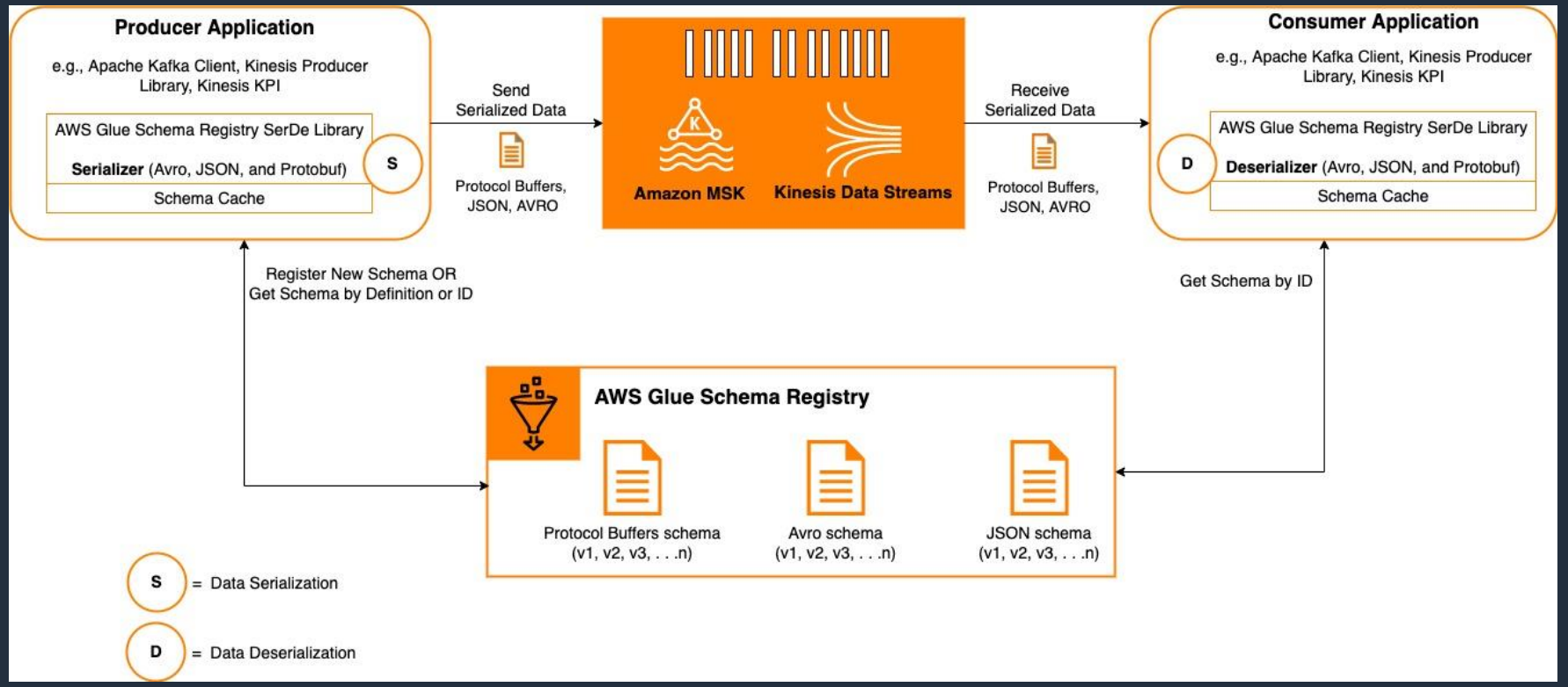
Amazon MSK の特徴

AWS サービスとの統合

データ ソースとしての AWS IoT、データ コンシューマーとしての AWS Lambda、AWS Glue Schema Registry によるスキーマ管理、Amazon Kinesis Data Analytics によるストリーム処理

Glue Schema Registry によるスキーマ管理

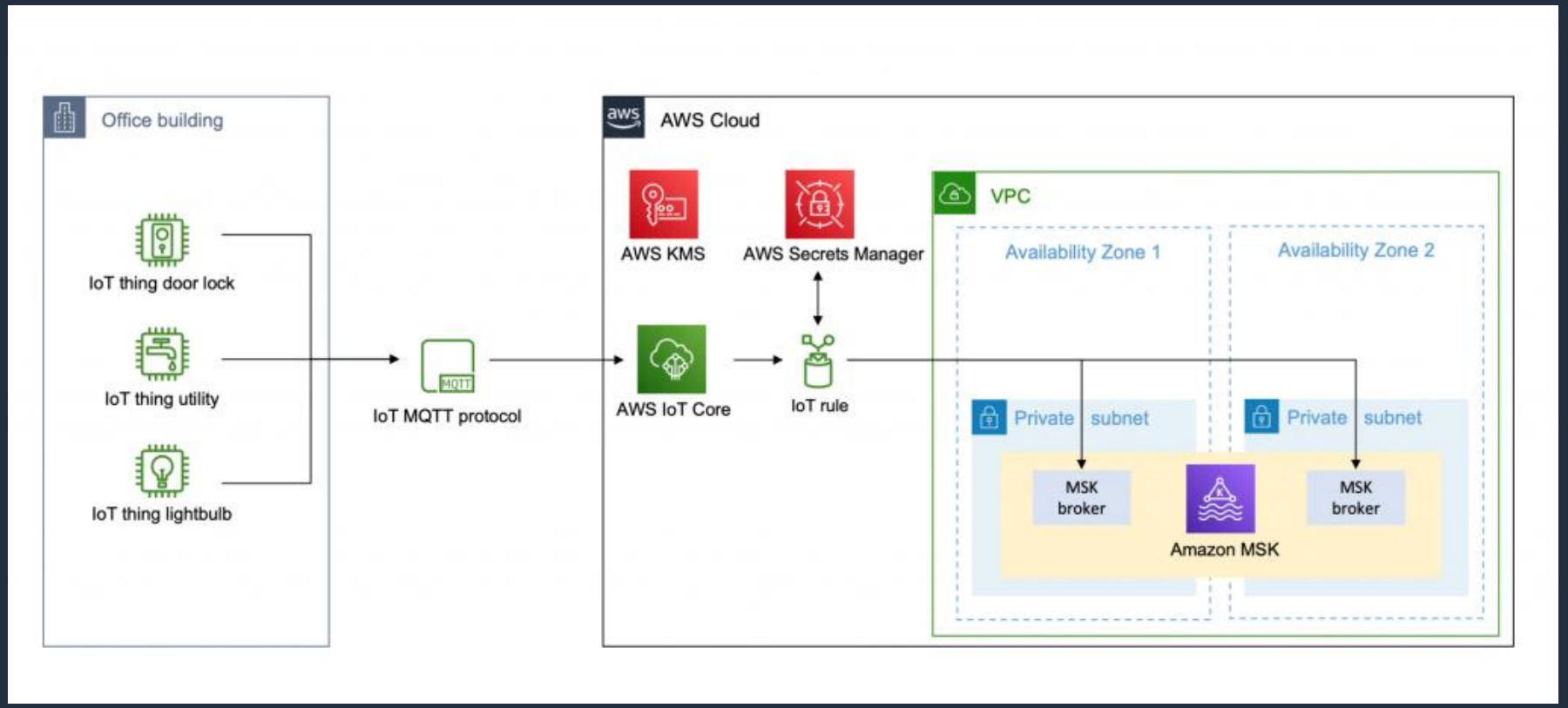
- ストリーミング処理におけるスキーマをバージョン管理することで、変更時の変化にも対応
- ライブラリを使用することで透過的なシリアライズ、デシリアライズを行うことが可能
- Protobuf, Avro, JSON フォーマットがサポートされている



https://aws.amazon.com/jp/blogs/big-data/introducing-protocol-buffers-protobuf-schema-support-in-amazon-glue-schema-registry/
https://github.com/awslabs/aws-glue-schema-registry

AWS IoT からのイベント連携

- MQTT で受信したイベントを IoT Core から直接 MSK に配信
- TLS クライアント認証もしくは SASL/SCRAM 認証が利用可能。認証情報は Secrets Manager に格納し、接続時に参照する



https://aws.amazon.com/jp/blogs/iot/how-to-integrate-aws-iot-core-with-amazon-msk/

Amazon Redshift Streaming Ingestion

AMAZON MSK のデータをダイレクトに REDSHIFT に取り込み

ストリームデータを直接参照するマテリアライズドビューを作成

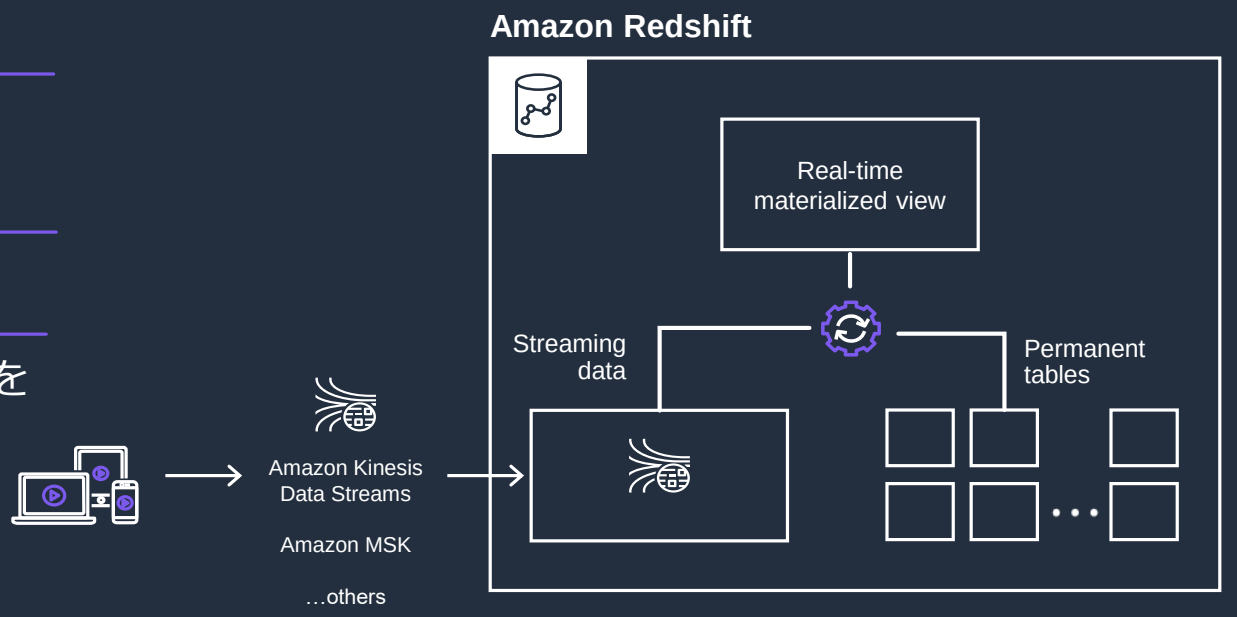
数秒の低レイテンシを実現

最大数百 MB のスループットをサポート

SQL のみで設定可能

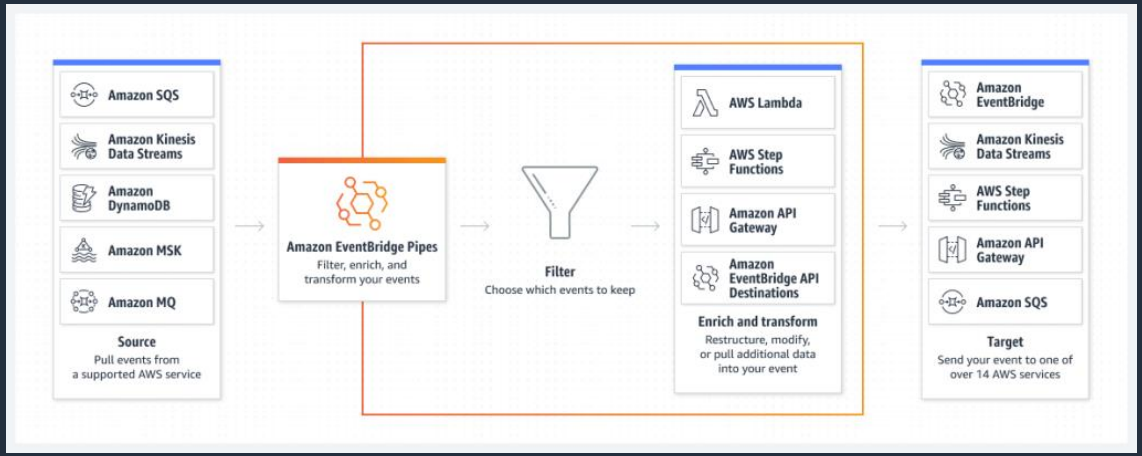
スキーマ定義、もしくは SUPER および VARBYTE データ型を使用しての半構造化データ取り込みが可能

