App Connect Enterprise Kafka node

- 第2.0版（2024.3.31）

日本アイ・ビー・エム システムズ・エンジニアリング

はじめに

- 本書は、IBM App Connect Enterprise(ACE) で提供するKafkaノードについて記載したものです

- 本書の内容はACE V12 Fixpack 11を使用した検証を元に執筆しています。
  - 将来のバージョンアップによってコマンド等が変わる可能性があります。

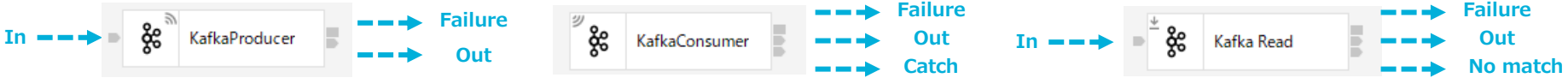
目次

- Kafkaノード概要
  - Kafkaノードとは
  - Kafkaノードのユースケース
  - Kafka Producerノード
  - Kafka Consumerノード
  - Kafka Readノード
  - Kafka ポリシー
  - Kafka カスタム・ヘッダー・プロパティ―
  - LocalEnvironmentツリーの使用
- メッセージの配信パターンとノード構成
  - 負荷分散型配信
  - 同報型配信
  - メッセージ順序保証型配信

Kafkaノード概要

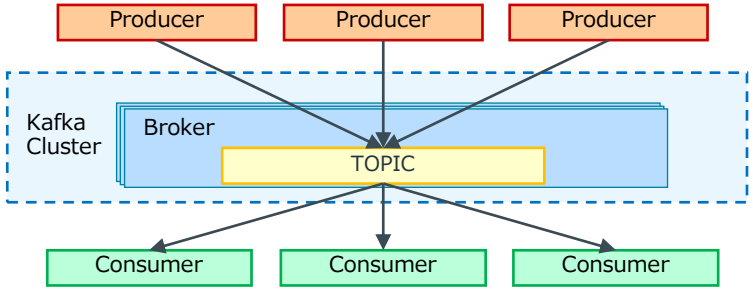
Kafkaノードとは

- Kafkaノードを使用してApache KafkaまたはIBM Event Streamsに接続し、メッセージをKafkaトピックにパブリッシュしたり、 Kafkaトピックにパブリッシュされたメッセージを受け取ることが可能
  - ただし、Kafkaノードで扱えるデータ形式には注意が必要
    - KafkaノードではAvroフォーマットやProtocol Buffersのデータをパース/シリアライズすることは不可
      - ✓ バイナリデータとして取得し、後続のJava Computeなどでデシリアライズするなどの作りこみが必要
- Apache Kafkaサーバーにクライアントとして接続し、Kafkaメッセージを処理
- V12.0.11.0で利用されているKafkaのライブラリ・バージョンは3.6.0
- Kafkaノードとして以下のノードを提供
  - Kafka Producerノード
    - Kafkaサーバー上のトピックに対し、メッセージフロー内で生成したメッセージをパブリッシュする
  - Kafka Consumerノード
    - Kafkaサーバー上のトピックに対しサブスクライブし、そのトピックにパブリッシュされたメッセージをメッセージフローの入力として受信する
  - Kafka Readノード
    - トピックパーティションのオフセットを指定することで、パブリッシュされたKafkaサーバー上のトピックから特定のされたメッセージを読み取る



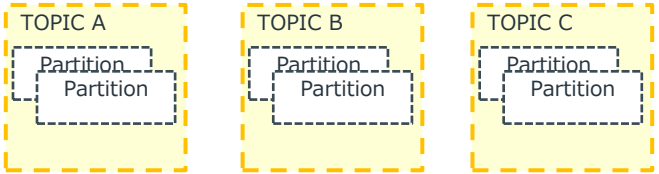
補足:Apache Kafka概要

- オープンソースの分散メッセージング・システム
  - LinkedInにより開発され、2011年にオープンソースとして公開
  - 大容量のログを高スループット/低レイテンシに収集/配信することを目的に開発された
- Apache Kafkaの４つの特徴
  - Fast：　高スループット/低レイテンシのメッセージング処理が可能
  - Scalable：　Kafkaクラスター構成によりスケールアウトが可能
  - Durable：　データの損失を防ぐため、メッセージを永続化しKafkaクラスター内で複製
  - Distributed by Design：　複数ブローカー構成により、Kafkaクラスターは高い耐障害性を持つ
- データの送受信方式
  - ProducerとConsumerがトピックを介してM対Nで通信するパブリッシュ/サブスクライブ・モデル
  - 負荷分散/耐障害性のためKafkaではトピックに送信されたデータを複数サーバーで管理

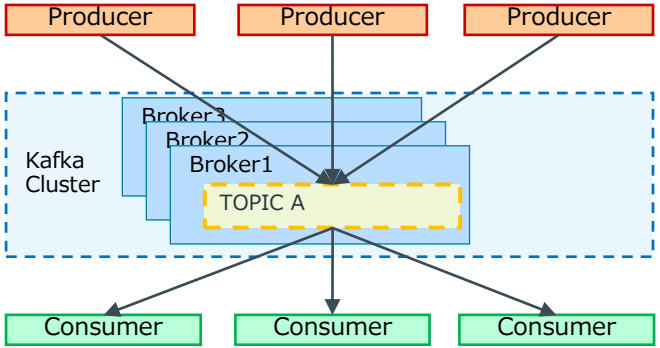


補足: Kafkaの用語解説

- トピック
  - メッセージを送受信する対象となるオブジェクト
    - ユーザーが任意で作成可能
    - それぞれのトピックは独立している
      - ✓ JMSやMQTTのトピックと異なり、階層構造になっていない
    - トピックは1つ以上のパーティションで構成される

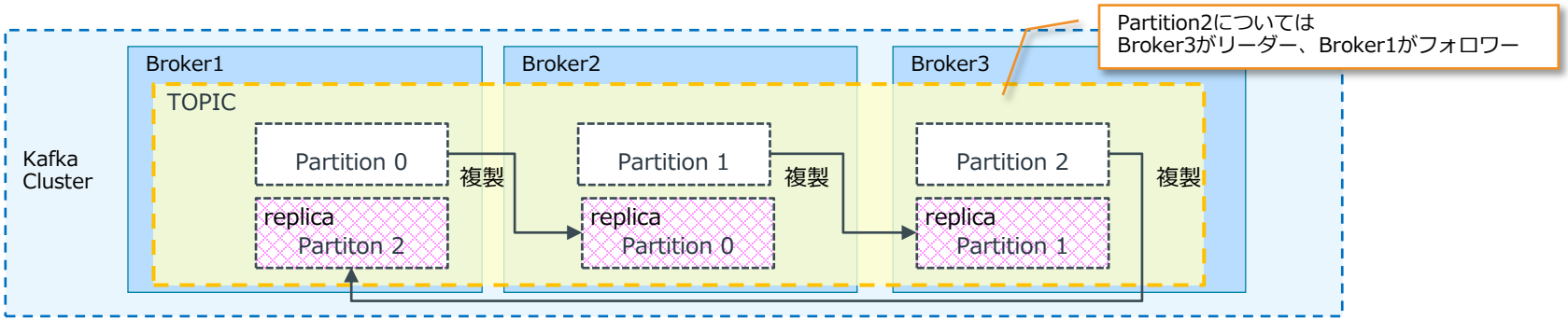


- プロデューサーとコンシューマー
  - メッセージを送受信するユーザーアプリケーション
    - プロデューサー: トピックに対してメッセージを送信する
    - コンシューマー: トピックをサブスクライブし、メッセージを受信する



