[AWS Black Belt Online Seminar] Amazon Managed Streaming for Kafka

サービスカットシリーズ

ソリューションアーキテクト
山﨑 翔太・倉光 怜

2019/11/20

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

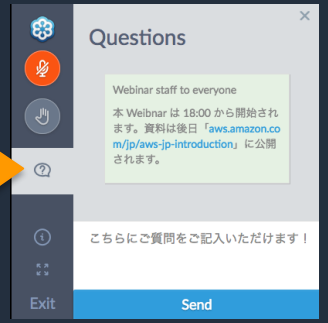
AWS Black Belt Online Seminar とは

「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分かれて、アマゾン ウェブ サービス ジャパン株式会社が主催するオンラインセミナーシリーズです。

質問を投げることができます！

- 書き込んだ質問は、主催者にしか見えません
- 今後のロードマップに関するご質問は
  お答えできませんのでご了承下さい

① 吹き出しをクリック
② 質問を入力
③ Sendをクリック



Twitter ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では2019年11月20日時点のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(http://aws.amazon.com)にてご確認ください。

- 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とさせていただきます。

- 価格は税抜表記となっています。日本居住者のお客様が東京リージョンを使用する場合、別途消費税をご請求させていただきます。

- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at http://aws.amazon.com/agreement/. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

自己紹介

- 名前：山﨑 翔太
- 所属：技術統括本部
  シニアソリューションアーキテクト
- 好きな AWS のサービス
  - Amazon Kinesis Family & AWS Lambda

自己紹介

倉光　怜

所属：ソリューションアーキテクト

自動車業界のお客様を担当

好きなサービス：

AWS Glue

Amazon Kinesis

Amazon S3

本日のアジェンダ

1. Amazon Managed Streaming for Kafka (Amazon MSK) とは
2. Apache Kafka の概要
3. Amazon MSK の機能
    - Amazon MSK を動かすまで
    - Amazon MSK として提供している機能
    - Amazon MSK の運用
4. Amazon MSK のユースケース
    - Amazon Kinesis との使い分け
    - Apache Kafka の周辺ツールとの組み合わせ
5. まとめ

aws

本日のアジェンダ

1. Amazon Managed Streaming for Kafka (Amazon MSK) とは
2. Apache Kafka の概要
3. Amazon MSK の機能
    - Amazon MSK を動かすまで
    - Amazon MSK として提供している機能
    - Amazon MSK の運用
4. Amazon MSK のユースケース
    - Amazon Kinesis との使い分け
    - Apache Kafka の周辺ツールとの組み合わせ
5. まとめ

aws

Amazon Managed Streaming for Apache Kafka (Amazon MSK)

フルマネージドで
可用性が高くセキュアな
Apache Kafka サービス

ストリームデータ

（多数のデータソースによって）

継続的に生成されるデータ

ストリームデータ処理の全体像

ストリームデータ処理の全体像

データソースからデータを取り込んで永続化し、他のサービスへの連携を担う

パブリッシュ・サブスクライブ・モデル

- サブスクライバーは受信したいメッセージを指定して取得する
- １つのメッセージを複数のサブスクライバーが受け取れる

Apache Kafka (Kafka) とは

分散データストリーミング・プラットフォーム

多くの導入実績があり、スケーラビリティに優れた、
パブリッシュ・サブスクライブ・モデルの分散メッセージング基盤を提供

aws

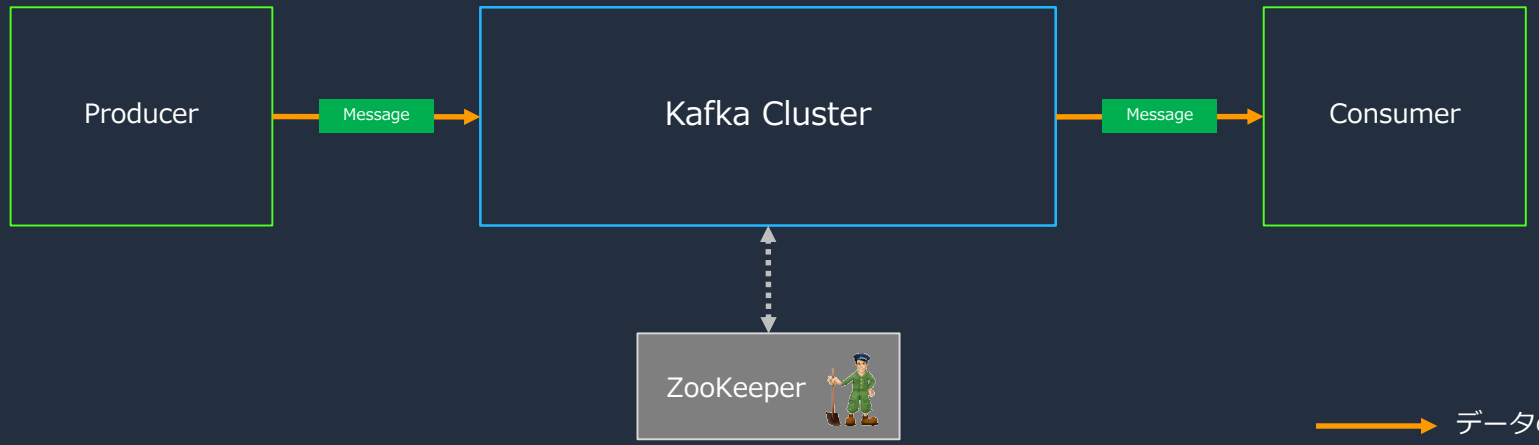
本日のアジェンダ

1. Amazon Managed Streaming for Kafka (Amazon MSK) とは
2. Apache Kafka の概要
3. Amazon MSK の機能
    - Amazon MSK を動かすまで
    - Amazon MSK として提供している機能
    - Amazon MSK の運用
4. Amazon MSK のユースケース
    - Amazon Kinesis との使い分け
    - Apache Kafka の周辺ツールとの組み合わせ
5. まとめ

aws

Apache Kafkaの全体像

- Kafkaのクライアントとして、プロデューサーとコンシューマーが存在する
- プロデューサーはメッセージの発行（パブリッシュ）を行い、
  コンシューマーはメッセージの購読（サブスクライブ）を行う
- Kafkaは複数のサーバーでクラスターを構成しており、スケーラビリティと可用性を
  確保している