Learn more: https://go.aws/3eG5kZF



Amazon EventBridge Pipes によるイベントの配信

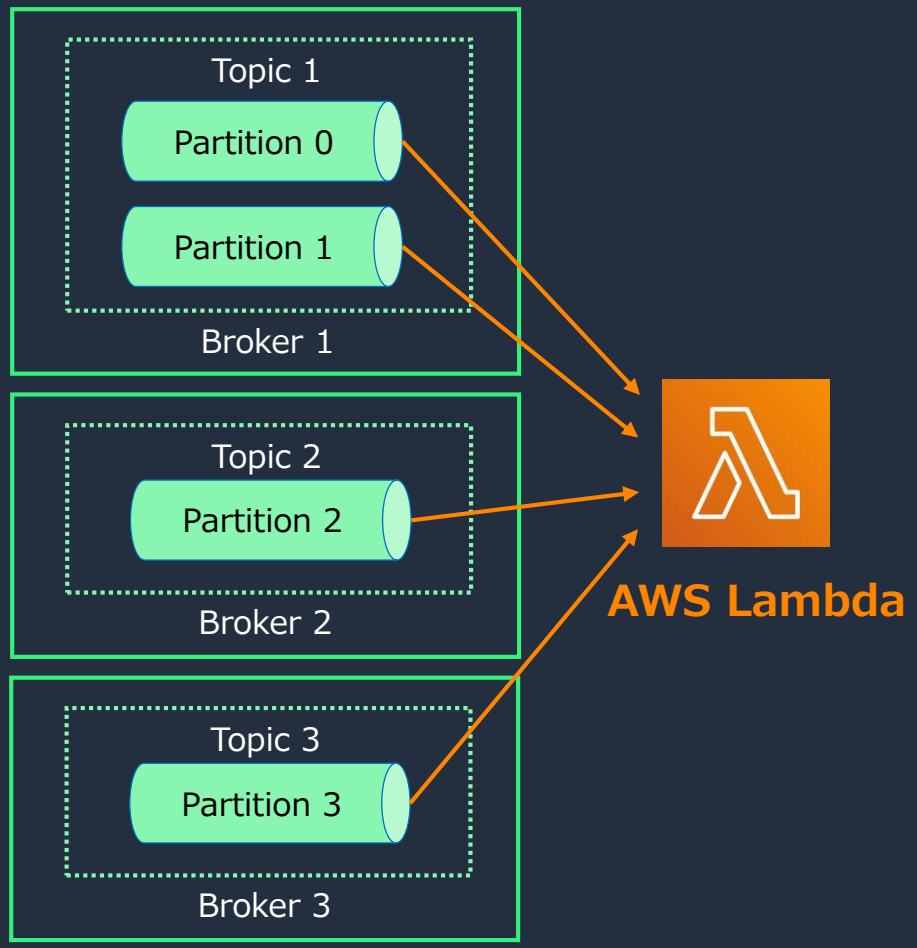
- イベントバス不要。ソースとターゲットを
  シームレスに接続
- イベントのフィルタリング、変換、
  Amazon API Gateway、AWS Step Functions、
  カスタム API と連携したエンリッチも可能
- 複数のイベントをバッファリングし、
  マイクロバッチとして一括処理することも可能
- ソース側でイベントの順序性が保証されている
  場合、順序を保ったままターゲットに配信
- Amazon MSK -> Amazon SQS など異なる
  サービス間の連携をノーコード/ローコードで
  実現



- **<u>ソースとして利用可能なサービス</u>**
  Amazon SQS, Amazon Kinesis, Amazon DynamoDB, **Amazon MSK**, セルフマネージド型 Apache Kafka, Amazon MQ
- **<u>ターゲットとして利用可能なサービス</u>**
  Amazon Kinesis Data Streams, Amazon Kinesis Data Firehose など、 イベントバスで利用できるターゲットと同様

AWS Lambda によるイベントの取得

Amazon MSK をサーバーレスのデータ処理フローに統合

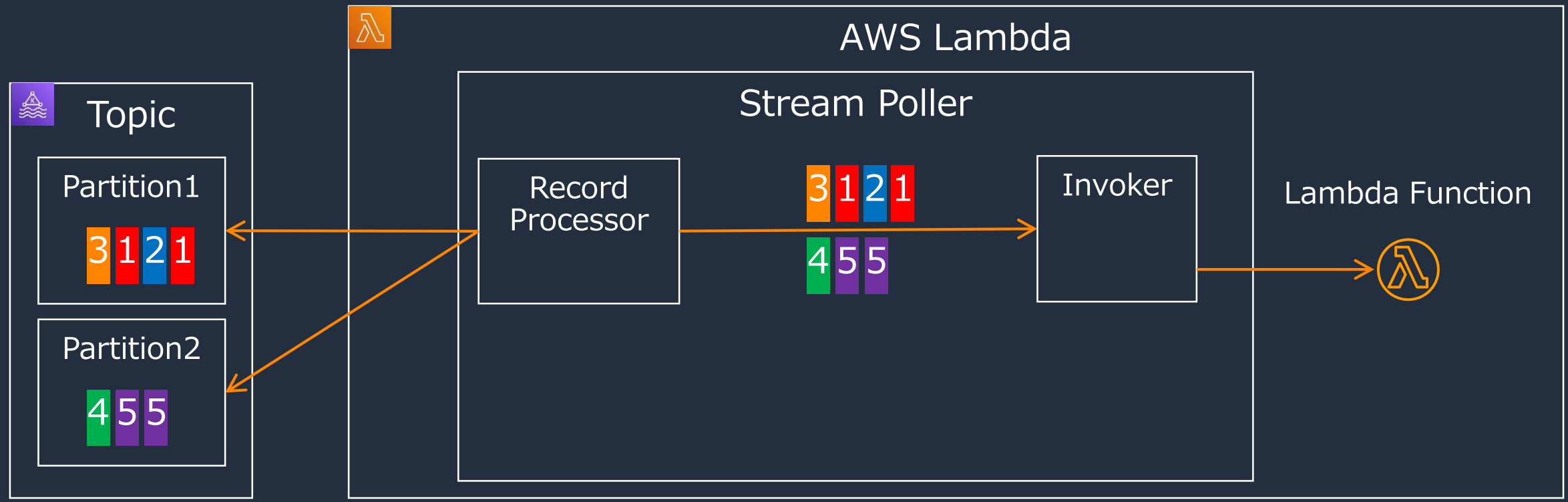



```
event: {
   "eventSource": "aws:kafka",
   "eventSourceArn": "arn:aws:kafka:us-west-
2:012345678901:cluster/ExampleMSKCluster/e9f754c6-d29a-4430-a7db-
958a19fd2c54-4",
   "records": {
    "AWSKafkaTopic-0": [
     {
      "topic": "AWSKafkaTopic",
      "partition": 0,
      "offset": 0,
      "timestamp": 1595035749700,
      "timestampType": "CREATE_TIME",
      "key": "OGQ1NTk2YjQtMTgxMy00MjM4LWIyNGItNmRhZDhlM2QxYzBj",
      "value": "OGQ1NTk2YjQtMTgxMy00MjM4LWIyNGItNmRhZDhlM2QxYzBj"
     }
    ]
   }
 }
```