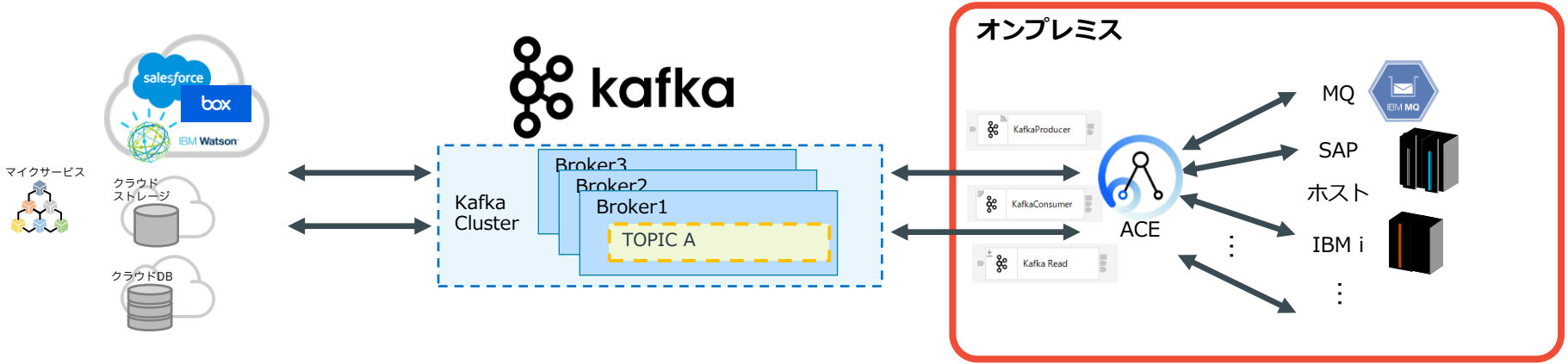
補足: Kafkaの用語解説

- パーティション
  - メッセージが保管される領域
  - メッセージには送信順にオフセットと呼ばれるIDが付与される
    - オフセットはパーティション内でユニークとなる
- レプリカ
  - トピックの各パーティションは他のブローカーにレプリカを作成することが可能
    - IBM Message Hubでは1パーティションに対して2つのレプリカが作成される
  - 全てのレプリカの内、1つがリーダー、その他のレプリカはフォロワーと呼ばれ、リーダーがProduce/Consumeのリクエストを受け付ける
    - 書き込まれたメッセージは同期/非同期でフォロワーのパーティションに転送される
      - ✓ 同期/非同期はProducerのackプロパティにて設定する
  - リーダーがダウンした場合、いずれかのフォロワーが自動的にリーダーに昇格する

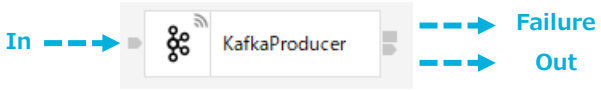


Kafkaノードのユースケース

- オンプレミスとクラウドの連携
  - メインフレームをはじめとした基幹システムをKafkaを通じてクラウド上のデータ・ストアに伝搬
    - クラウド上のアプリケーションは鮮度の高いデータをローカルのストレージで利用できるようになる
  - クラウドやSaaS上で発生したイベントをKafkaで収集し、基幹システムに伝搬
    - 基幹システムのデータベースとのデータ同期など



Kafka Producerノード



- Kafka Producerノード
  - Kafkaクラスター上のトピックに対し、メッセージフロー内で生成したメッセージをパブリッシュする
  - Producerノードでのメッセージのパブリッシュは非トランザクション
    - Producerノードを通過後にフローがロールバックしてもパブリッシュされたメッセージはロールバックしない
  - Acksプロパティを使用して、Kafkaサーバーでメッセージが正常に受信されたことの確認を待機することで、同期的な処理が可能
- Kafka Producerノードの主なプロパティ

タブ	設定項目	必須	備考
Basic	Topic name	○	メッセージの配信先トピックを指定する。
	Bootstrap servers	○	接続先Kafkaクラスターを指定する。カンマ区切りで複数指定することも可能。
	Client ID		クライアントIDを指定する。クライアントIDを論理アプリケーション名としてKafkaログに含めることで、ユーザーがIPとポート以外の情報でリクエスト元を追跡するために使用する。
	Add IIB suffix to client ID		チェックを入れると、クライアントIDに統合サーバー名と統合ノード名を接尾語として付加する。
	Acks	○	メッセージが正常にパブリッシュされたと判断するためにKafkaから返されるACKを待機するかを指定する。 0: ACKを待たない　（デフォルト） 1: Kafkaからの1つのACKで正常にパブリッシュされたと判断する All: トピックの全てのレプリカからのACKを待つ
	Timeout (sec)	○	リクエストのタイムアウト値を指定する。　（デフォルト60秒）
Advanced	Properties file		Kafkaクライアント・プロパティを設定するkafka.propertiesファイルの絶対パス名を指定する。 プロパティ・ファイルを使用することで、KafkaProducerノードのプロパティで設定できないKafka Producerのプロパティを設定可能。

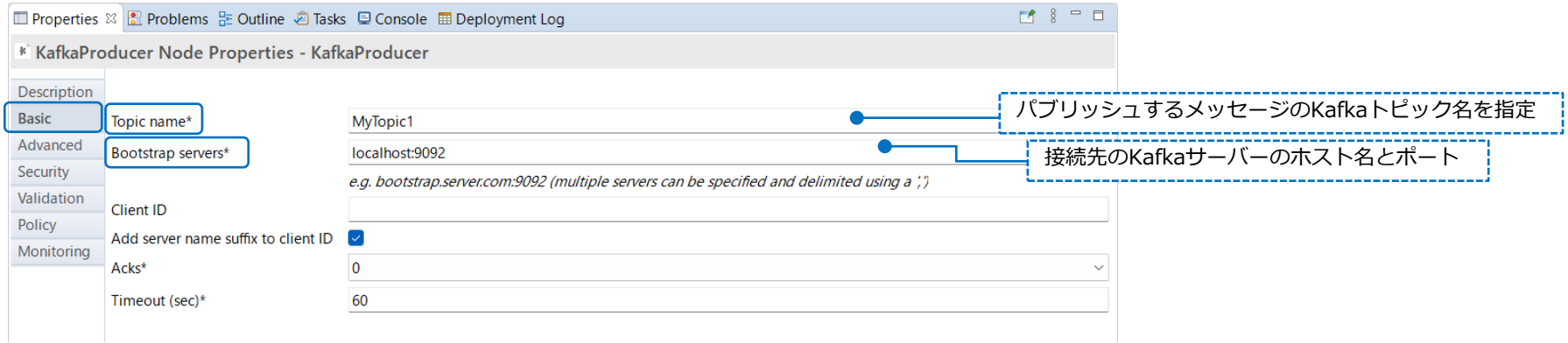
Kafka Producerノード

- Kafka Producerノードの主なプロパティ

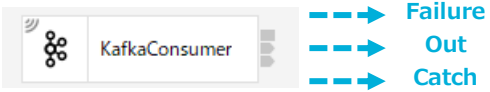
タブ	設定項目	必須	備考
Security	Security identity		Vaultに登録されたユーザーIDとパスワードを取得するための資格情報名を指定する。
	Security protocol	○	使用するセキュリティプロトコルを以下から選択する。 • PLAINTEXT （デフォルト） • SSL • SASL_PLAINTEXT • SASL_SSL IBM Event Streamsに接続する場合はSASL_SSLを指定する必要がある。
	SSL Protocol	○	Security protocolでSSLまたはSASL_SSLを指定する場合、使用するSSLプロトコルを指定する。
Policy	Policy		使用するKafkaポリシー名を指定する。