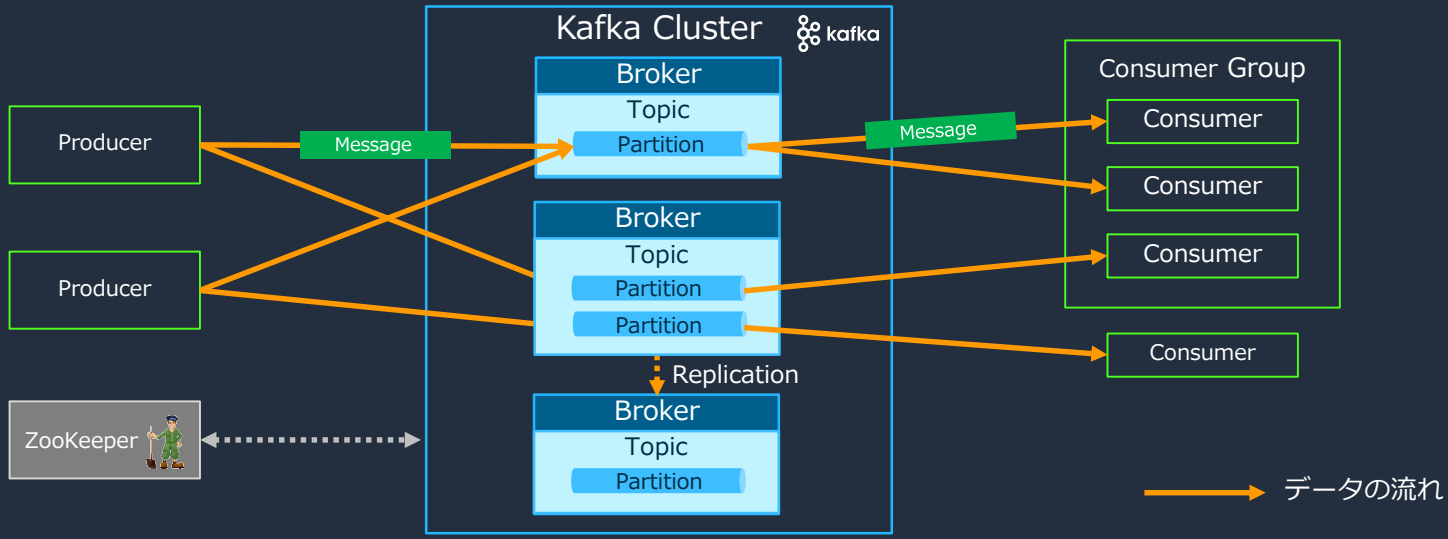


Apache Kafkaの全体像

- Apache Kafka の構成要素として、プロデューサーとコンシューマーが存在する
- プロデューサーはメッセージの発行 (Publish) を行い、コンシューマーはメッセージの購読 (Subscribe) を行う
- 複数のKafkaサーバーでクラスターを構成しており、可用性を確保している

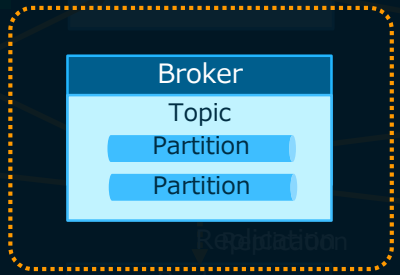
Apache Kafkaの全体像

- Kafkaのクラスターは、複数のブローカーで構成され、トピック内のパーティションを分散キューとしてブローカーに配置してメッセージを管理する
- ZooKeeper が、トピックやパーティションのメタ情報を管理する
- プロデューサーはブローカーにメッセージを送信し、コンシューマーはブローカーから取り出して利用する