https://docs.aws.amazon.com/lambda/latest/dg/lambda-kafka.html

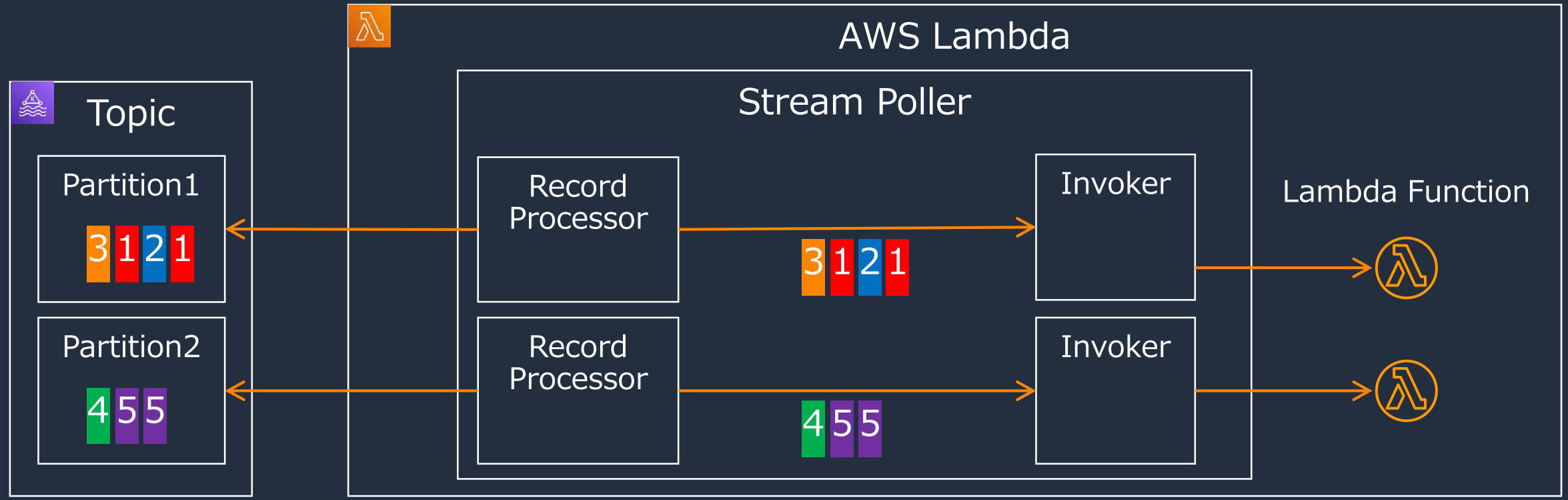
Lambda Consumer のアーキテクチャ

- 最初は全パーティションのデータを 1 つの Lambda Function で処理



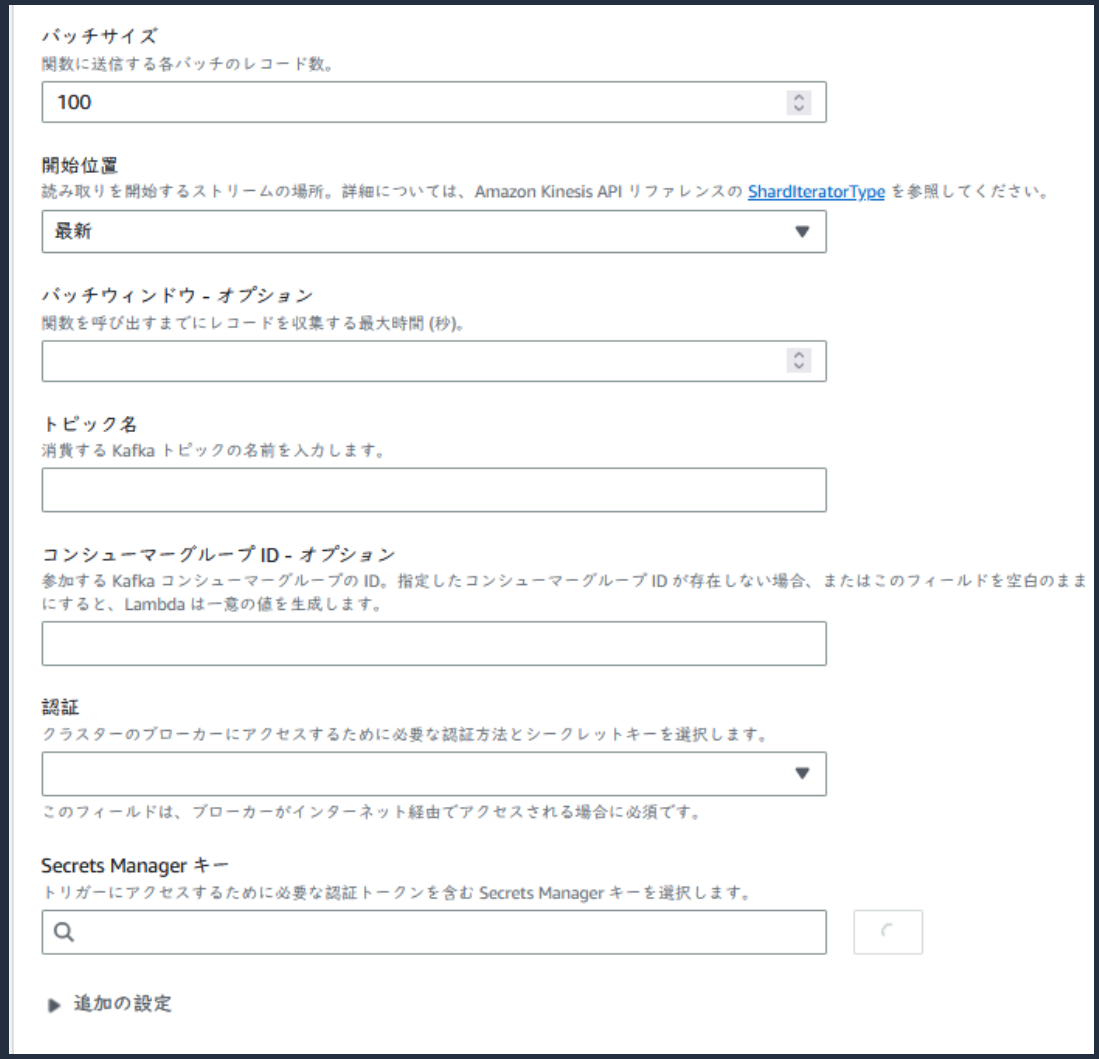
Lambda Consumer のアーキテクチャ

- 最初は全パーティションのデータを 1 つの Lambda Function で処理
- 1 分毎に OffsetLag メトリクスから Consumer の処理遅延を評価し、 遅延が判断された場合は Lambda Function 呼び出しの並列数が増加する。スケーリング処理は評価から 3 分以内に開始される。最大並列数はパーティション数と同数
- Parallelization Factor には未対応



Lambda Consumer の Event Source Mapping

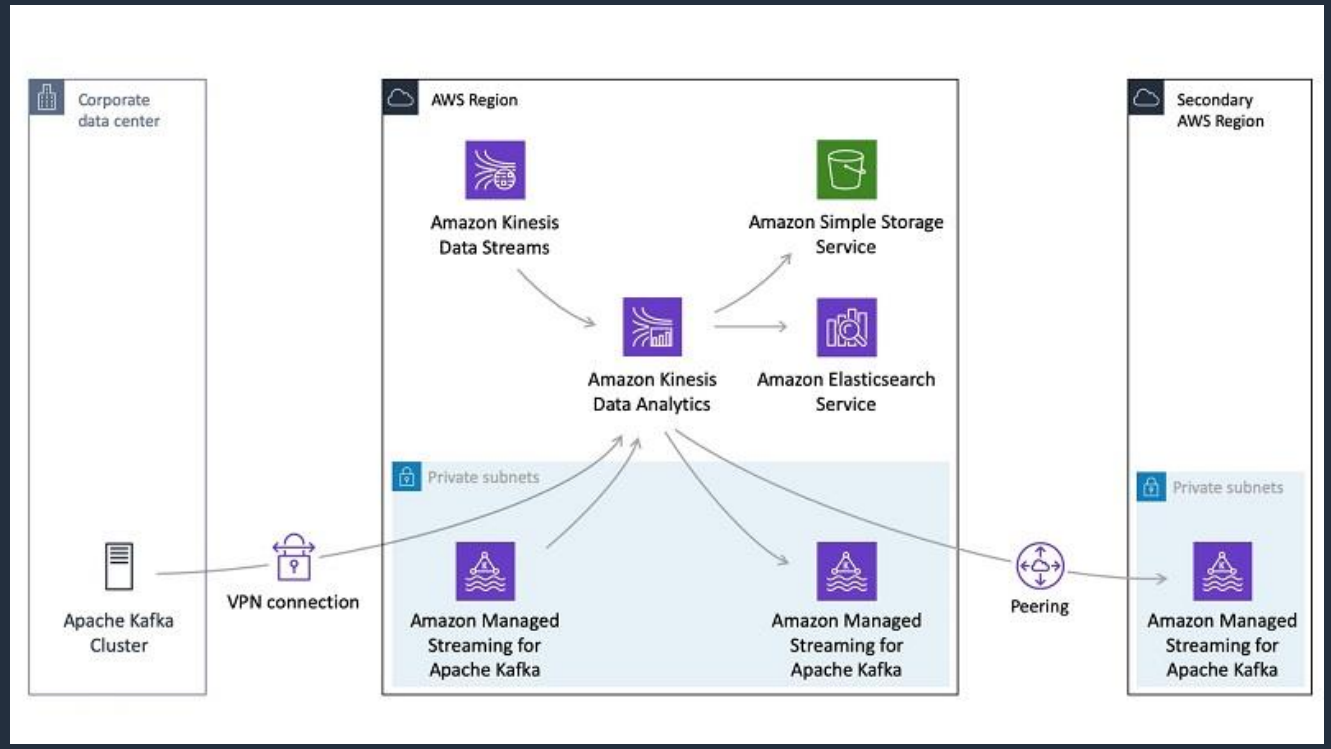
- Topic ごとに ESM を作成する
- Lambda Consumer は Consumer Group をサポート
  - Mirror Maker 2.0 の複製対象となっているトピックに対して、
    別 Consumer として Lambda を割り当てて
    イベントを処理することが可能
  - Consumer Group ID は Event Source Mapping 側で指定する
- MSK が提供する各種認証方式を利用可能
  - TLS クライアント認証、SASL/SCRAM を
    利用する場合は Secrets Manager キーを指定
- 実行時間のタイムアウト上限が **14 分**となる
  点に注意 (通常は 15 分)



https://docs.aws.amazon.com/ja_jp/lambda/latest/dg/with-msk.html

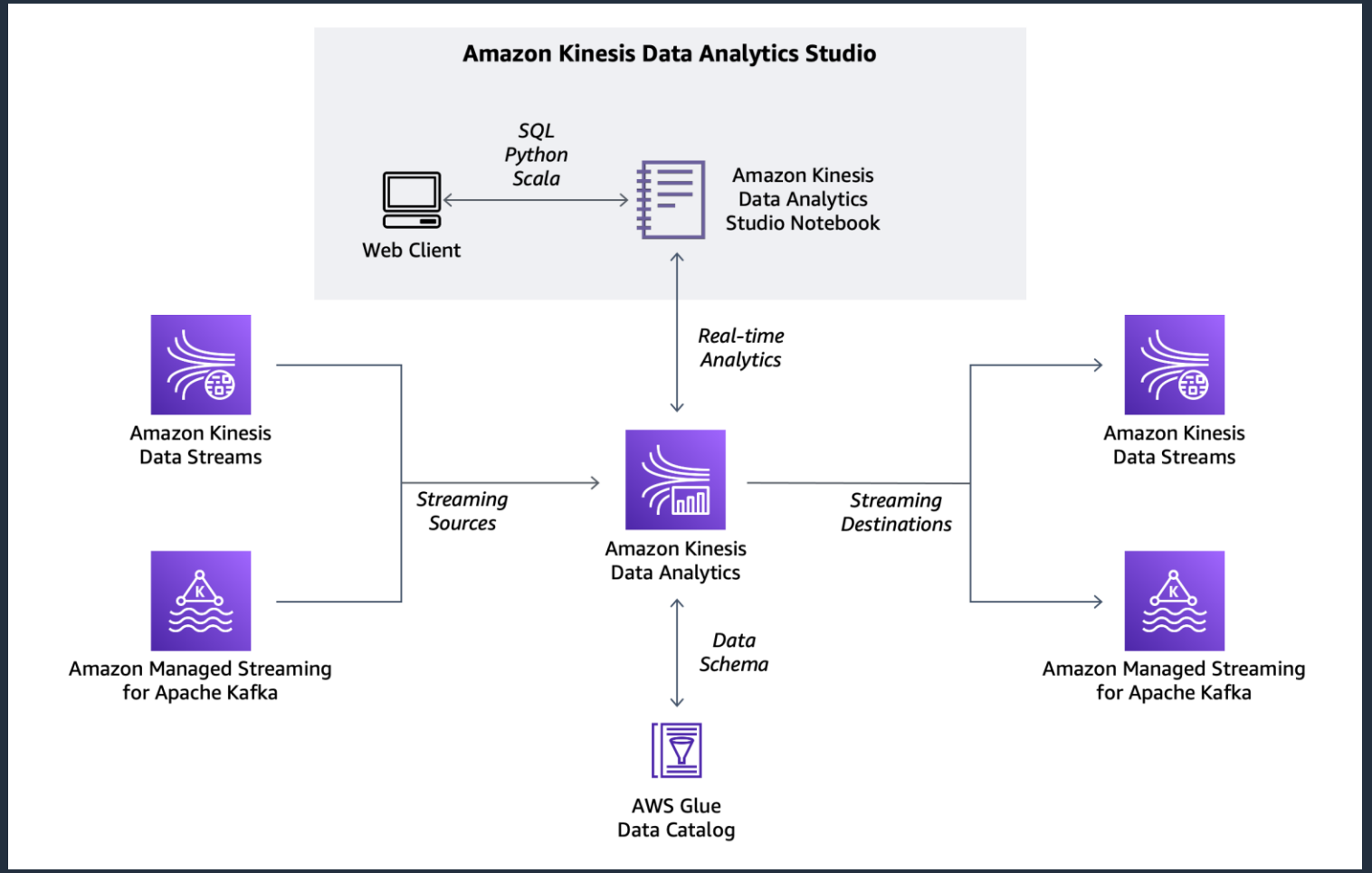
Amazon Kinesis Data Analytics for Flink との連携

- Kinesis Data Analytics for Flink は、Flink アプリケーションを実行するフルマネージドサービス
- Flink は Kafka をはじめとして多数のサービス、データストアに対応しており、柔軟なデータ処理、データ連携を実現可能
- Java, SQL, Python による記述をサポート