Kafka Producerノード

- 設定イメージ



Kafka Consumerノード



- Kafka Consumerノード
  - Kafkaクラスター上のトピックに対しサブスクライブし、そのトピックにパブリッシュされたメッセージをメッセージフローの入力として受信する
  - 各Kafka Consumerノードは単一のトピックからメッセージをサブスクライブする
    - トピックが複数のパーティションを持つように定義されている場合は、任意のパーティションからメッセージを受信
  - Consumerノードでのメッセージのサブスクライブは非トランザクション
    - フローが入力ノードにロールバックする場合を考慮し、必要に応じてConsumerノードのCatchターミナルを使用し、エラー時にメッセージを再処理または対比するような実装を検討する
  - Kafkaサーバでオフセットのコミットが完了するまで、フローによるメッセージの処理を待機することも可能
    - ノードのプロパティ「Wait for message offset commit to complete」で制御
    - At most onceレベルの送達保証を実現
  - 追加インスタンス設定時のメッセージの処理
    - 追加インスタンスを構成した場合でも、Kafka側から見えるコンシューマーインスタンスは1つ
    - コンシューマーインスタンスが追加インスタンスにメッセージを配布する
    - メッセージの順序性は保証されないため、順序保証が必要な場合は注意が必要
  - Group ID設定時のメッセージの処理
    - 同一のGroup IDを持つ複数のKafka Consumerノード間で、トピックにパブリッシュされたメッセージの並列処理が可能

Kafka Consumerノード

■ Kafka Consumerノードの主なプロパティ

タブ	設定項目	必須	備考
Basic	Topic name	○	メッセージを受け取るトピックを指定する。
	Bootstrap servers	○	接続先Kafkaクラスターを指定する。
	Consumer group ID	○	コンシューマーグループIDを指定する。
	Default message offset	○	メッセージを受け取るオフセットのデフォルト位置を以下の2つから選択する。 • Earliest • Latest （デフォルト）
	Commit message offset in Kafka		有効にすると、Kafka上に受け取ったメッセージのオフセット位置を保存する。 このパラメーターが有効、かつ保存されたオフセットのメッセージが有効な場合、そのオフセット番号からメッセージを受け取る。
	Wait for message offset commit to complete		有効にすると、KafkaConsumer ノードは、メッセージ・オフセットが Kafka にコミットされるまで待機してから、メッセージ・フローにメッセージを配信する。（at-most-once レベルの送達保証）
	Client ID		クライアントIDを指定する。クライアントIDを論理アプリケーション名としてKafkaログに含めることで、ユーザーがIPとポート以外の情報でリクエスト元を追跡するために使用する。
	Add server name suffix to client ID		チェックを入れると、クライアントIDに統合サーバー名と統合ノード名を接尾語として付加する。
Advanced	Connection timeout (sec)	○	接続タイムアウトを指定する。セッションタイムアウトより大きくする。（デフォルト15秒）
	Session timeout (sec)	○	セッションタイムアウトを指定する。Kafkaがこの値の間にConsumerからのハートビートを受信できなかった場合、このクライアントをConsumer Groupから削除する。接続が切れたクライアントが受け持っていたパーティションは、別の同一グループ内のConsumerに自動的に割り当てられる。 （デフォルト10秒）
	Receive batch size	○	単一のバッチでKafkaから受け取るレコードの最大数を指定する。

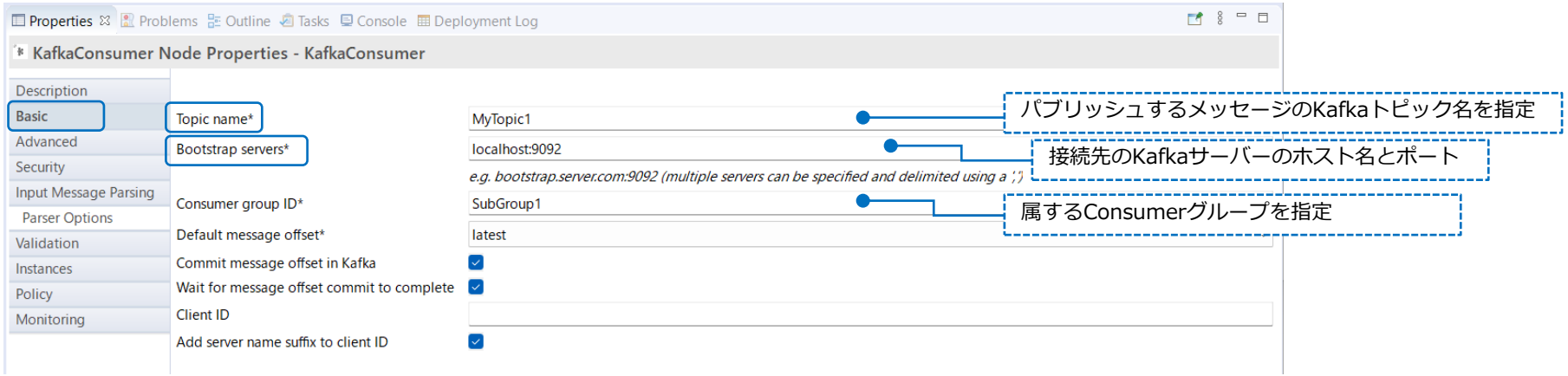
Kafka Consumerノード

- Kafka Consumerノードの主なプロパティ(続き)

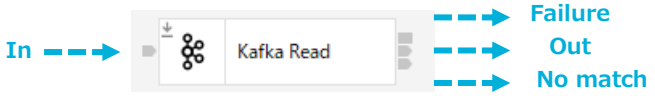
タブ	設定項目	必須	備考
Advanced	Properties file		Kafkaクライアント・プロパティを設定するkafka.propertiesファイルの絶対パス名を指定する。 プロパティ・ファイルを使用することで、KafkaConsumerノードのプロパティで設定できないKafka Consumerのプロパティを設定可能。
Security	Security identity		Vaultに登録されたユーザーIDとパスワードを取得するための資格情報名を指定する。
	Security protocol	○	使用するセキュリティプロトコルを以下から選択する。 • PLAINTEXT （デフォルト） • SSL • SASL_PLAINTEXT • SASL_SSL IBM Event Streamsに接続する場合はSASL_SSLを指定する必要がある。
	SSL Protocol	○	Security protocolでSSLまたはSASL_SSLを指定する場合、使用するSSLプロトコルを指定する。
Policy	Policy		使用するKafkaポリシー名を指定する。
Instances	Additional Instances pool		追加インスタンスの取得元のプールを指定する。 • Use Pool Associated with Message Flow （デフォルト） メッセージフローの値から追加インスタンスを取得する • Use Pool Associated with Node Additional Instancesの値を元に追加インスタンスを割り当てる
	Additional Instances		ノードが開始できる追加インスタンスの数。

Kafka Consumerノード

- 設定イメージ



Kafka Readノード



- Kafka Readノード
  - トピック内のメッセージのオフセット位置を指定することで、Kafkaトピックにパブリッシュされたメッセージから特定のメッセージを読み取ることが可能
    - Kafka Consumerノードの処理で失敗した場合に、Kafka Readノードを使用して、エラーのあったメッセージを再処理することも可能
  - 指定したオフセットのメッセージを取得できない場合、No matchターミナルに処理が流れる
    - Kafka Readノードの「Action when message unavailable」プロパティを指定することで、代わりにパーティション内の最も古いメッセージ、または最新のメッセージのいずれかを受信することも可能