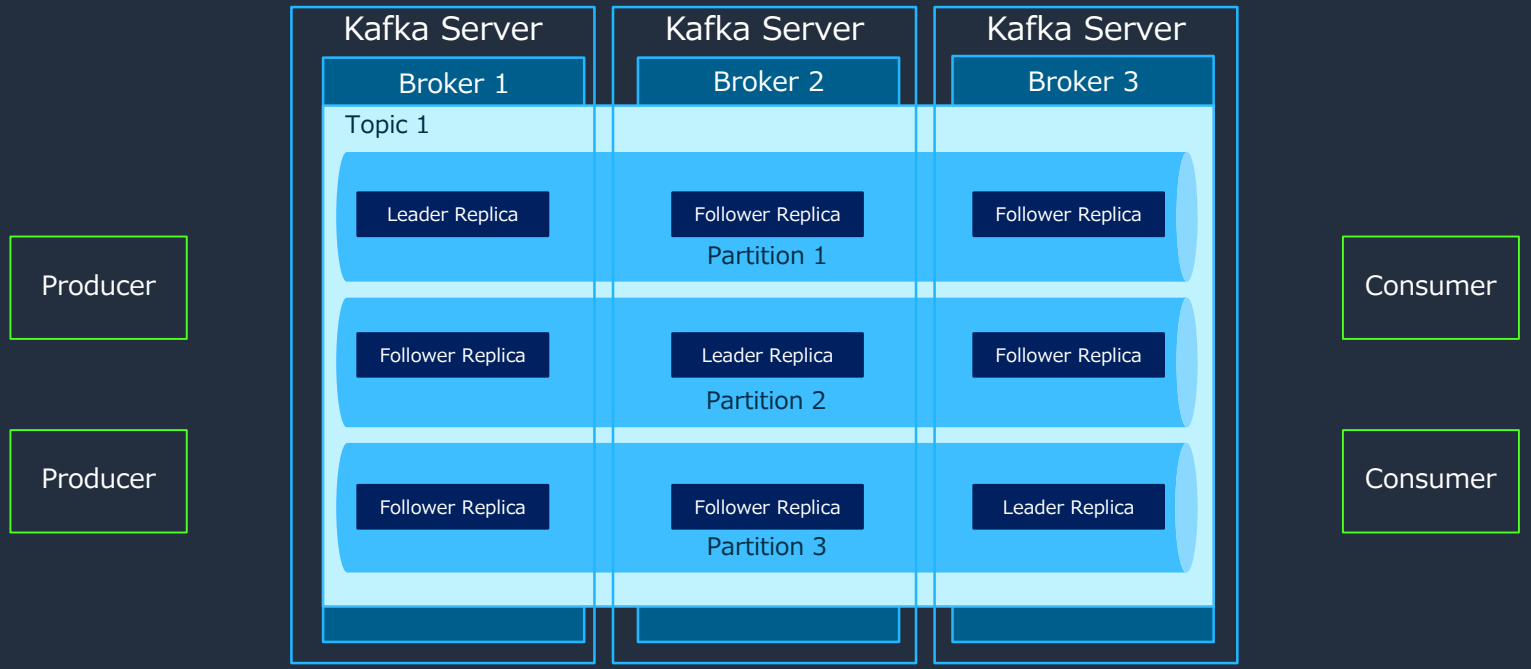
Apache Kafkaの全体像

- ブローカー：クラスターとして動作し、データの受配信を担う
- トピック：メッセージを種別で管理する
- パーティション：ブローカー上に分散配置され、トピックのメッセージが格納される



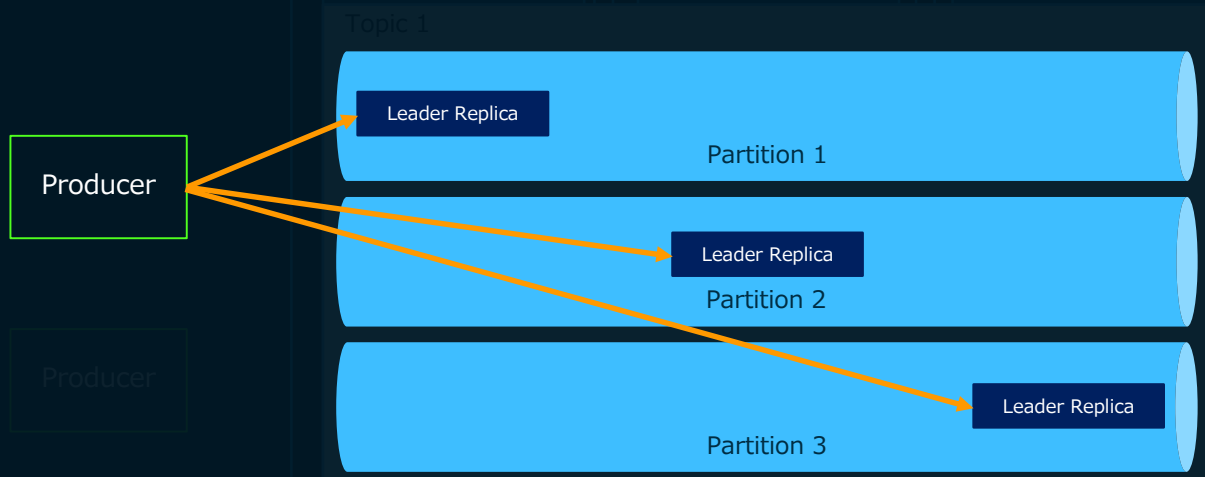
トピックとパーティション

- メッセージの冗長性確保のために、各パーティションにはブローカー間でコピーされた複数のレプリカを構成する（１つのリーダーレプリカと複数のフォロワーレプリカ）



トピックとパーティション

- メッセージの冗長性確保のために、各パーティションにはブローカー間でコピーされた複数のレプリカを構成する（１つのリーダーレプリカと複数のフォロワーレプリカ）
- **プロデューサーはトピックと値を指定することで書き込みを行う**
- **各パーティションへのメッセージの書き込みはリーダーレプリカのみ実施される**



トピックとパーティション

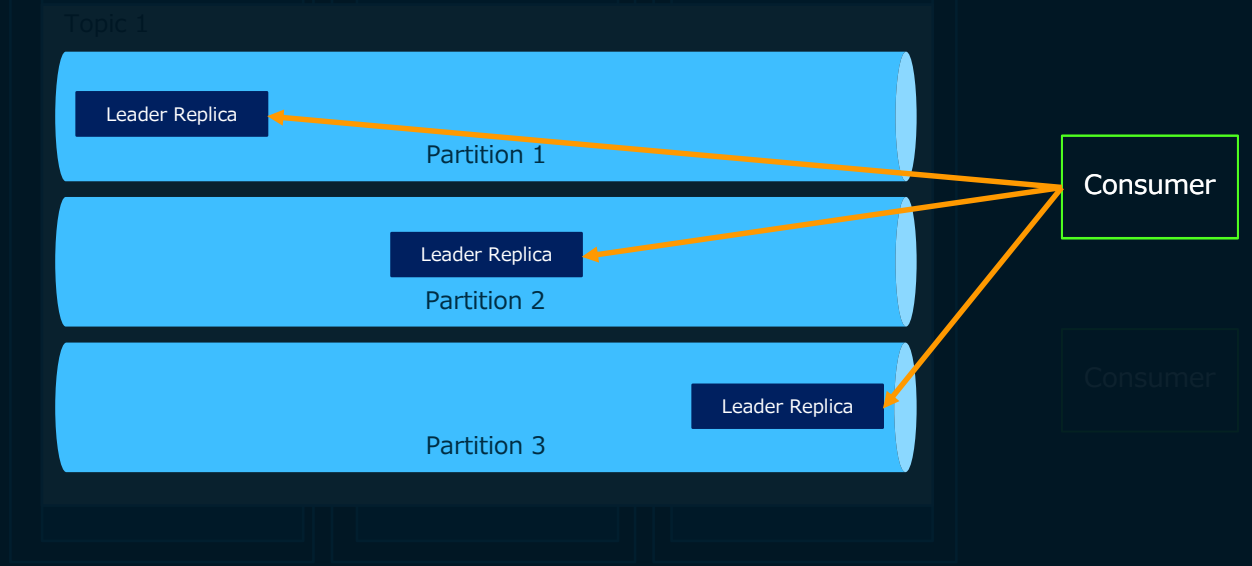
- メッセージの冗長性確保のために、各パーティションにはブローカー間でコピーされた複数のレプリカを構成する（１つのリーダーレプリカと複数のフォロワーレプリカ）