Amazon Kinesis Data Analytics Studio を利用した開発

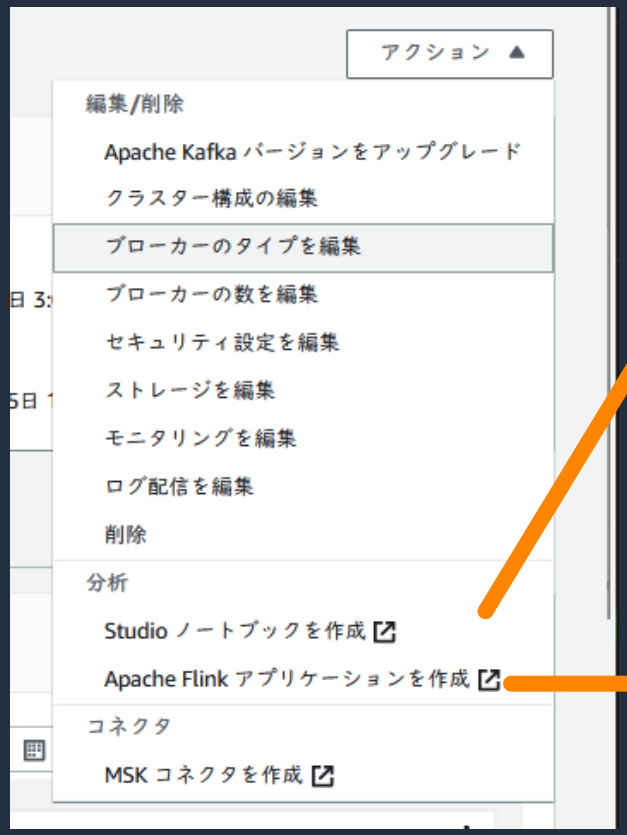
- マネージドな Zeppelin ノートブックを使用して、アドホックなトピックのデータ分析が可能
- 作成した分析ロジックを Flink アプリケーションとして保存、本番へデプロイすることも可能
- Java, Scala, Python, SQL など複数言語に対応

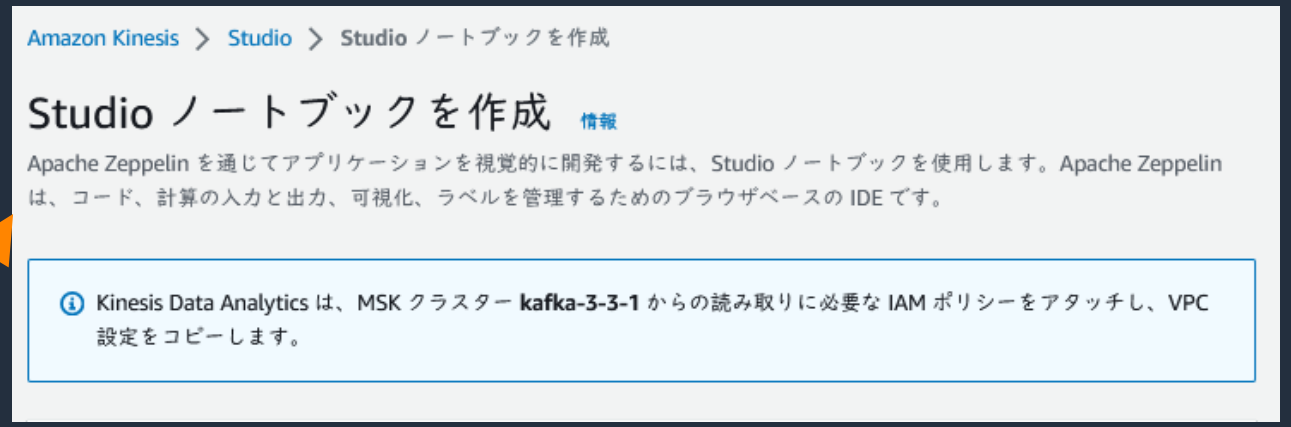


https://aws.amazon.com/jp/blogs/news/introducing-amazon-kinesis-data-analytics-studio-quickly-interact-with-streaming-data-using-sql-python-or-scala/

Flink アプリケーション、ノートブックの作成

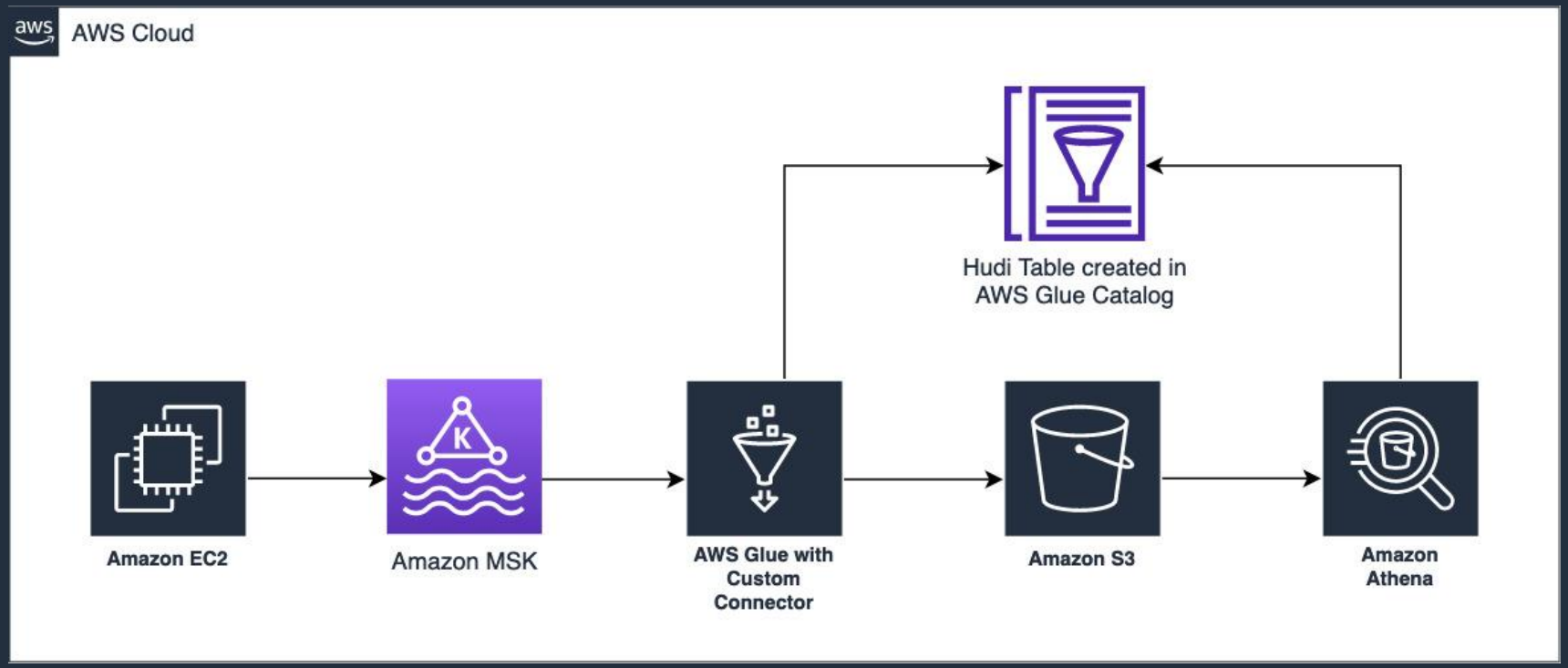
MSK コンソールから Kinesis Data Analytics のアプリケーション、ノートブック作成のウィザードを呼び出すことが可能





AWS Glue Streaming ETL ジョブによるデータ処理

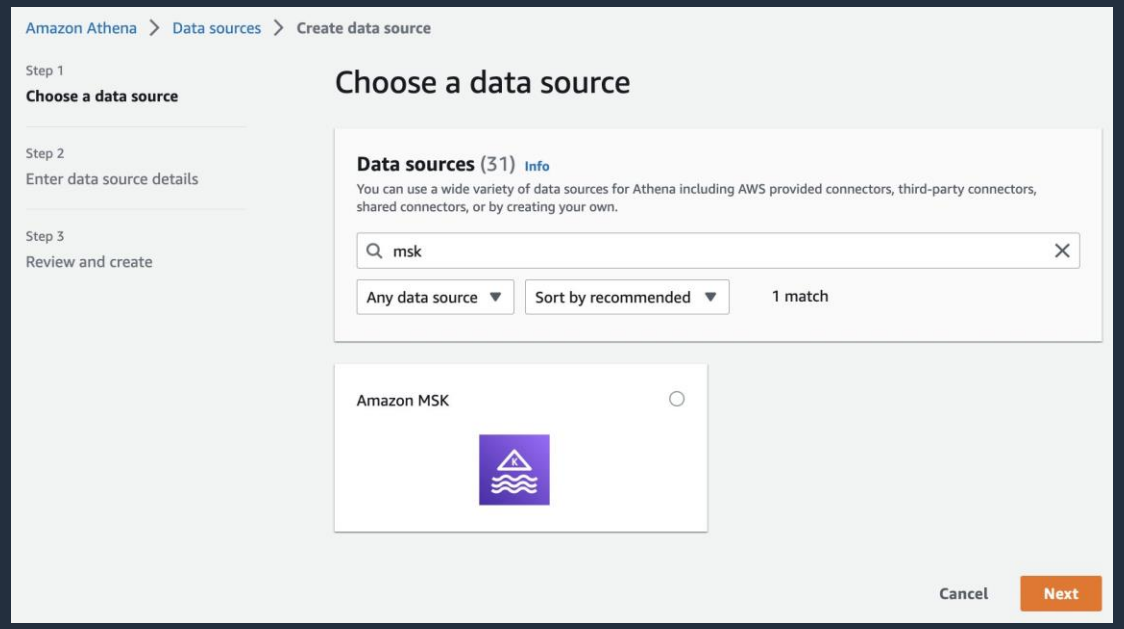
- AWS Glue ではストリームデータを継続的に処理するための Spark Streaming ジョブタイプを使用可能
- Glue がネイティブサポートしている Apache Hudi などのデータレイクストレージフレームワークと連携させることで、データレイクに対するニアリアルタイムな増分データ処理も実装可能



https://docs.aws.amazon.com/ja_jp/glue/latest/dg/add-job-streaming.html
https://aws.amazon.com/jp/blogs/big-data/ingest-streaming-data-to-apache-hudi-tables-using-aws-glue-and-apache-hudi-deltastreamer/

Amazon Athena よるストリームデータへのクエリ実行

- Athena の MSK Connector を使用することでトピック内のデータを直接取得することが可能。アドホック分析やデータ加工に活用できる
- Lambda Consumer と同様に様々な認証方法をサポート




```
# 2 つのトピック内のデータを結合した結果を取得
SELECT t1.order_id, t2.item_id
FROM msk.customer_schema.orders as t1
JOIN msk.customer_schema.items as t2
ON t1.id = t2.id
```

```
# トピックに対するクエリ実行結果を S3 に保存
CREATE TABLE my_kafka_data
WITH (format = 'Parquet',
      write_compression = 'SNAPPY')
AS
SELECT order_id, item_id, timestamp
FROM msk.customer_schema.orders
```


https://docs.aws.amazon.com/athena/latest/ug/connectors-msk.html
https://aws.amazon.com/jp/blogs/big-data/analyze-real-time-streaming-data-in-amazon-msk-with-amazon-athena/

Amazon MSK の特徴

AWS サービスとの統合

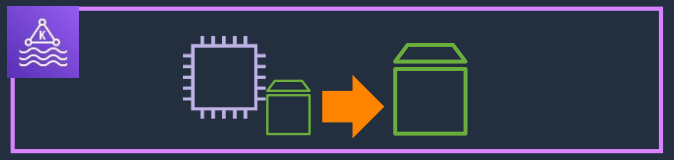
データ ソースとしての AWS IoT、データ コンシューマーとしての
AWS Lambda、AWS Glue Schema Registry によるスキーマ管理、
Amazon Kinesis Data Analytics によるストリーム処理