Kafka Readノード

- Kafka Readノードの主なプロパティ

タブ	設定項目	必須	備考
Basic	Topic name	○	メッセージを受け取るトピックを指定する。
	Bootstrap servers	○	接続先Kafkaクラスターを指定する。
	Partition number	○	使用するKafkaのパーティション番号を指定する。
	Message offset	○	取得するメッセージのオフセット位置を指定する。
	Action when message unavailable	○	指定したオフセットのメッセージが使用不可の場合に実行するアクションを指定する。 • no match: No matchターミナルに伝搬 （デフォルト） • earliest: パーティション内の最初のオフセットを持つメッセージを取得 • latest: パーティション内の最新のオフセットを持つメッセージを取得
	Client ID		クライアントIDを指定する。クライアントIDを論理アプリケーション名としてKafkaログに含めることで、ユーザーがIPとポート以外の情報でリクエスト元を追跡するために使用する。
	Add server name suffix to client ID		チェックを入れると、クライアントIDに統合サーバー名と統合ノード名を接尾語として付加する。
	Timeout (sec)	○	リクエストのタイムアウト値を指定する。 （デフォルト5秒）
Advanced	Properties file		Kafkaクライアント・プロパティを設定するkafka.propertiesファイルの絶対パス名を指定する。 プロパティ・ファイルを使用することで、KafkaConsumerノードのプロパティで設定できないKafka Consumerのプロパティを設定可能。

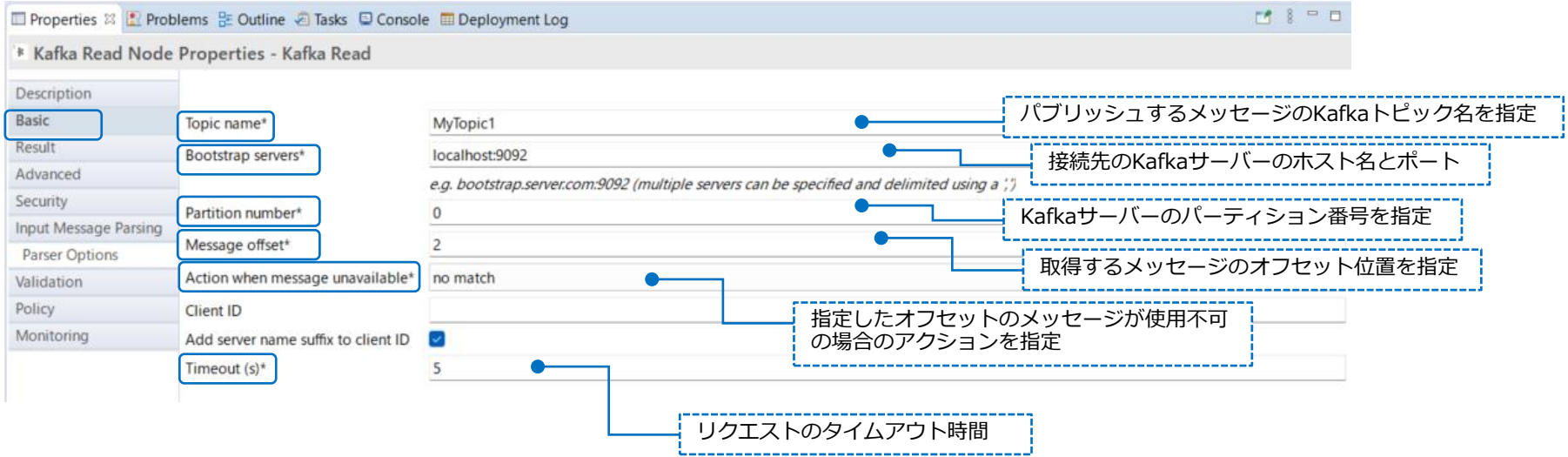
Kafka Readノード

- Kafka Readノードの主なプロパティ(続き)

タブ	設定項目	必須	備考
Security	Security identity		Vaultに登録されたユーザーIDとパスワードを取得するための資格情報名を指定する。
	Security protocol	○	使用するセキュリティプロトコルを以下から選択する。 • PLAINTEXT （デフォルト） • SSL • SASL_PLAINTEXT • SASL_SSL IBM Event Streamsに接続する場合はSASL_SSLを指定する必要がある。
	SSL Protocol	○	Security protocolでSSLまたはSASL_SSLを指定する場合、使用するSSLプロトコルを指定する。
Policy	Policy		使用するKafkaポリシー名を指定する。

Kafka Readノード

■ 設定イメージ



Kafka ポリシー

- Kafka ポリシーを使用し、Kafkaノードの設定を動的に変更可能（11.0.0.7）
- Kafka ポリシーの主なプロパティ

プロパティ名	説明
bootstrapServers	Kafkaノードの「Bootstrap servers」プロパティをオーバーライド
securityProtocol	Kafkaノードの「Security protocol」プロパティをオーバーライド
saslMechanism	Kafkaサーバーに接続するときに使用されるSASLメカニズムを設定
sslProtocol	Kafkaノードの「SSL Protocol」プロパティをオーバーライド
securityIdentity	Kafkaノードの「Security identity」プロパティをオーバーライド
saslConfig	SASL構成を指定
sslKeystoreLocation	SSL 接続の使用時に使用される鍵ストアのロケーションへの完全修飾パスを指定
sslKeystoreType	鍵ストアのタイプを指定
sslKeystoreSecurityIdentity	鍵ストアにアクセスするために使用されるセキュリティー ID を指定
sslKeySecurityIdentity	鍵ストア内の鍵にアクセスするために使用されるセキュリティー ID を指定
sslTruststoreLocation	SSL 接続の使用時に使用されるトラストストアのロケーションへの完全修飾パスを指定
sslTruststoreType	トラストストアのタイプを指定
sslTruststoreSecurityIdentity	トラストストアにアクセスするために使用されるセキュリティー ID を指定
sslEnableCertificateHostnameChecking	Kafka サーバーのホスト名を Kafka サーバーの証明書内のホスト名と照合するかどうかを指定

Kafka ポリシー

Policy

Set the attributes for a Policy

The property "Bootstrap servers" requires a value to be specified.