- **リーダーレプリカに書き込まれた内容はフォロワーレプリカに複製する**
- **リーダーレプリカはフォロワーレプリカと同期がとれているかを把握している**

トピックとパーティション

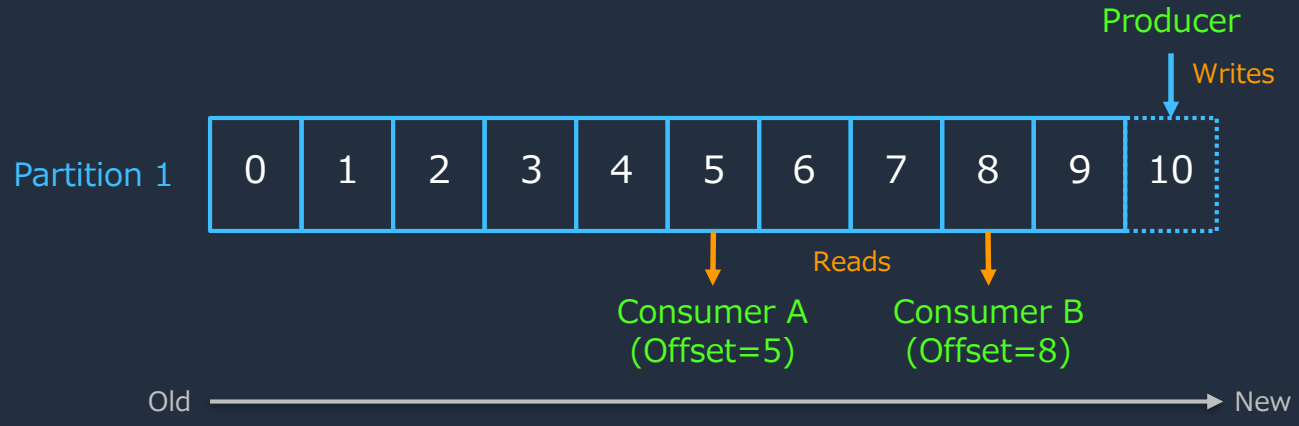
- メッセージの冗長性確保のために、各パーティションにはブローカー間でコピーされた複数のレプリカを構成する（１つのリーダーレプリカと複数のフォロワーレプリカ）

コンシューマーのメッセージの読み込みについても、
プロデューサーと同様に、リーダーレプリカのみから行われる



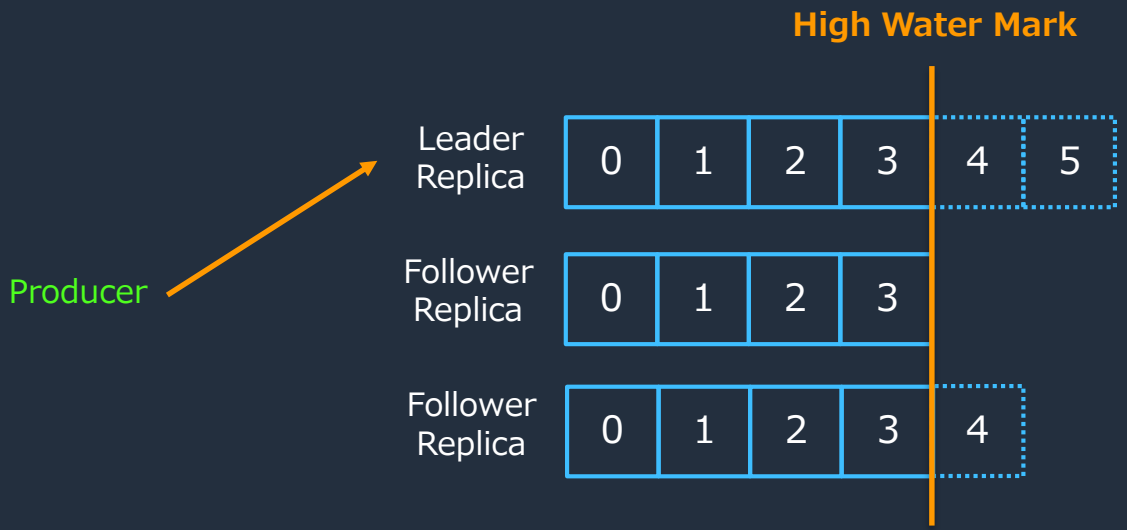
オフセット

- 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
- パーティション単位で最後に取得したメッセージを ZooKeeper もしくは Kafka自体が保存しており、コンシューマーが追跡している
- 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



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



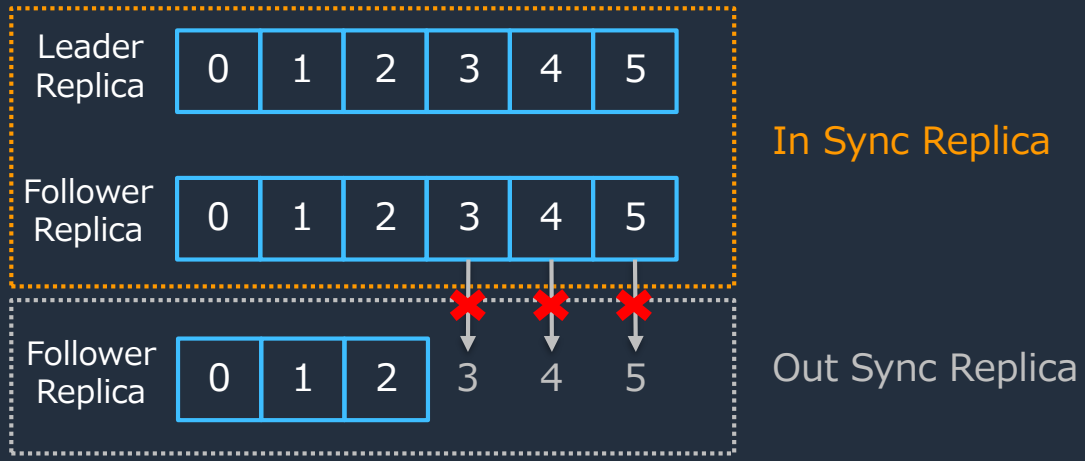
ハイウォーターマーク

- 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

In Sync Replica と Out Sync Replica

- リーダーとフォロワーが同じ状態であるレプリカを In Sync Replica と言う
- ブローカーやネットワークの問題により、一定時間以上で同期できない場合は Out Sync Replica と見なされる
- min.insync.replicas パラメーターで最低限必要な In Sync Replica を設定する