スケーラビリティ

オンラインでのブローカーの追加、ブローカーのサイズの変更、ストレージの追加。

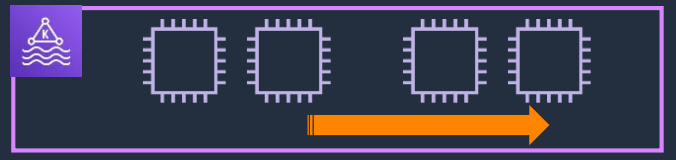
Tired Storage による低コストのデータ長期保管

Amazon MSK クラスターのスケーリング



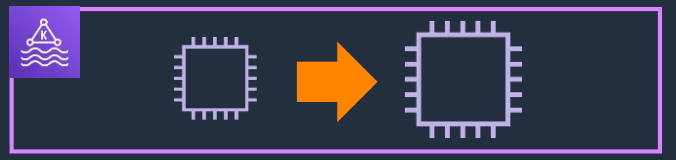
ストレージの拡張

- ストレージサイズの増加、スループットの増減が可能
- ストレージサイズについては自動スケールもサポート
- ストレージサイズを減らすことはできない
- 全ブローカーが同じサイズ、スループットに統一される



ブローカーの追加

- オンラインで処理される
- 指定可能なブローカー数はAZ 数の倍数
- ブローカーを減らす(スケールイン)は不可
- ブローカー追加後にパーティションを再配置する必要あり



ブローカーサイズの変更

- ローリングアップグレードにより、同時にオフラインになるブローカーは 1 つのみ
- 適切なレプリカ数を設定していれば、サイズ変更中もトピックに対するイベントの配信、取得は可能
- 条件付きでスケールダウンも可能

https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/msk-update-storage.html
https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/msk-update-broker-count.html
https://docs.aws.amazon.com/msk/latest/developerguide/msk-update-broker-type.html

スケールアップ vs スケールアウト

- スケールアップはローリングリスタートが発生するものの、パーティション再割り当てが不要で、スケールダウンも可能などのメリットがある。スケールアウトはリスタートは伴わないものの、スケールインができないなど不可逆であることがデメリットとなる
- 1 ノードにそれなりのリソースが割り当てられている MSK Provisioned Cluster では、スケールアップ>スケールアウトの順番でスケールを考慮するとよい

スケールアウト

- Kafka ブローカーの追加
- AZ 数の倍数である必要あり
- スケールアウトのみ、スケールイン不可
- パーティションの再割り当てが必要
- ブローカー単体の性能が頭打ち、ボトルネックになる場面で有効

スケールアップ

- Kafka ブローカーのインスタンスタイプ変更
- スケールアップ、スケールダウン可能
- クラスターの I/O 停止無し
- パーティションの再割り当ては不要
- ブローカーあたりの制限がある点は要考慮 (ディスクサイズ、パーティション数、帯域)

https://aws.amazon.com/jp/blogs/big-data/best-practices-for-right-sizing-your-apache-kafka-clusters-to-optimize-performance-and-cost/

サポートされるブローカータイプ・スケールの上限

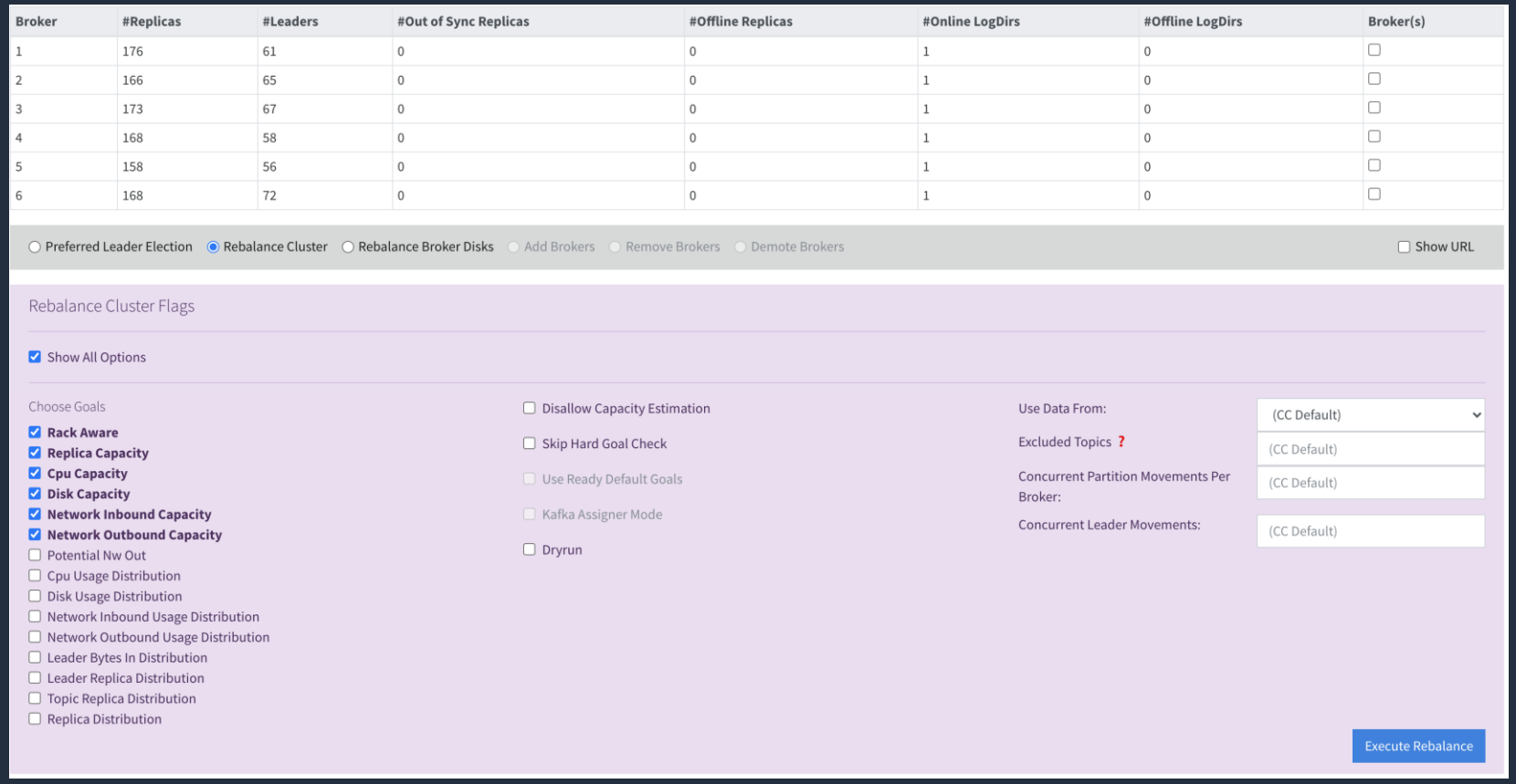
- リソース、推奨上の最大パーティション数をベースにスケールアップ、スケールアウトを検討
- デフォルトでは 1 クラスターに 30 ブローカーまで配置可能。
  更にノードの追加が必要な場合は、カスタマーサービスに上限緩和を申請する

ブローカータイプ	vCPU	Memory (GiB)	ネットワーク帯域 (Gbps)	推奨上の最大パーティション数	最大ストレージスループット (MiB/second)
kafka.t3.small	2	2	5	300	250
kafka.m5.large	2	8	10	1000	250
kafka.m5.xlarge	4	16	10	1000	250
kafka.m5.2xlarge	8	32	10	2000	250
kafka.m5.4xlarge	16	64	10	4000	593
kafka.m5.8xlarge	32	128	10	4000	850
kafka.m5.12xlarge	48	192	12	4000	1000
kafka.m5.16xlarge	64	256	20	4000	1000
kafka.m5.24xlarge	96	384	25	4000	1000

https://docs.aws.amazon.com/msk/latest/developerguide/bestpractices.html#partitions-per-broker

ブローカー追加時のパーティション再割り当て

Cruise Control や、Kafka に含まれる
kafka-reassign-partitions.sh などを利用して実施



Broker	#Replicas	#Leaders	#Out of Sync Replicas	#Offline Replicas	#Online LogDirs	#Offline LogDirs	Broker(s)
1	176	61	0	0	1	0	☐
2	166	65	0	0	1	0	☐
3	173	67	0	0	1	0	☐
4	168	58	0	0	1	0	☐
5	158	56	0	0	1	0	☐
6	168	72	0	0	1	0	☐

https://catalog.workshops.aws/msk-labs/en-US/addingbrokers/reassignpartitions
https://catalog.workshops.aws/msk-labs/en-US/cruisecontrol/cc-rebalance