Name	KafkaPolicy
Type	Kafka
Template	Kafka

Property	Value
Bootstrap servers*	
Security protocol*	PLAINTEXT
SASL Mechanism	
SSL protocol*	TLSv1.2
Security identity (DSN)	
SASL config	
SSL keystore location	
SSL keystore type	JKS
SSL keystore security identity	
SSL key security identity	
SSL truststore location	
SSL truststore type	JKS
SSL truststore security identity	
Enable SSL certificate hostname checking	true
Kerberos service name	

Kafkaカスタム・ヘッダー・プロパティー

- Kafkaカスタム・ヘッダー・プロパティを使用し、Kafkaメッセージにメタデータを付与することが可能
  - データのフォーマット情報（スキーマ名）をカスタム・ヘッダー・プロパティにもたせるなど
  - name:valueのペア形式でセット
  - メッセージのパブリッシュ時にメッセージにセットされ、メッセージのサブスクライブ時に取得される
  - LocalEnvironmentのKafkaHeaderフォルダーに保持
    - KafkaProducerノード：LocalEnvironment.Destination.Kafka.Output.KafkaHeader
    - KafkaConsumerノード：LocalEnvironment.Kafka.Input.KafkaHeader
    - KafkaReadノード：LocalEnvironment.Kafka.Read.KafkaHeader

Kafkaカスタム・ヘッダー・プロパティー

- KafkaProducerノードでKafkaカスタム・ヘッダー・プロパティをセットする例
  - LocalEnvironment.Destination.Kafka.Output.KafkaHeaderの下に任意のプロパティをセット

```
# ‘Occupation’と’Country’というKafkaカスタム・ヘッダー・プロパティーをセットする
SET OutputLocalEnvironment.Destination.Kafka.Output.KafkaHeader.Occupation = 'Student';
SET OutputLocalEnvironment.Destination.Kafka.Output.KafkaHeader.Country = 'USA';
```

- KafkaConsumerノードで取得すると
  - LocalEnvironment.Kafka.Input.KafkaHeaderの下に格納される

```
(0x01000000:Name):Kafka = (
  (0x01000000:Name):Input = (
    (0x01000000:Name     ):KafkaHeader = (
      (0x03000000:NameValue):Country    = 'USA' (CHARACTER)
      (0x03000000:NameValue):Occupation = 'Student' (CHARACTER)
    )
    (0x03000000:NameValue):partition   = '0' (CHARACTER)
    (0x03000000:NameValue):topicName   = 'MyTopic1' (CHARACTER)
    (0x03000000:NameValue):offset      = '6' (CHARACTER)
  )
)
```

- KafkaReadノードで取得すると
  - LocalEnvironment.Kafka.Read.KafkaHeaderの下に格納される

```
(0x01000000:Name):Kafka       = (
  (0x01000000:Name):Read = (
    (0x01000000:Name     ):KafkaHeader = (
      (0x03000000:NameValue):Country    = 'USA' (CHARACTER)
      (0x03000000:NameValue):Occupation = 'Student' (CHARACTER)
    )
    (0x03000000:NameValue):partition   = '0' (CHARACTER)
    (0x03000000:NameValue):topicName   = 'MyTopic1' (CHARACTER)
    (0x03000000:NameValue):offset      = '6' (CHARACTER)
  )
)
```

参考: Apache Kafkaに接続する際の構成

- Apache Kafkaではブローカー同士や、クライアント(Producer/Consumer)、ツールからの接続認証にSSLまたはSASLを使用可能
  - Apache KafkaではSASLの方式として以下をサポート
    - SASL/GSSAPI(Kerberos) 0.9.0.0以降
    - SASL/PLAIN 0.10.0.0以降
    - SASL/SCRAM-SHA-256, SASL/SCRAM-SHA-512 0.10.2.0以降
    - SASL/OAUTHBEARER 2.0以降
  - セキュリティはオプションのため、平文でのやり取りも可能
  - Apache KafkaでのSASL, SSLの構成は公式のドキュメントを参照
    - https://kafka.apache.org/documentation/#security
- ACE側の設定
  - mqsisetdbparmsコマンド、またはmqsicredentialsコマンドでユーザーID/パスワードの資格情報を登録する
    - 参考：https://www.ibm.com/docs/ja/app-connect/12.0?topic=messages-configuring-security-credentials-connecting-kafka
  - Kafkaノードのプロパティ設定
    - Securityタブ
      - ✓ Security protocol: SASL_PLAITEXT、またはSASL_SSLを選択
      - ✓ SLL protocol: Security protocolでSSLまたはSASL_SSLを選択した場合にはSSLのプロトコルを選択
      - ✓ Security identity : mqsisetdbparmsコマンド、またはmqsicredentialsコマンドで登録した資格情報名を指定

参考: Apache Kafkaに接続する際の構成

- SASL/SCRAMを使用したKafkaクラスターへの接続認証
  - Salted Challenge Response Authentication Mechanism (SCRAM)を使用したKafkaクラスターでの接続認証をサポート（11.0.0.9）
  - Kafkaポリシーを使用して構成
- ACE側の設定
  - mqsisetdbparmsコマンド、またはmqsicredentialsコマンドでユーザーID/パスワードの資格情報を登録する
    - 参考：https://www.ibm.com/docs/ja/app-connect/12.0?topic=messages-authenticating-connections-kafka-cluster-by-using-saslscram
  - Kafkaポリシーの作成
    - プロパティ設定
      - ✓ Security protocol: SASL_SSLを選択
      - ✓ Security mechanism： Kafkaクラスターに応じてSCRAM-SHA-256、またはSCRAM-SHA-512を設定
      - ✓ Security identity： mqsisetdbparmsコマンド、またはmqsicredentialsコマンドで登録した資格情報名を指定
      - ✓ SASL config： org.apache.kafka.common.security.scram.ScramLoginModule required;に設定
      - ✓ SSL truststore location: トラストストアの場所を指定
      - ✓ SSL truststore type: トラストストアのタイプを指定
      - ✓ SSL truststore security identity: トラストストアへのアクセスに使用するセキュリティIDを指定
  - Kafkaノードのプロパティ設定
    - Policyタブ
      - ✓ Policy: 作成したポリシーを指定

LocalEnvironmentツリーの使用

- Kafka(Message Hub)呼び出しに関するLocalEnvironment変数
  - LocalEnvrionment.Destination.Kafka.Output
    - KafkaProducerノードのプロパティを上書き指定することが可能
  - LocalEnvironment.WrittenDestination.Kafka
    - 出力メッセージの伝搬時にKafkaProducerノードによって書き込まれる
  - LocalEnvironment.Kafka.Input
    - 出力メッセージの伝搬時にKafkaConsumerノードによって書き込まれる
  - LocalEnvironment.Destination.Kafka.Read
    - KafkaReadノードのプロパティを上書き指定することが可能
  - LocalEnvironment.Kafka.Read
    - 出力メッセージの伝搬時にKafkaReadノードによって書き込まれる

  - LocalEnvironment.Destination.Kafka.Output：KafkaProducerノードのプロパティを上書きする

変数	上書きするノードプロパティ	備考
topicName	Basicタブ Topic Name	メッセージの配信先トピックを指定
key	N/A	メッセージに関連付けるキー値を指定。同一のキー値が指定されたメッセージは、同一トピック内の同じパーティションに配信される ※ノードプロパティでの設定は不可。

LocalEnvironmentツリーの使用

- LocalEnvironment.WrittenDestination.Kafka: 出力メッセージの伝搬時にKafkaProducerノードによって書き込まれる

変数	タイプ	備考
partition	String	メッセージが格納されたパーティション
topicName	String	メッセージがパブリッシュされたトピック名
offset	Integer	メッセージのパーティション内でのオフセット番号
checksum	Integer	メッセージのチェックサム
key	String	LocalEnvrionment.Destination.Kafka.Output.keyで指定されたキー値(指定した場合)

- LocalEnvironment.Kafka.Input: 出力メッセージの伝搬時にKafkaConsumerノードによって書き込まれる

変数	タイプ	備考
partition	String	メッセージが格納されていたパーティション
topicName	String	メッセージがパブリッシュされたトピック名
offset	Integer	メッセージのパーティション内でのオフセット番号
checksum	Integer	メッセージのチェックサム
key	String	受信したメッセージに関連付けられたキー(存在する場合)

LocalEnvironmentツリーの使用

- LocalEnvironment.Destination.Kafka.Read：KafkaReadノードのプロパティを上書きする

変数	タイプ	備考
partitionNumber	String	メッセージが格納されたパーティション
topicName	String	メッセージがパブリッシュされたトピック名
offset	Integer	メッセージのパーティション内でのオフセット番号
timeoutInterval	Integer	Kafkaトピックからメッセージを読み取る際のタイムアウト時間