Apache Kafka のまとめ

- **Apache Kafka は、分散データストリーミング・プラットフォームである**
  - ✓ 多くの導入実績がある
  - ✓ スケーラビリティに優れたパブリッシュ・サブスクライブ・モデルの分散メッセージング基盤を提供する
- **Kafka のクラスターは、ブローカー・トピック・パーティションで構成される**
  - ✓ ブローカー：データの受配信を担う
  - ✓ トピック：メッセージを種別で管理する
  - ✓ パーティション：ブローカー上に配置され、トピックのメッセージを格納する
- **スケーラビリティのために分散アーキテクチャを採用していて構成要素が多く、環境構築や運用の負担は大きい**
  - ✓ Kafkaクラスターの環境構築においては、多くのサーバーに対する設定が必要となる
  - ✓ 稼働後にも、ブローカーや Zookeeper 自体の運用に加えて、スケールの管理や監視などの多くの運用作業が必要となる

aws

本日のアジェンダ

1. Amazon Managed Streaming for Kafka (Amazon MSK) とは
2. Apache Kafka の概要
3. Amazon MSK の機能
    - Amazon MSK を動かすまで
    - Amazon MSK が提供する機能
    - Amazon MSK の運用
4. Amazon MSK のユースケース
    - Amazon Kinesis との使い分け
    - Apache Kafka の周辺ツールとの組み合わせ
5. まとめ

aws

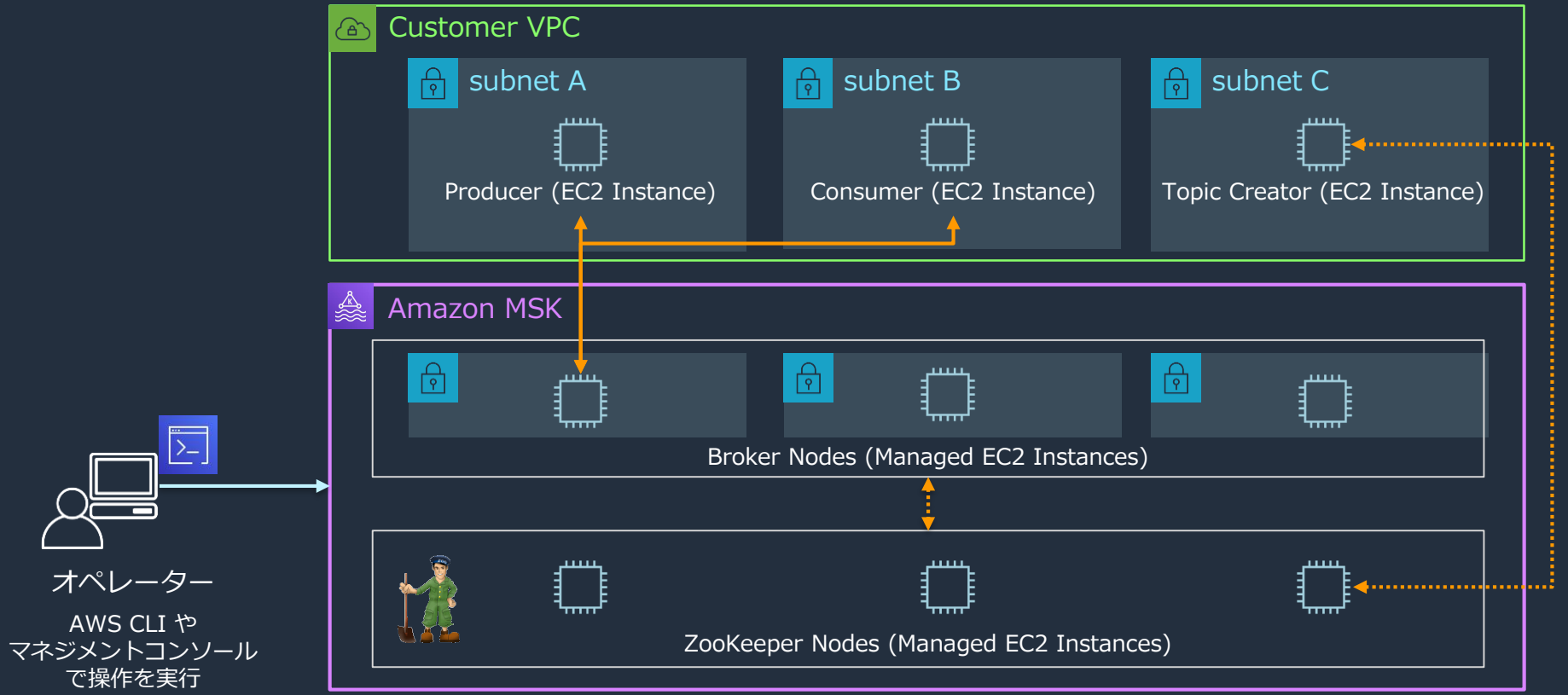
Amazon MSK とは

フルマネージドで可用性が高くセキュアな Apache Kafka サービス

- Amazon MSK は、コントロールプレーンの操作を提供
  - クラスターの作成、更新、削除などの API を提供
- データプレーンの操作は、Apache Kafka の API をそのまま使用可能
  - トピックの作成や管理
  - プロデューサーからのデータの入力や、コンシューマーからのデータの取得
- Amazon MSK は Apache Kafka のオープンソースバージョンを実行
  - Kafka のバージョン 1.1.1 と 2.2.1 をサポート（2019年11月現在）

aws

Amazon MSK のアーキテクチャ



Amazon MSK を動かすまで

注意点

最新の AWS CLI がインストールされていることをご確認下さい

```
$ aws –version
aws-cli/1.16.266 Python/3.7.3 Linux/4.14.138-114.102.amzn2.x86_64 botocore/1.13.2


$ pip3 install --upgrade --user awscli
…
```

https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-chap-install.html

クラスターの準備

Amazon MSK の Create Cluster API で簡単に Kafka クラスターを作成可能

```
$ aws kafka create-cluster ¥
      --cluster-name "AWSKafkaTutorialCluster” ¥
      --broker-node-group-info file://brokernodegroupinfo.json ¥
      --encryption-info file://encryptioninfo.json ¥
      --kafka-version "2.2.1" ¥
      --number-of-broker-nodes 3 ¥
      --enhanced-monitoring PER_TOPIC_PER_BROKER ¥
      --region ap-northeast-1
{
   "ClusterArn": ”ClusterArn",
   "ClusterName": "AWSKafkaTutorialCluster",
   "State": "CREATING"
}
```