ストレージの手動拡張

ストレージサイズ、スループットともに、 6 時間につき 1 回までの
変更可能制限がある点に注意

ストレージサイズ

- 1 GiB – 16384 GiB まで拡張可能。一度拡張したボリュームは縮小不可

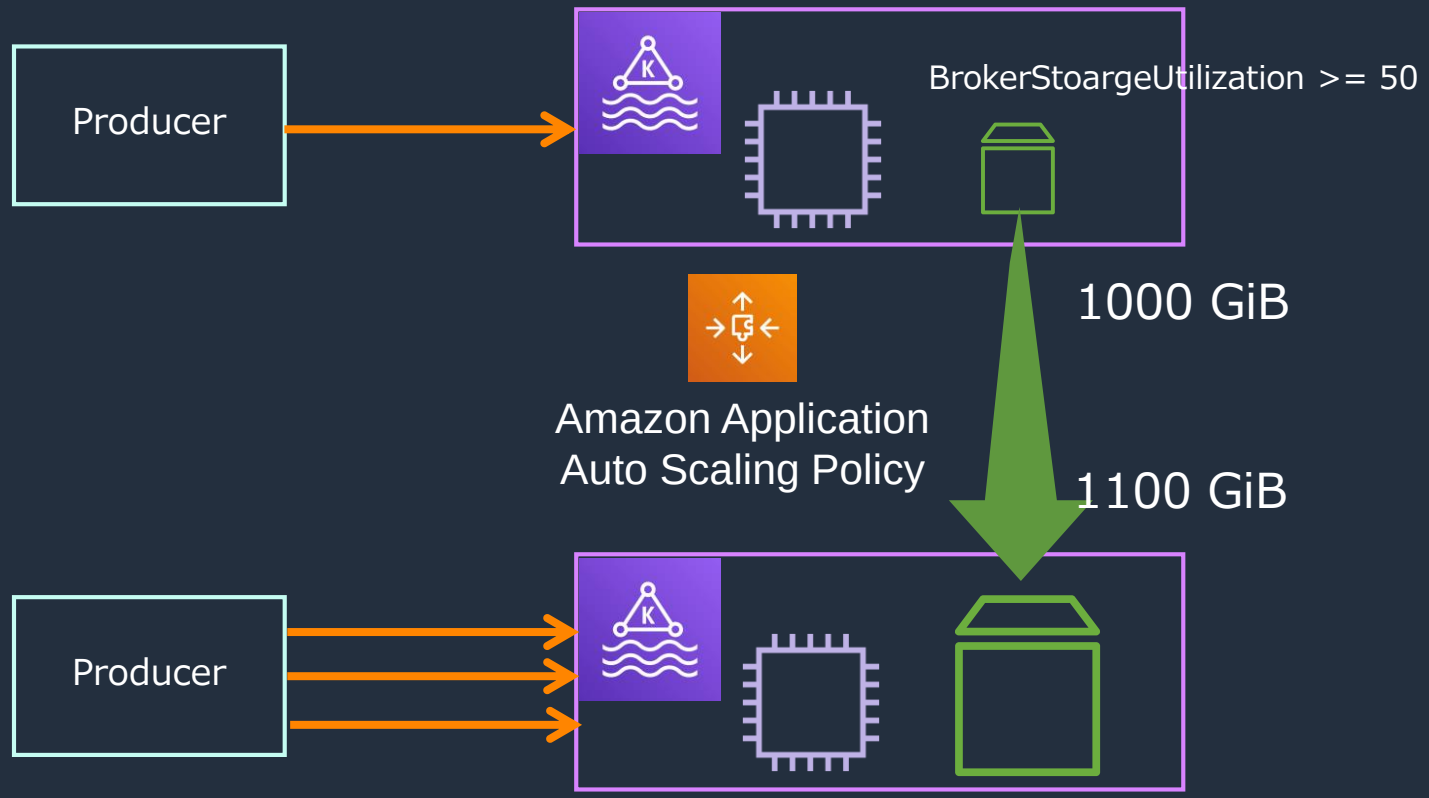
スループット

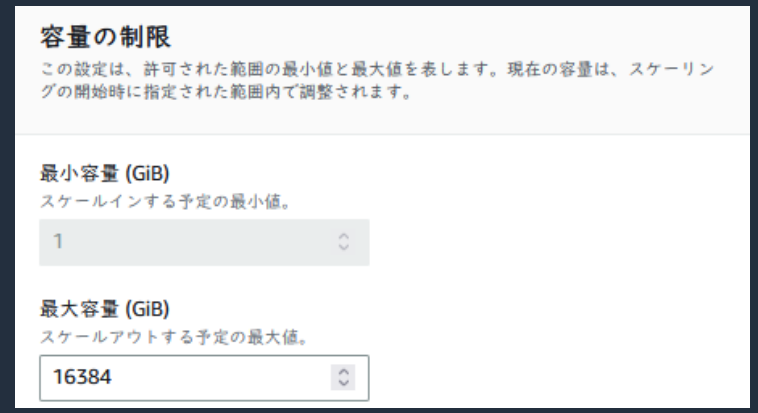
- デフォルトは 250 MB/s
- ブローカータイプが kafka.m5.4xlarge 以上、かつストレージサイズが
10 GiB 以上の場合は、追加スループットを割り当て可能。
スループット上限はブローカータイプに依存
- 追加のスループット割り当てを有効化すると、レプリカスレッド
(num.replica.fetchers) を増やすことが可能になる

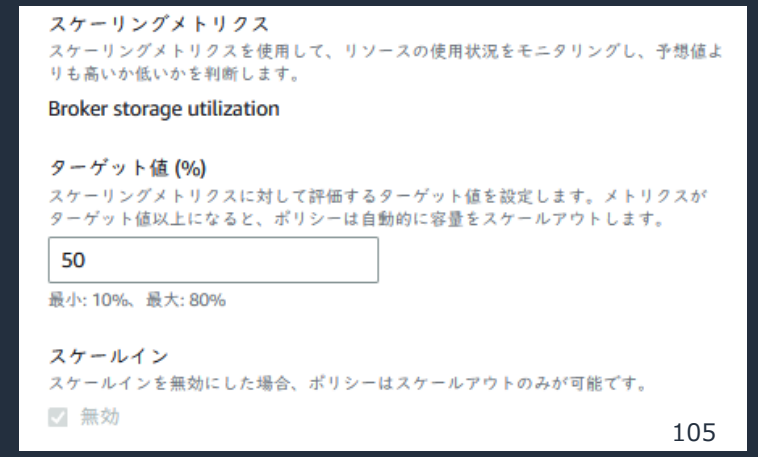
https://docs.aws.amazon.com/msk/latest/developerguide/msk-provision-throughput.html
https://docs.aws.amazon.com/msk/latest/developerguide/manually-expand-storage.html

クラスターストレージ領域の自動拡張

- Application Auto Scaling と連携してストレージ領域の監視・自動拡張を行う
- 拡張幅は元のストレージサイズの 10%。ストレージサイズが 100 GiB 未満の場合は 10 GiB ずつ拡張
- スケーリングのターゲット値は 50% - 60% を推奨。スケールは 6 時間に 1 度の制限あり



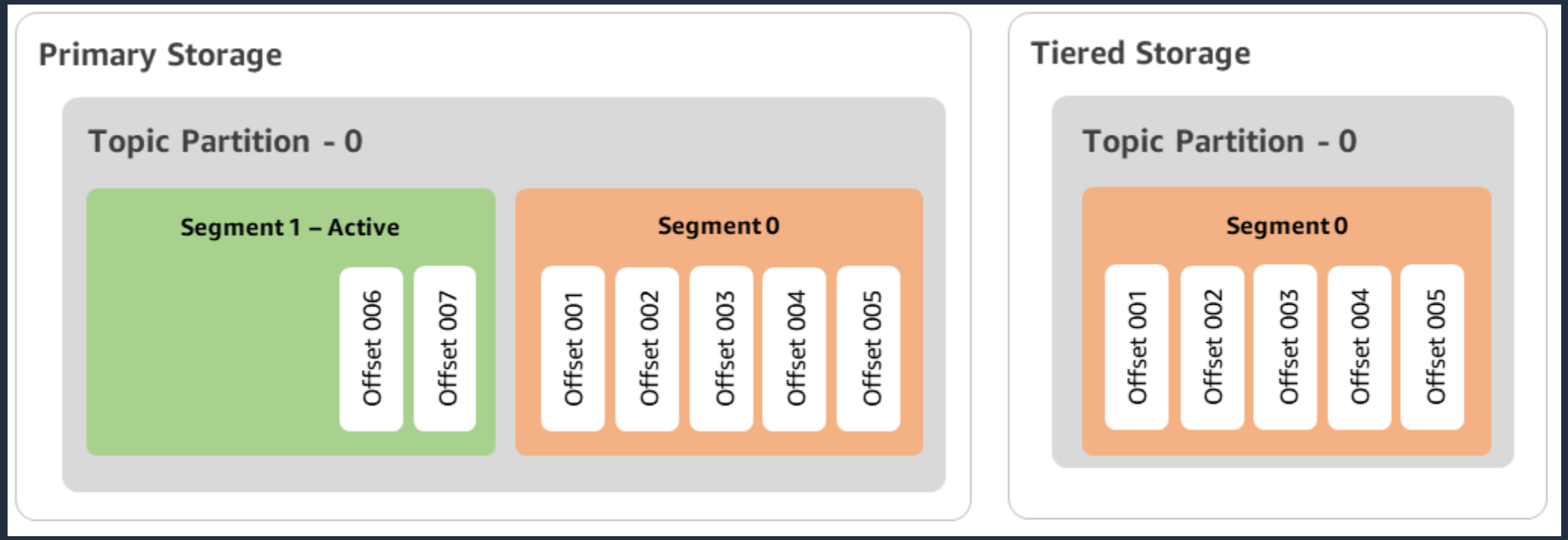




https://docs.aws.amazon.com/msk/latest/developerguide/msk-autoexpand.html

階層型ストレージ (Tiered Storage)

- ローカルストレージとリモートストレージを組み合わせた階層構造を構築することが可能
- ローカルからリモートへの移動はセグメント単位で行われる。また VPC を介して行われる
- トピック単位で有効・無効の切り替え、リモートストレージへの移動タイミングを制御可能



https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/msk-tiered-storage.html

階層型ストレージにおけるデータ保持期間の調整

セグメントのローテーション、ローカルストレージからリモートストレージへのデータ移行、リモートストレージ上のデータ削除のタイミングを制御可能

- **segment.bytes, segment.ms**
  セグメントのローテーションを判断するための閾値
- **local.retention.ms, local.retention.bytes**
  ローカルストレージからリモートストレージにデータを移動する閾値
- **log.retention.ms, log.retention.bytes**
  リモートストレージも含むトピック内の**ログ**保管期間、または保管上限

階層型ストレージの制限事項

- 現状はクラスターバージョンが 2.8.2.tiered である必要あり
- **log.cleanup.policy** が **compact** の場合は使用不可
- ブローカータイプに t3.small を使用している場合は使用不可

その他詳細な制約については階層型ストレージのドキュメントに記載有り

https://docs.aws.amazon.com/msk/latest/developerguide/msk-tiered-storage.html#msk-tiered-storage-constraints

クラスターサイジングのポイント

- サイジングシートを元に机上サイジングを行うことが可能
- レイテンシ要求、暗号化の有無、パーティション数、リテンションなどの要件を明確にすることで、サイジングの精度が向上する
- サイジングシートによるサイジングはあくまで机上のものに過ぎない。実ワークロードベースの負荷テストは必要
- 構成検討に際しては、ベストプラクティスを合わせて確認することを推奨

ベストプラクティス
https://docs.aws.amazon.com/msk/latest/developerguide/bestpractices.html
https://aws.amazon.com/jp/blogs/big-data/best-practices-for-right-sizing-your-apache-kafka-clusters-to-optimize-performance-and-cost/

サイジングシート
https://view.officeapps.live.com/op/view.aspx?src=https%3A%2F%2Fdy7oqpxkwhskb.cloudfront.net%2FMSK_Sizing_Pricing.xlsx&wdOrigin=BROWSELINK

パフォーマンステストツール
https://github.com/aws-samples/performance-testing-framework-for-apache-kafka/

サイジングシート

各種パラメーターから推奨構成を算出

	Input Cells
CLUSTER DESIGN INPUT PARAMETERS	
Ingestion Rate, MB/s	66.0
Replication Factor	3
Total Data Out, MB/s	66.0
Retention, Hrs	72
Retention in primary storage, Hrs (Optional)	2
Encryption in-transit	TRUE
Provisioned Storage Throughput (PST) per broker, MB/s (option	0.00
Write Latency (P99)	None
EC2 Comparison Inputs	
Engineering Cost	$300,000
Number of Engineers	1
EC2 Reserved Instances (Years) **	1
Advanced parameters	
Number of AZs (3 Recommended)	3
Nearest Replica Fetching (Kafka 2.4.x)	TRUE
Lagging Consumer Percentage	5%
Broker Safety Factor	1
EBS Disk Utilization	80%
Ram Data Cached Seconds	90
Data retrieval from low-cost storage, Percentage (Optional)	10%

Note: Hover over the input (red triangle) to get a description of the field.