- LocalEnvironment.Kafka.Read :出力メッセージの伝搬時にKafkaReadノードによって書き込まれる

変数	タイプ	備考
partitionNumber	String	メッセージが格納されていたパーティション
topicName	String	メッセージがパブリッシュされたトピック名
offset	Integer	受信したメッセージのパーティション内でのオフセット番号
checksum	Integer	メッセージのチェックサム
key	String	受信したメッセージに関連付けられたキー(存在する場合)

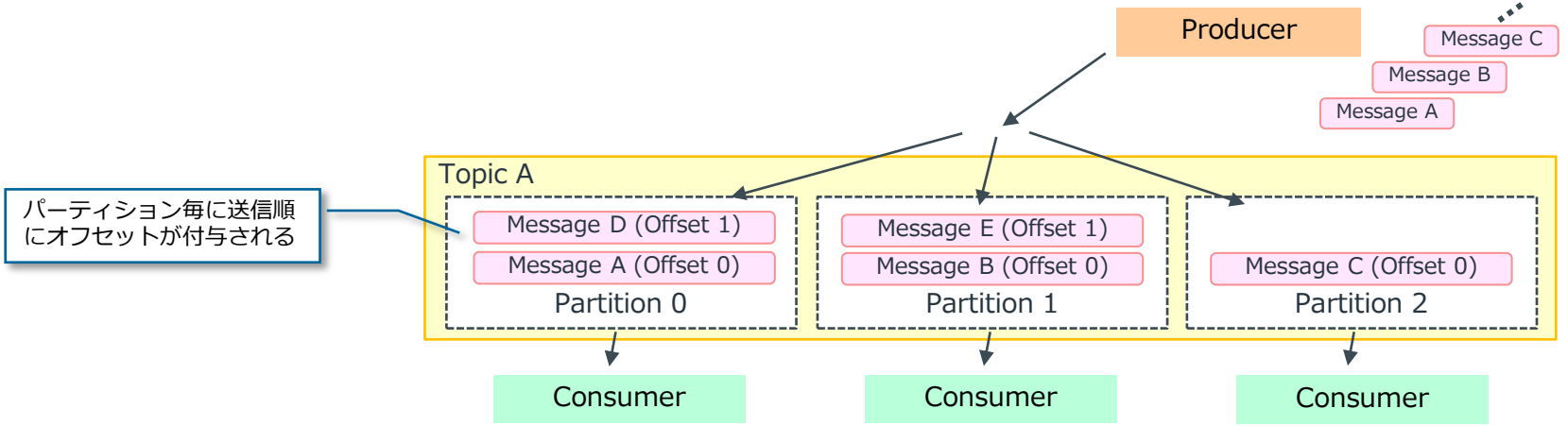
メッセージの配信パターンとノード構成

メッセージの配信パターン

- パーティションやプロデューサー、コンシューマーの構成によって、メッセージの配信パターン がある
  - パターン1:　　負荷分散型の配信
  - パターン2:　　同報配信
  - パターン3:　　メッセージの順序性を保証した配信

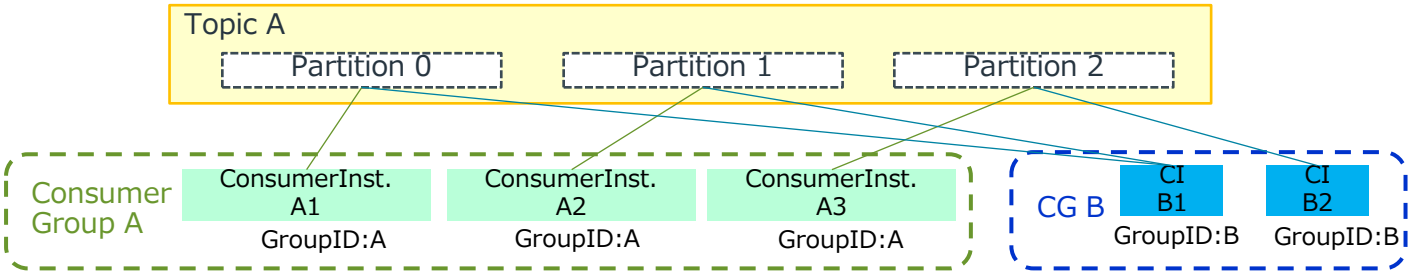
補足: Kafkaの用語解説

- 複数パーティションでの構成
  - 1つのトピックは1つ以上のパーティションで構成される
    - トピック作成時にパーティション数を指定
  - プロデューサーから送信されたメッセージが複数のパーティションに分散して格納される
    - プロデューサーはパーティションを明示的に指定して送信することも、指定せずラウンドロビンで振り分けることも可
  - コンシューマーは特定のパーティションからメッセージを受信
  - パーティション毎にコンシューマーを用意することで分散処理を実現



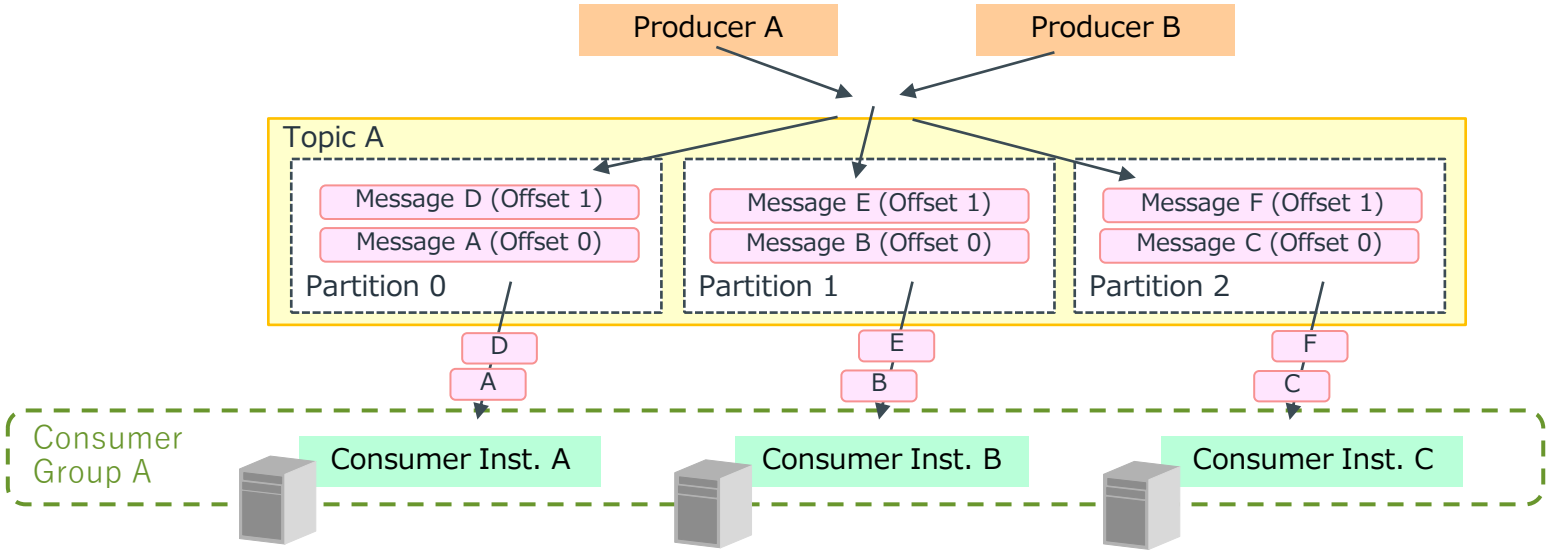
補足: Kafkaの用語解説

- コンシューマーグループ
  - 接続時に指定したグループIDに従ってコンシューマー・グループを形成する
    - 同一グループIDを指定したコンシューマー・インスタンスは同じグループに属する
  - グループ内ではトピックを構成する全てのパーティションがそれぞれのインスタンスにアサインされる
    - 1インスタンスが複数のパーティションを受け持つことはあるが、複数のインスタンスが1つのパーティションを受け持つことはない
    - 自身にアサインされたパーティションからしかメッセージを受け取らないため、トピックに配信されたメッセージを分散して処理する事ができる(負荷分散型)
    - 異なるグループ間のインスタンス同士は同じパーティションから同じメッセージを受信することになる(同報配信型)