ブローカーを配置する
VPCサブネット等の設定

クラスターのストレージと
転送中のデータに対する暗号化の設定

https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/getting-started.html
https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/create-cluster.html

aws

トピックの準備

Kafka のクライアントツールをインストール

```
$ sudo yum install java-1.8.0

$ wget https://archive.apache.org/dist/kafka/2.2.1/kafka_2.12-2.2.1.tgz

$ cd kafka_2.12-2.2.1/

$ ls
LICENSE  NOTICE  bin  config  libs  site-docs
```

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

aws

トピックの準備

```
# Amazon MSK の Describe Cluster API で 接続するZooKeeper (ZookeeperConnectString) を取得
$ aws kafka describe-cluster --cluster-arn "ClusterArn”
{
   "ClusterInfo": {
      …,
      "ClusterName": "AWSKafkaTutorialCluster",
      "NumberOfBrokerNodes": 3,
      "ZookeeperConnectString": "z-3.awskafkatutorialc.6xjbqy.c4.kafka.ap-northeast-
1.amazonaws.com:2181,z-1.awskafkatutorialc.6xjbqy.c4.kafka.ap-northeast-1.amazonaws.com:2181,z-
2.awskafkatutorialc.6xjbqy.c4.kafka.ap-northeast-1.amazonaws.com:2181",
      …
   }
}

# Kafka の API でトピックを作成
$ bin/kafka-topics.sh --zookeeper ZookeeperConnectionString ¥
       --create --replication-factor 3 --partitions 1 --topic AWSKafkaTutorialTopic
Created topic AWSKafkaTutorialTopic.
```

トピックの準備

ワンラインで記述

```
$ sudo yum -y install jq

# Amazon MSK の Describe Cluster API で取得した ZookeeperConnectString に対してトピックを作成
$ bin/kafka-topics.sh --zookeeper `aws kafka describe-cluster
--cluster-arn “ClusterArn” | jq -r .ClusterInfo.ZookeeperConnectString`
--create --replication-factor 3 --partitions 1 --topic AWSKafkaTutorialTopic

Created topic AWSKafkaTutorialTopic.

#トピックの一覧を表示
$ bin/kafka-topics.sh --zookeeper `aws kafka describe-cluster
--cluster-arn “ClusterArn” | jq -r .ClusterInfo.ZookeeperConnectString` --list

AWSKafkaTutorialTopic
```

プロデューサーとコンシューマー の接続

```
# 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 API でプロデューサーを接続
$ bin/kafka-console-producer.sh --broker-list BootstrapBrokerString ¥
        --producer.config client.properties --topic AWSKafkaTutorialTopic
> Hello Kafka!
> Hello MSK!
```

```
# Kafka の API でコンシューマーを接続
$ bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString ¥
        --consumer.config client.properties --topic AWSKafkaTutorialTopic --from-beginning
Hello Kafka!
Hello MSK!
```

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

aws

AWS Big Data Blog - ”Best Practices for Running Apache Kafka on AWS”

AWS での Apache Kafka の実行のためのベストプラクティス

「Amazon EC2 上で Apache Kafka を動かす際のベストプラクティス」
を以下の項目毎にガイドしたブログ

1. デプロイメントのパターン
2. ストレージオプション
3. インスタンスタイプ
4. ネットワーキング
5. アップグレード
6. パフォーマンスチューニング
7. モニタリング
8. セキュリティ
9. バックアップとリストア

https://aws.amazon.com/jp/blogs/news/best-practices-for-running-apache-kafka-on-aws/

Kafka on EC2 と Amazon MSK を利用した場合の違い

＊ 後述

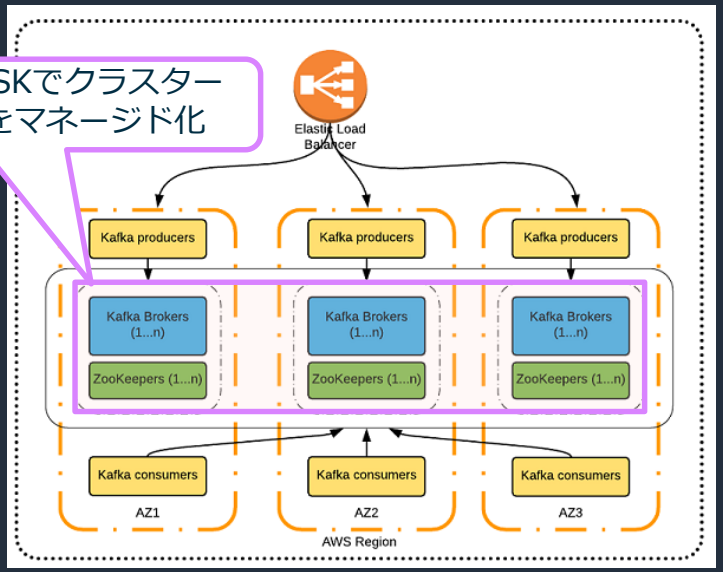
項目	Kafka on EC2	Amazon MSK
デプロイメントのパターン	複数AZへのデプロイを軸に自身で構築	複数AZに自動的にデプロイ*
ストレージオプション	EBS または インスタンスストア	（MSKが管理する）EBS
インスタンスタイプ	任意のインスタンスタイプ	m5ファミリー（large～24xlarge）
ネットワーキング	インスタンスタイプやENI構成を 考慮して自身で設計	クラスター内はマネージドとなり、 クライアントからはENI経由で接続*
バージョンアップグレード	要件に合わせて選択	現時点は Blue/Green が基本*
パフォーマンスチューニング	Kafka のパフォーマンスチューニングにおいては、ほとんど違いはない	
監視	自身で監視ツールとの連携を構築	MSKの監視機能を利用*
セキュリティ	自身で暗号化や認証を設定	MSKのセキュリティ機能を利用*
バックアップとリストア	EBSスナップショットを取得 MirrorMaker でクラスターをコピー	クラスターのバックアップ機能は未提供 MirrorMaker でクラスターをコピー