SUMMARY

Selected Cluster Instance Type	m5.xlarge	Drop down selection. Select instance based on
Cluster Size (Brokers)	6	We recommend referencing the MSK default lir
Provisioned Storage per Broker (GB)	290	
Provisioned Storage Throughput per broker (MB/s)	0.00	0 means provisioned storage throughput has nc
Tiered Storage Usage (GB)	16706.25	

Cluster	Monthly Cost	$/GB	% Difference to MSK
Amazon MSK	$5,274	$0.030	-
Amazon MSK without TS	$10,363	$0.06	96%
Amazon MSK with data retrieval from tiered storage	$5,277		
Self Managed EC2	$16,102	$0.093	205%
Self Managed EC2 with Engineers	$41,102	$0.237	679%

Amazon MSK の特徴

AWS サービスとの統合

データ ソースとしての AWS IoT、データ コンシューマーとしての AWS Lambda、AWS Glue Schema Registry によるスキーマ管理、Amazon Kinesis Data Analytics によるストリーム処理

スケーラビリティ

オンラインでのブローカーの追加、ブローカーのサイズの変更、ストレージの追加。

Tired Storage による低コストのデータ長期保管

モニタリング・オブザーバビリティ

Amazon CloudWatch を介したログとメトリクスの監視、Open Monitoring for Prometheus による JMX メトリクスの抽出などをサポート

MSK のモニタリング

CloudWatch メトリクス

DEFAULT（無償）、PER_BROKER、PER_TOPIC_PER_BROKER の 3 つのモニタリングレベルを設定可能

ブローカーログを他の AWS サービスに連携

ブローカーログを CloudWatch Logs、S3、Kinesis Data Firehose に出力可能。Kinesis Data Firehose を介すことで、Amazon OpenSearch Service や外部サービスへの継続的なストリーミングを実現

Amazon MSK 独自の機能

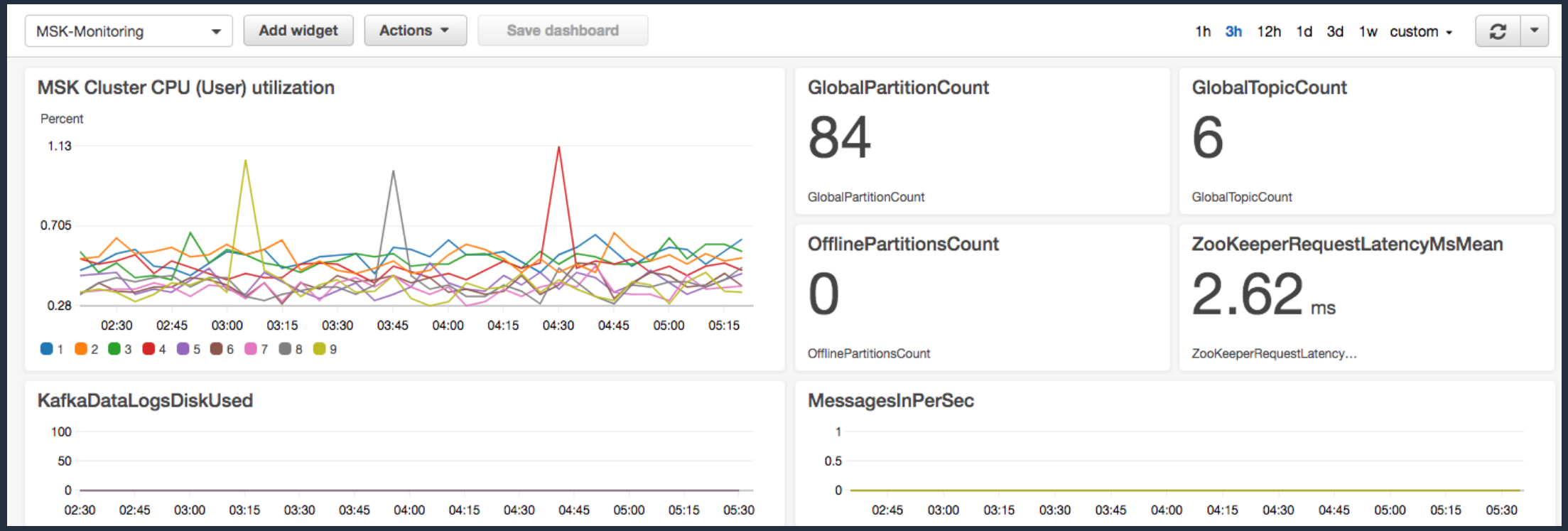
Prometheus によるオープンモニタリング

Prometheus でオープンモニタリングを有効にし、Datadog、Lenses、New Relic、Sumo Logic 等のサードパーティ互換ツールをモニタリングに利用可能

Apache Kafka としての機能

CloudWatch メトリクスによるモニタリング

- Amazon MSK から CloudWatch へのメトリクスデータの送信は自動で行われる
- 保存されたメトリクスを元にダッシュボードやアラートを利用可能



Amazon MSK が提供する CloudWatch メトリクス

- メトリクスレベルを変更することでより詳細な粒度のデータを取得可能
- ブローカーレベル以上の追加メトリクスについては料金が発生

デフォルト

- クラスター、コンシューマー、プロデューサーの全体的なスループットやエラー状況を把握する際に参照する
- 無料で利用可能

UnderReplicatedPartitions

OfflinePartitions

LeaderCount

Bytes In/Out

Connection Count

…

ブローカーレベル

- 特定ブローカーがボトルネックになっていないかを調査する際に使用する
- メトリクスはブローカーごとに生成される
- CloudWatch カスタムメトリクスの料金が発生

TrafficBytes

TcpConnections

ProduceThrottleByteRate

…

トピックレベル

- トピック単位のスループット、エラーの確認に使用する
- メトリクスはブローカー、トピックごとに生成される
- CloudWatch カスタムメトリクスの料金が発生

MessagesInPerSec

ProduceMessage-ConversionsPerSec

…

パーティションレベル

- ラグの発生状況をパーティション単位で確認する際に使用する
- メトリクスはトピック、パーティション、コンシューマーグループごとに作成される
- CloudWatch カスタムメトリクスの料金が発生

EstimatedTimeLag

OffsetLag

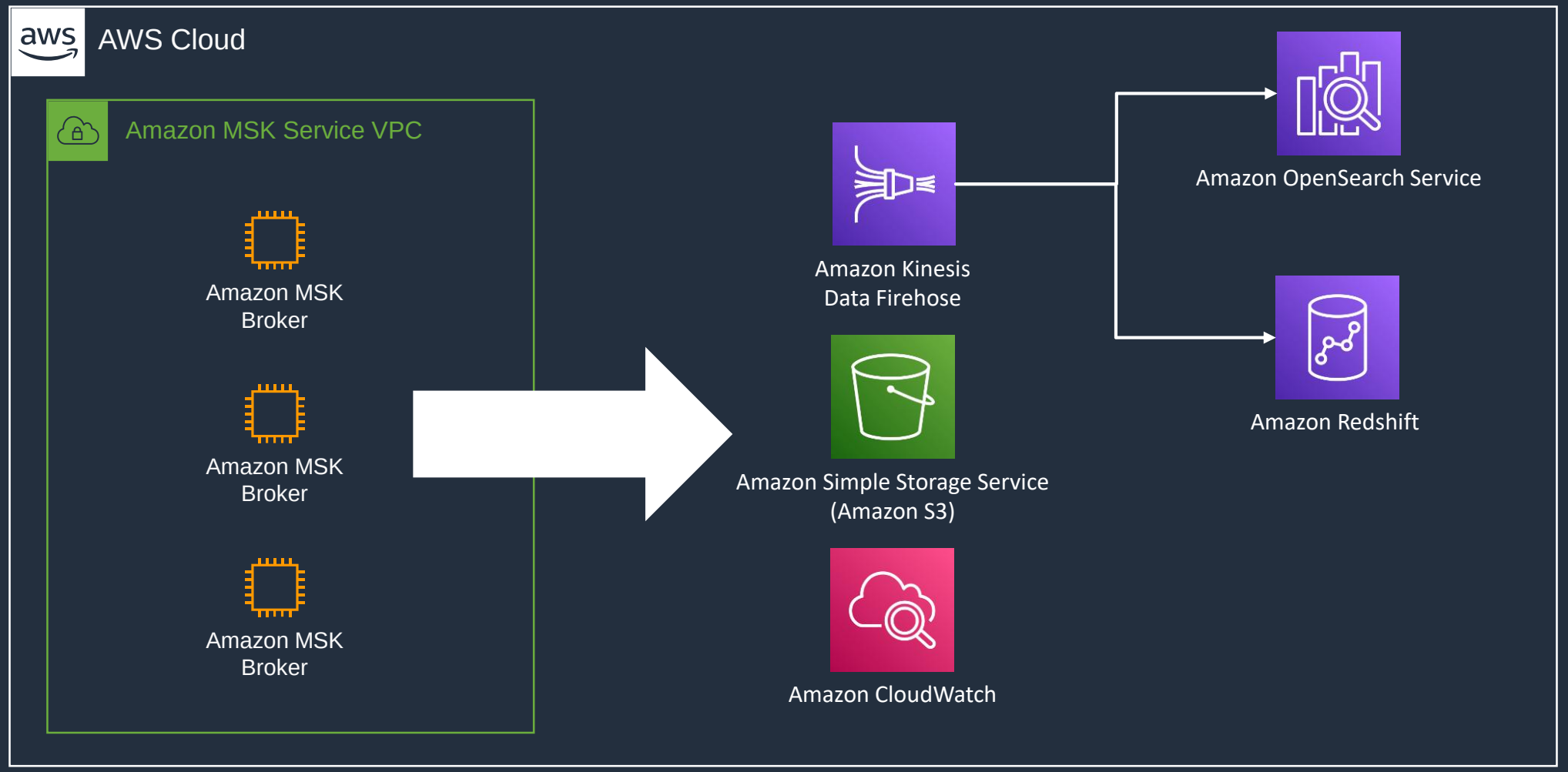
モニタリングのベストプラクティス

- CPU 使用率を 60% 以下で維持するよう、モニタリングと通知の設定を行う。CPU 使用率が高い場合はスケールで対処する
- ログ領域(トピックのデータを保持する領域)が不足した場合にスケール対応ができるように、使用率が 85% 以上になった際にアラートを通知する設定を行う。オートスケールを有効化すると運用負荷が減らせる
- GC 後のヒープ領域の使用率が 60% 以下であることを維持するよう、モニタリングと通知の設定を行う
- コンシューマーアプリケーションの処理遅延発生を検知するために、オフセットのラグを監視する

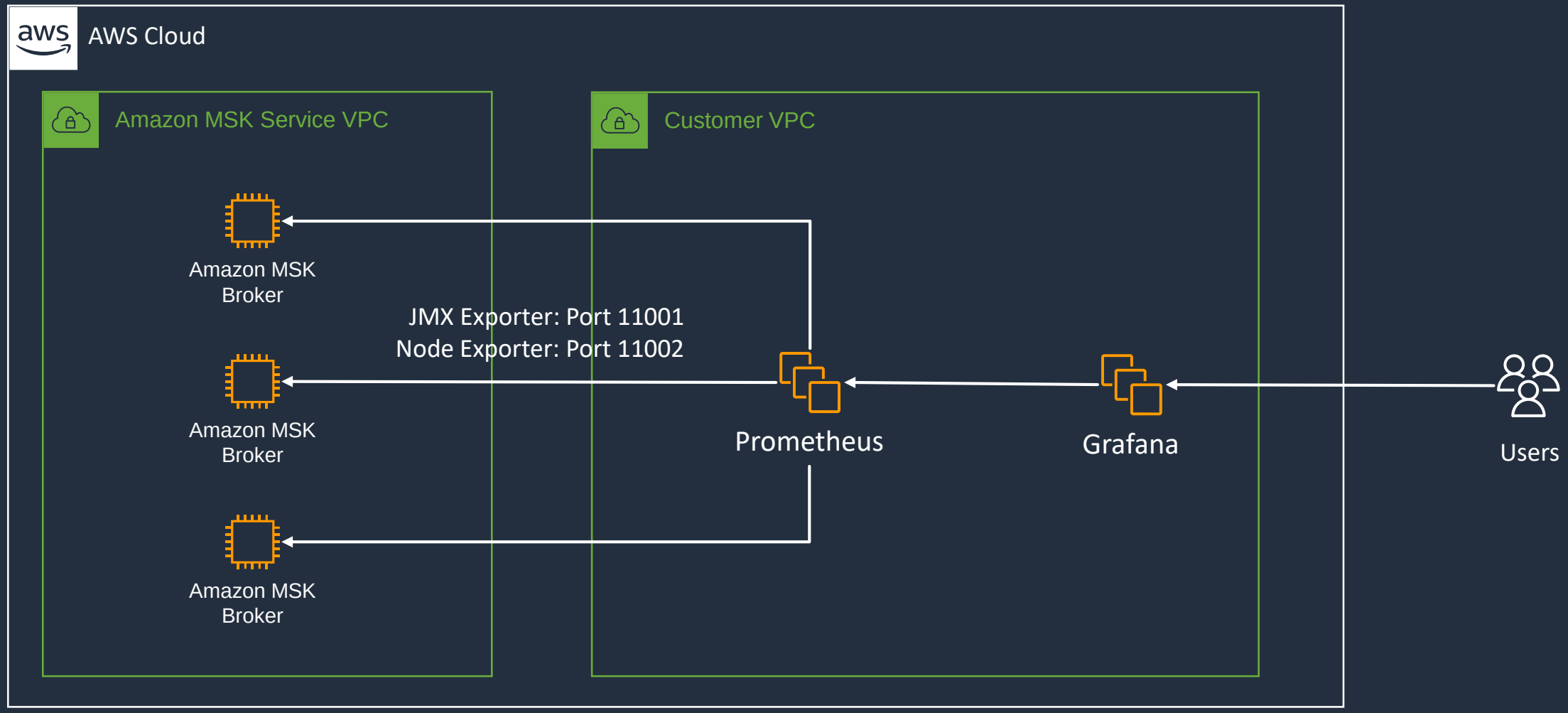
https://docs.aws.amazon.com/msk/latest/developerguide/bestpractices.html#bestpractices-monitor-cpu
https://docs.aws.amazon.com/msk/latest/developerguide/bestpractices.html#bestpractices-monitor-disk-space
https://docs.aws.amazon.com/msk/latest/developerguide/bestpractices.html#bestpractices-monitor-memory
https://docs.aws.amazon.com/msk/latest/developerguide/consumer-lag.html