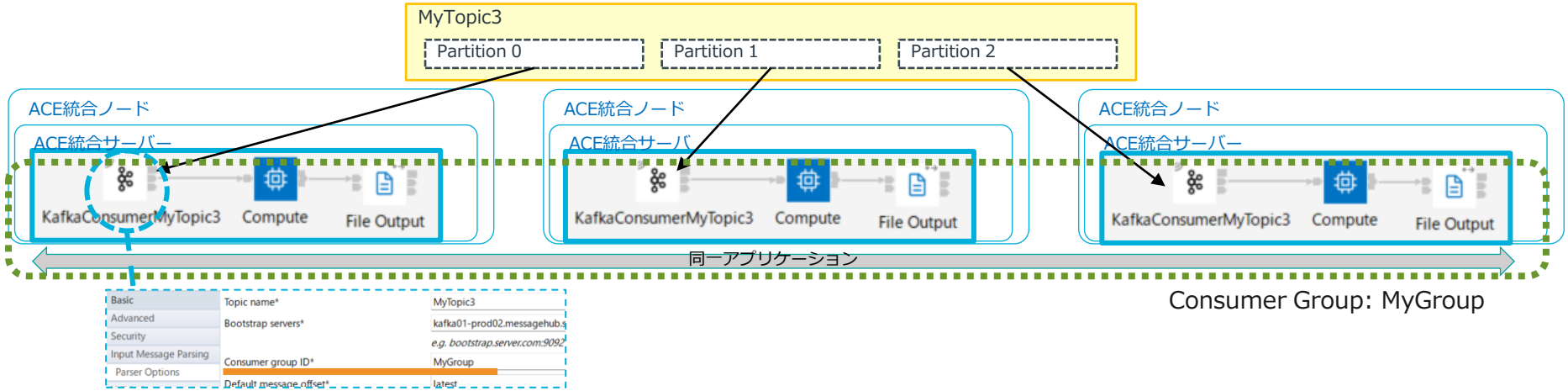
パターン1:負荷分散型の配信

- 大量のメッセージを複数のコンシューマーで分散して処理させたい場合
  - トピックは複数パーティションで構成する
  - コンシューマーは同一のグループIDを指定し、複数のコンシューマー・インスタンスを1つのグループに含める
  - インスタンス数 = パーティション数 とすることで、各インスタンスに均等にメッセージを分散できる
    - いずれかのコンシューマーが障害等でダウンした場合は、パーティションのリバランスが行われ、ダウンしたコンシューマーが受け持っていたパーティションを別のコンシューマーが受け持つ



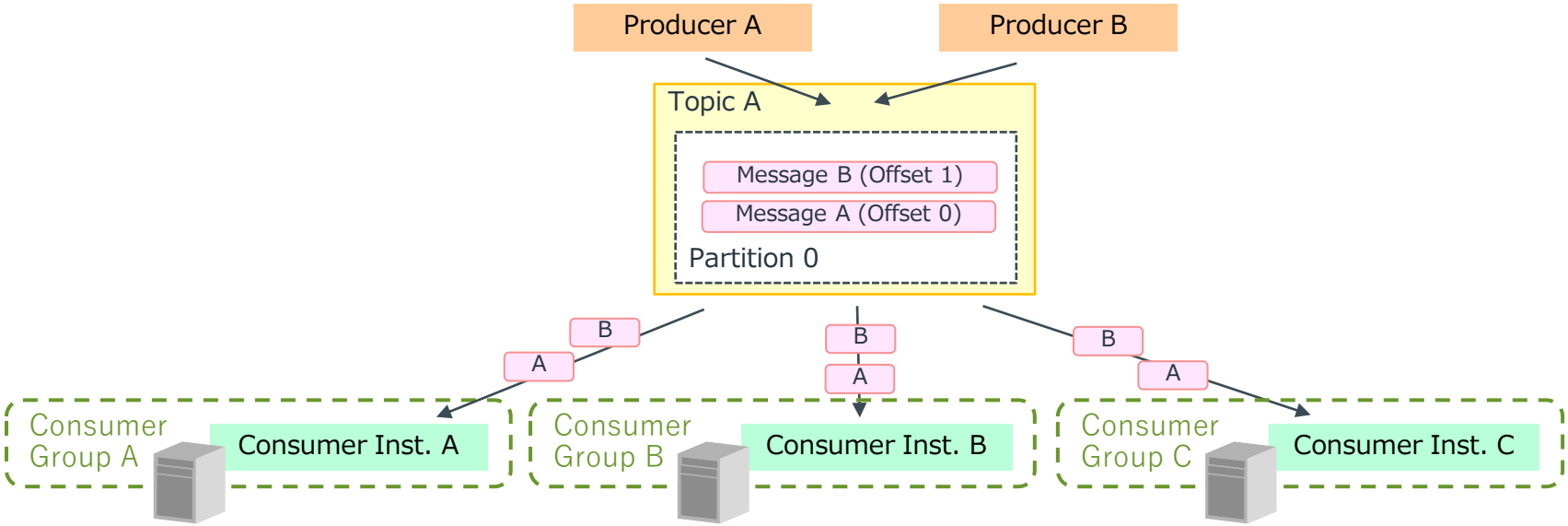
パターン1:負荷分散型の配信 - ノード構成例

- トピックのパーティション構成
  - コンシューマー数と同数またはそれ以上になるように構成
    - コンシューマー数 > パーティション数の場合、パーティション数以上のコンシューマーには、割り当てられるパーティションが無いためメッセージを受け取らない
- Kafka Consumerノード
  - 任意のコンシューマー・グループIDを指定
- アプリケーションの配置
  - 同一のアプリケーションを複数の統合サーバーへデプロイ
    - ✓ コンシューマーグループ(MyGroup)の中で、そのコンシューマーが受け持つパーティションが振り分けられる