aws

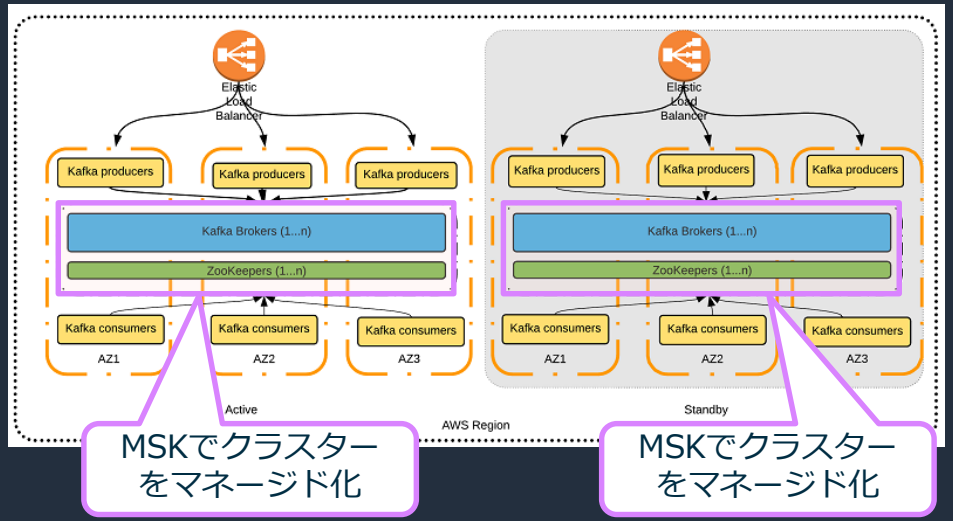
クラスターのデプロイメント

- AZ間でブローカーを均等にデプロイするベストプラクティスを適用
- 2つもしくは3つのAZに含まれるサブネットをクラスター作成時に指定
- AZ当たりのブローカー数のみ、あるいはAZ数の倍数でブローカー数を指定