ブローカーログの送信

ログの配信先に応じて 3 つの配信手段を選択可能



Prometheus を利用したモニタリング



Amazon MSK の特徴

AWS サービスとの統合

データ ソースとしての AWS IoT、データ コンシューマーとしての AWS Lambda、AWS Glue Schema Registry によるスキーマ管理、Amazon Kinesis Data Analytics によるストリーム処理

スケーラビリティ

ブローカーの追加、ブローカーのサイズの変更、ストレージの追加

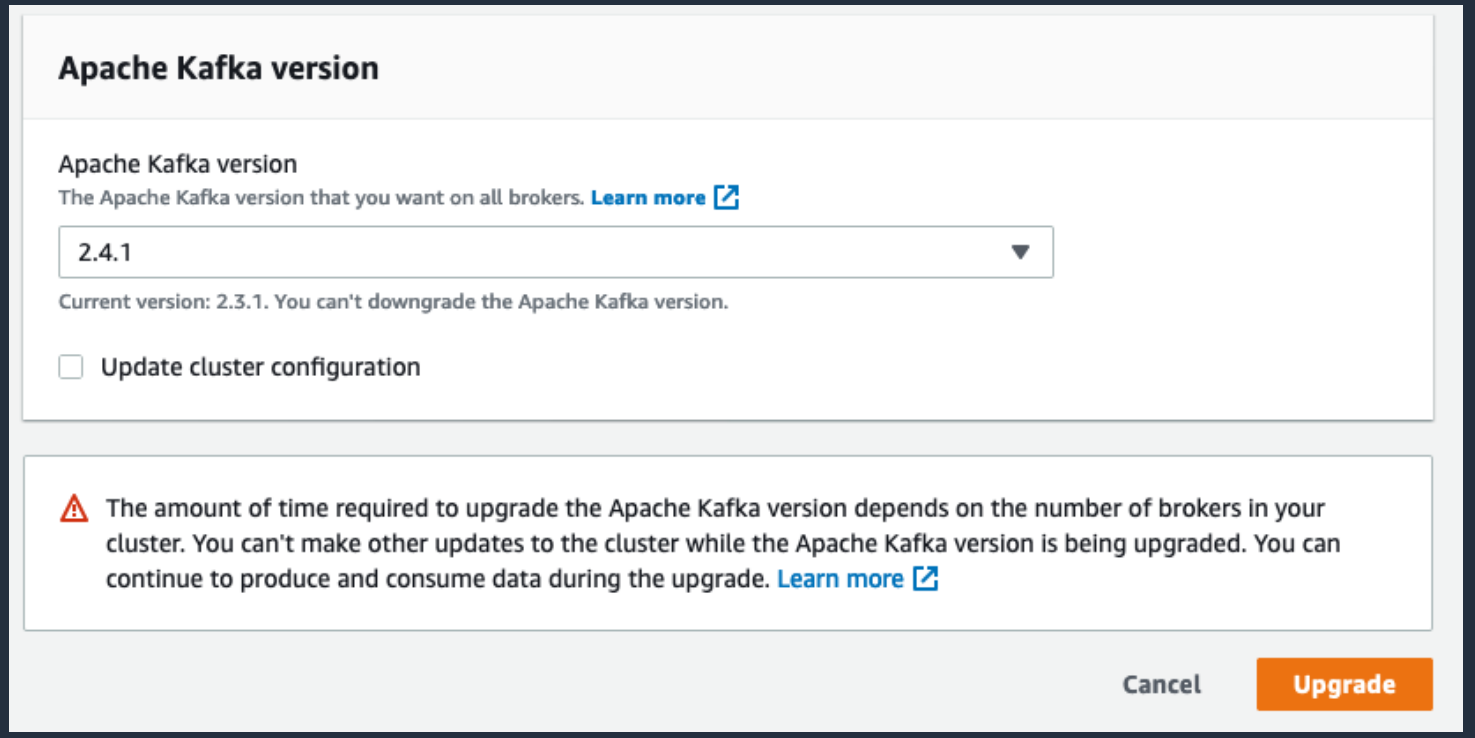
モニタリング・オブザーバビリティ

Amazon CloudWatch を介したログとメトリクスの監視、Open Monitoring for Prometheus による JMX メトリクスの抽出などをサポート

ローリングバージョンアップグレード

クラスターのダウンタイムなしで Kafka バージョンをアップグレードする

Apache Kafka バージョンのアップグレード

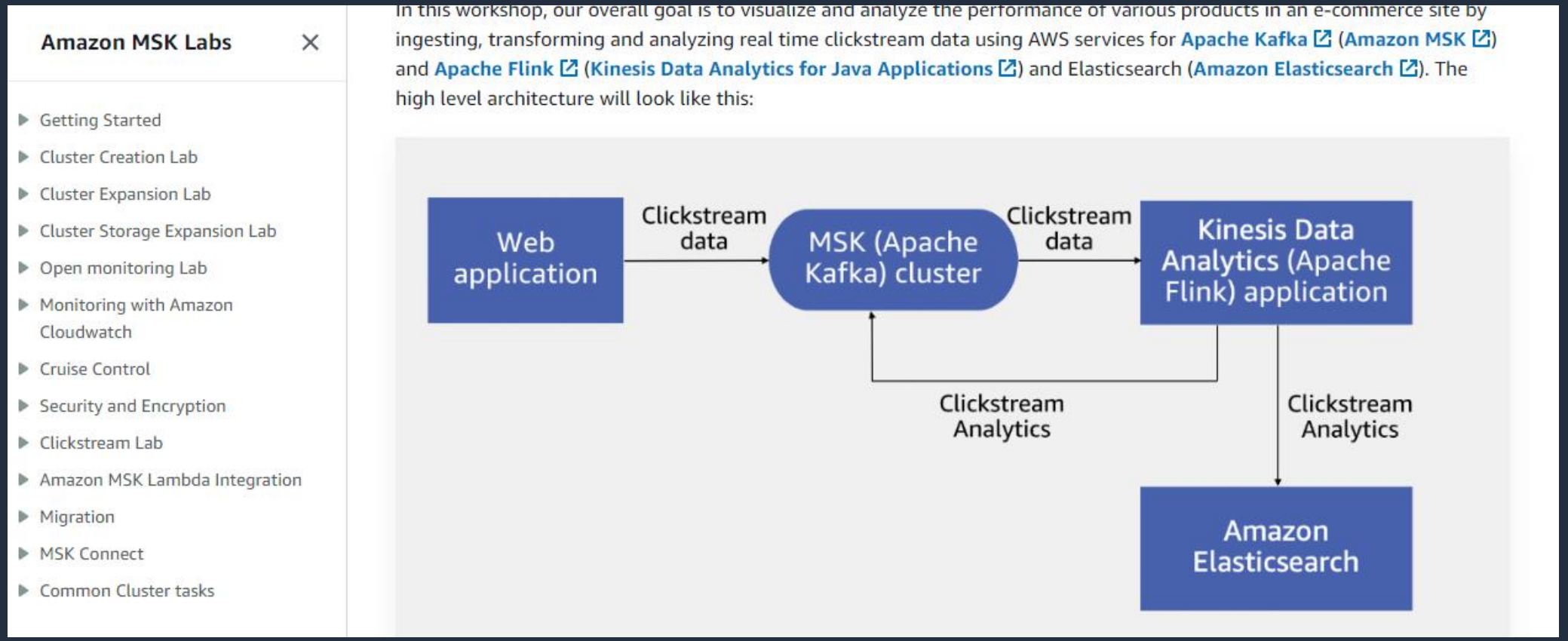


- 現在使用しているバージョン以降のバージョンにアップグレード可能
- アップグレードはバックグラウンドで実行可能
- レプリケーションを設定することでアップグレード中もクラスターを利用可能
- ブローカーのローリングリスタートが発生する

ワークショップ

Amazon MSK labs

クラスターの各種運用管理、移行、サンプルアプリケーションなど、
MSK に関する総合的なワークショップを提供



https://catalog.workshops.aws/msk-labs/en-US

まとめ

まとめ

- Apache Kafka はオープンソースの分散イベント処理プラットフォーム。Pub/Sub のメッセージ配信や、ストリーム処理、データ連携などを実装するための仕組みが提供されている点や、要件に応じた細かなチューニング、周辺のエコシステムが充実している点が特徴

- Amazon MSK は Apache Kafka のフルマネージドサービス。MSK の持つ AWS サービスと統合されたモニタリング機能や、障害からの自動復旧機能を活用することで、運用負荷を削減しながら既存の Kafka アプリケーションを実行することが可能

その他リソース

- サービス概要: https://aws.amazon.com/jp/msk/
- よくある質問: https://aws.amazon.com/jp/msk/faqs/
- 料金: https://aws.amazon.com/jp/msk/pricing/
- ドキュメント: https://docs.aws.amazon.com/msk/latest/developerguide/what-is-msk.html
- 制限事項: https://docs.aws.amazon.com/msk/latest/developerguide/limits.html
- トラブルシューティング: https://docs.aws.amazon.com/msk/latest/developerguide/troubleshooting.html
- ナレッジセンター: https://repost.aws/tags/knowledge-center/TAhybM_7u2QN2GsWOc4aJVDg/amazon-managed-streaming-for-apache-kafka-amazon-msk

本資料に関するお問い合わせ・ご感想

技術的な内容に関しましては、有料のAWSサポート窓口へ
お問い合わせください

https://aws.amazon.com/jp/premiumsupport/

料金面でのお問い合わせに関しましては、カスタマーサポート窓口へ
お問い合わせください（マネジメントコンソールへのログインが必要です）

https://console.aws.amazon.com/support/home#/case/create?issueType=customer-service

具体的な案件に対する構成相談は、後述する個別相談会をご活用ください

ご感想はTwitterへ！ハッシュタグは以下をご利用ください
#awsblackbelt

その他コンテンツのご紹介

ウェビナーなど、AWSのイベントスケジュールをご参照いただけます

https://aws.amazon.com/jp/events/

ハンズオンコンテンツ

https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-hands-on/

AWS 個別相談会

AWSのソリューションアーキテクトと直接会話いただけます

https://pages.awscloud.com/JAPAN-event-SP-Weekly-Sales-Consulting-Seminar-2021-reg-event.html

Thank you!