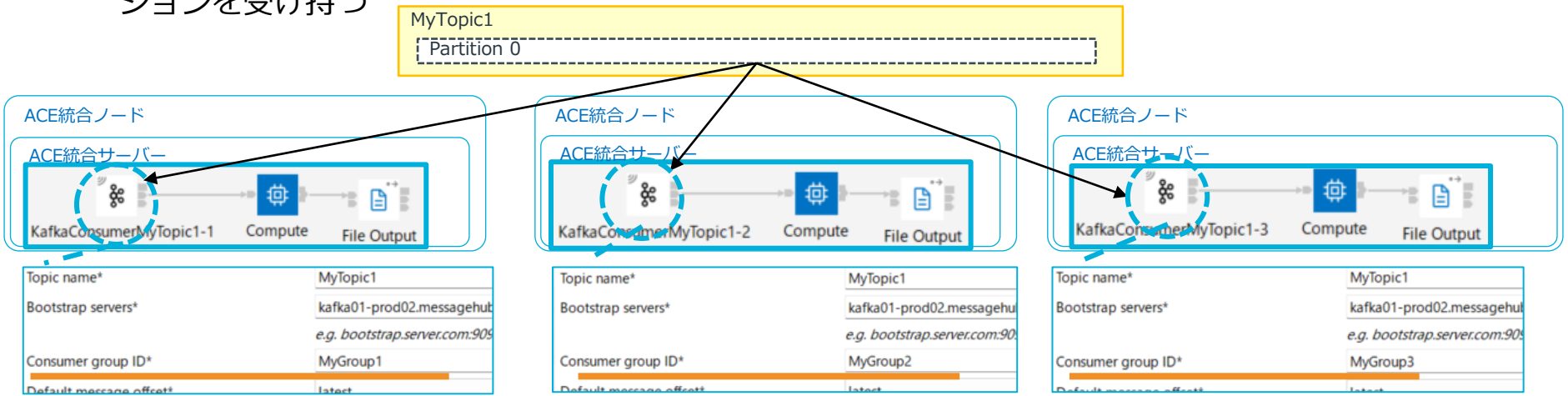
パターン2:同報配信

- 複数のコンシューマーに同一メッセージを配信したい場合
  - トピックは1つ、または複数のパーティションで構成
  - コンシューマーはそれぞれのインスタンス毎に異なるグループIDを指定して、複数のグループを構成
  - 各インスタンス(グループ)に同一メッセージが配信される



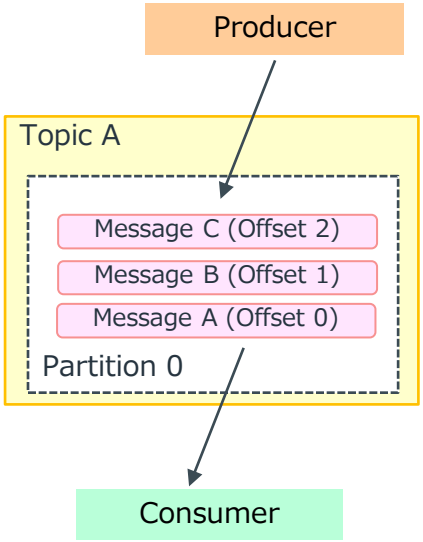
パターン2:同報配信 - ノード構成例

- トピックのパーティション構成
  - 順序性を意識する必要がある場合はパーティションは1つで構成する
  - 順序性を保証しない場合は、複数パーティションで構成することも可能
- Kafka Consumerノード
  - コンシューマー・グループIDの指定
    - それぞれの統合サーバーで稼働するアプリケーション間でユニークなIDを指定する
- アプリケーションの配置
  - コンシューマーグループIDの異なるフローが稼働する為、各々のコンシューマーが全てのパーティションを受け持つ

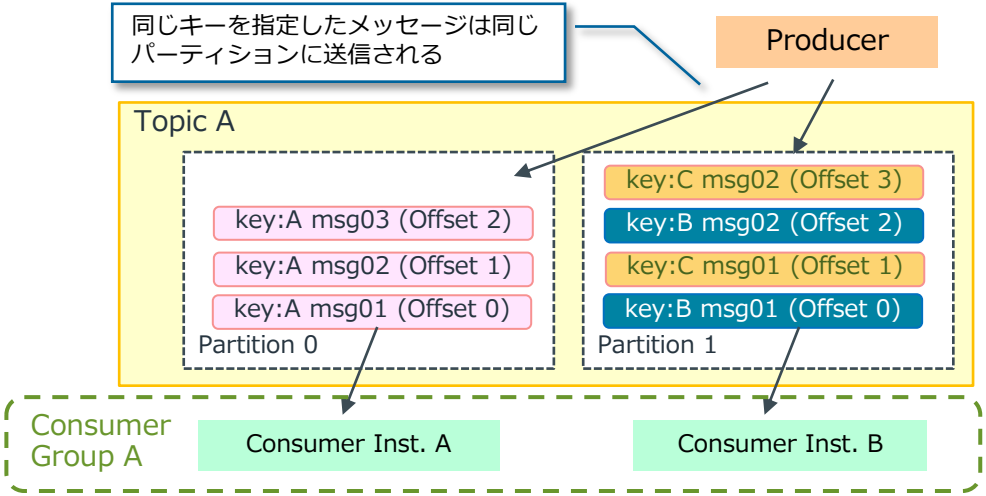


パターン3:順序性を保証した配信

- a. 全てのメッセージの順序性を保証したい場合
  - トピックは1つのパーティションで構成
  - コンシューマーは1つのインスタンスを構成



- b. 特定の複数メッセージの順序性を保証したい場合
  - 複数のメッセージを論理的にグルーピングし、その中での順序性は保証するが、グループ間では順序性を保証しなくて良い場合
  - プロデューサーがキーを使用してメッセージを配信することで、パーティションやコンシューマーを複数構成する事が可能
    - 同一のキーを指定したメッセージは同じパーティ

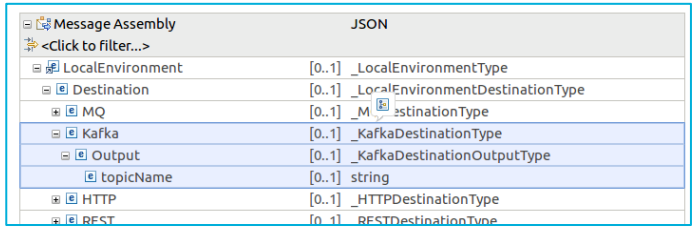


パターン3:順序性を保証した配信 - ノード構成例

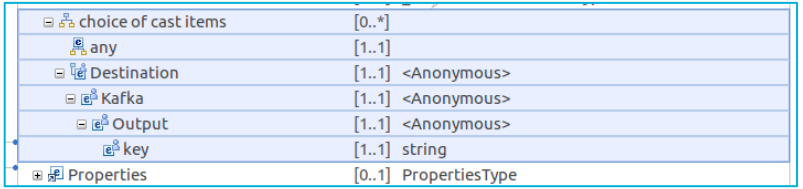
- a. 全てのメッセージの順序性を保証する場合
  - パーティション数1のトピックをサブスクライブする、単一のコンシューマーを構成
    - 構成方法は省略
- b. 特定の複数メッセージの順序性を保証する場合
  - トピックのパーティション構成
    - パーティション数に特に制限なし(単一でも複数でも可)
  - Kafka Producerノードでの配信前にkeyをセット
    - ローカル環境変数でkeyを設定
      - ✓ Computeノードの場合
        - InputLocalEnvironment.Destination.Kafka.Output.keyにセット
      - ✓ Mappingノード
        - ローカル環境変数、Destinationツリー下にkeyの項目が存在しないため、ユーザー定義を作成して指定する

```
CREATE COMPUTE MODULE ProducerFlow_Compute
    CREATE FUNCTION Main() RETURNS BOOLEAN
    BEGIN
        -- CALL CopyMessageHeaders();
        CALL CopyEntireMessage();
        SET InputLocalEnvironment.Destination.Kafka.Output.key = InputRoot.JSON.Data.key;
        RETURN TRUE;
    END;
```

Computeノードでの指定例



MappingノードのDestinationツリー下にはkeyの項目がない



ユーザー定義を追加してkeyをセットする