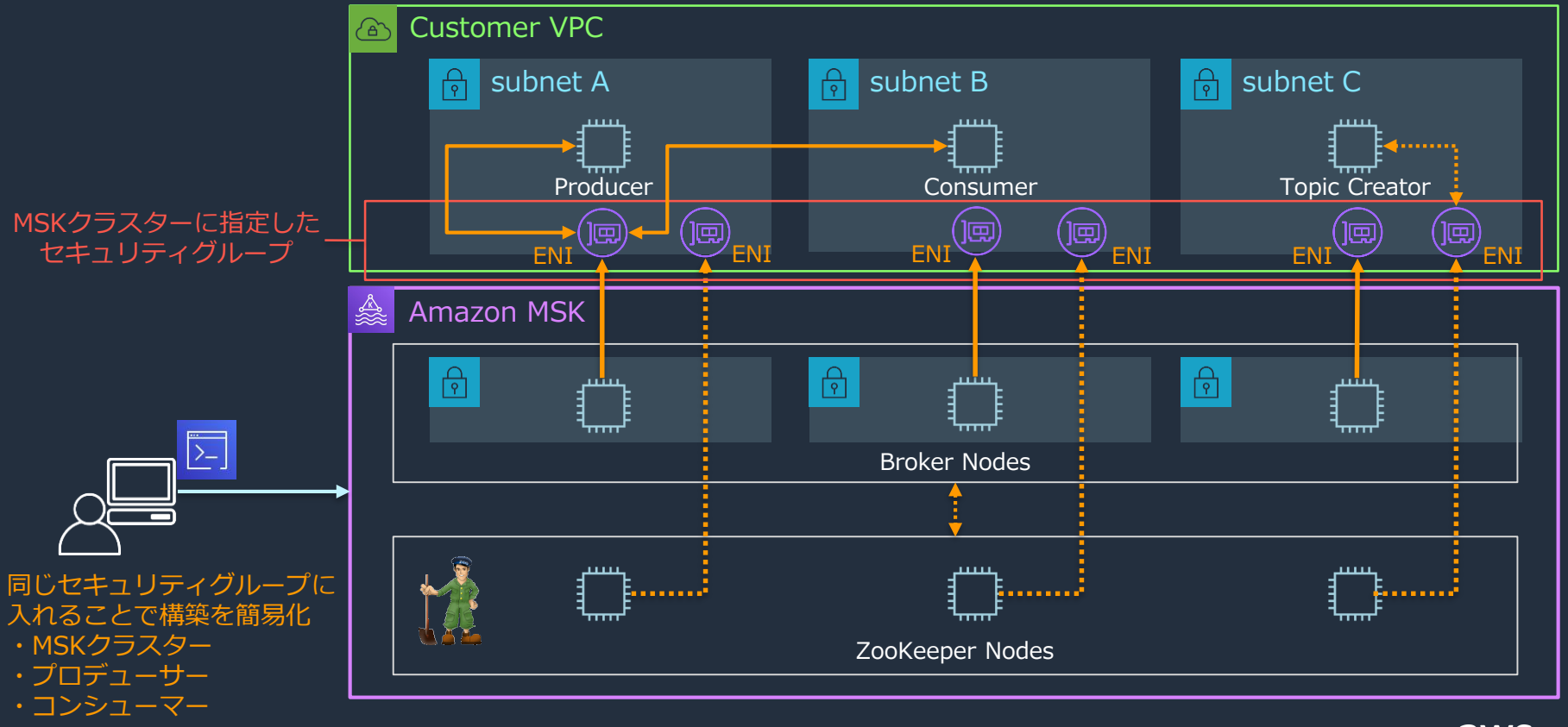
シングルリージョン・3AZ
シングルクラスター構成



シングルリージョン・3AZ
アクティブ - スタンバイ構成



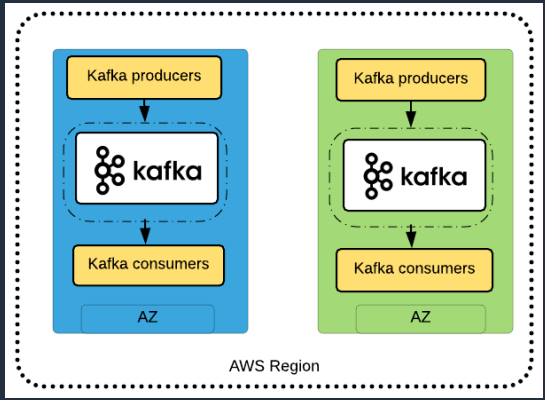
Amazon MSK のネットワーク構成



クラスターのバージョンアップグレード

Kafka on EC2

要件に合わせて選択
- ローリング や インプレース
- クラスターの一時停止
- Blue/Green



Amazon MSK

- クラスターのアップグレードは未サポート
  （今後の機能追加でサポート予定）
- Blue/Green での切り替えは容易に

1. **新しいKafkaクラスター**を作成（**Green**）
2. **GreenのKafkaクラスター**を指す
   新しいKafkaのプロデューサースタックを作成
3. **GreenのKafkaクラスター**に関するトピックを作成
4. **新しいGreen環境**へのデプロイを一気通貫でテスト
5. Amazon Route 53 を使用して、
   新しいKafkaのプロデューサースタックを更新し、
   GreenのKafkaクラスターを指すように変更

Blue/Green での切り替え

https://aws.amazon.com/jp/msk/faqs/
https://aws.amazon.com/jp/blogs/big-data/best-practices-for-running-apache-kafka-on-aws/

Amazon MSK の監視機能

- Amazon CloudWatch でメトリクスを取得可能
- 以下の 3 つからクラスターのモニタリングレベルを選択可能

モニタリングレベル	拡張モニタリングの設定値	メトリクスの内容	利用料
基本モニタリング	DEFAULT	基本的なクラスターレベルおよびブローカーレベルのメトリクス	無料
拡張ブローカーレベルモニタリング	PER_BROKER	基本モニタリングおよび拡張ブローカーレベルのメトリクス	有料
拡張トピックレベルモニタリング	PER_TOPIC_PER_BROKER	拡張ブローカーレベルおよび拡張トピックレベルのメトリクス	有料

よくある Kafka の監視項目

- ブローカーの死活監視 ⇨ 基本モニタリングのメトリクスで正常性確認
- トピックレベルでの流入量 ⇨ 拡張トピックレベルのメトリクスで監視
- エンドツーエンドのレイテンシー ⇨ Burrow などのツールも活用してコンシューマー側で計測

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

取得可能なメトリクスの例：基本モニタリング

メトリクス名	モニタリングレベル	ディメンション	説明
ZooKeeperSessionState	DEFAULT	Cluster Name, Broker ID	ブローカーのZooKeeper セッションの接続状態
GlobalPartitionCount	DEFAULT	Cluster Name	クラスター内の全ブローカーにあるパーティションの総数
GlobalTopicCount	DEFAULT	Cluster Name	クラスター内の全ブローカーにあるトピックの総数
MemoryUsed	DEFAULT	Cluster Name, Broker ID	ブローカーに使用されているメモリのサイズ（bytes）
CpuIdle	DEFAULT	Cluster Name, Broker ID	CPUアイドル時間の割合
KafkaDataLogsDiskUsed	DEFAULT	Cluster Name, Broker ID	データログに使用されているディスク容量の割合
NetworkRxErrors	DEFAULT	Cluster Name, Broker ID	ブローカーのネットワーク受信エラーの数

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

取得可能なメトリクスの例：拡張ブローカーレベル

メトリクス名	モニタリングレベル	ディメンション	説明
LeaderCount	PER_BROKER	Cluster Name, Broker ID	ブローカーが持つ リーダーレプリカの数
PartitionCount	PER_BROKER	Cluster Name, Broker ID	ブローカーが持つ パーティションの数
MessagesInPerSec	PER_BROKER	Cluster Name, Broker ID	1秒あたりにクライアント から受信したメッセージの数
ProduceThrottleByteRate	PER_BROKER	Cluster Name, Broker ID	1秒あたりにスロットリング されたバイト数
RequestTime	PER_BROKER	Cluster Name, Broker ID	ブローカーのネットワークと I/Oスレッドでリクエストの 処理に費やした平均時間
RequestThrottleQueueSize	PER_BROKER	Cluster Name, Broker ID	スロットリングした キュー内のメッセージの数

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

aws

取得可能なメトリクスの例：拡張トピックレベル

メトリクス名	モニタリングレベル	ディメンション	説明
BytesInPerSec	PER_TOPIC_PER_BROKER	Cluster Name, Broker ID, Topic	（トピック毎の）1秒あたりの受信バイト数
BytesOutPerSec	PER_TOPIC_PER_BROKER	Cluster Name, Broker ID, Topic	（トピック毎の）1秒あたりの送信バイト数
MessagesInPerSec	PER_TOPIC_PER_BROKER	Cluster Name, Broker ID, Topic	（トピック毎の）1秒あたりにクライアントから受信したメッセージの数

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

Amazon MSK のセキュリティ機能

データの保護

- 保存データの暗号化
- データ通信の暗号化

IAMによる操作権限管理

- IAMポリシーやIAMロールでの制御
- タグ付けとタグベースのIAM設定

クライアントの認証

TLSクライアント認証

コンプライアンス

PCI や HIPAA BAA の認証

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

Amazon MSK のセキュリティ機能

データの保護

- 保存データの暗号化
- データ通信の暗号化

IAMによる操作権限管理

- IAMポリシーやIAMロールでの制御
- タグ付けとタグベースの認証

クライアントの認証

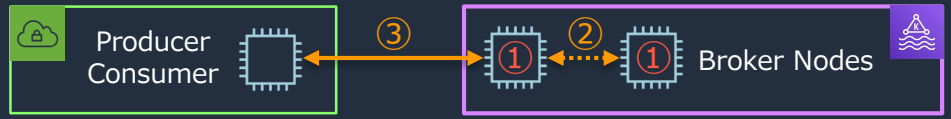
TLSクライアント認証

コンプライアンス

PCI や HIPAA BAA の認証

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

データの保護



保存データの暗号化

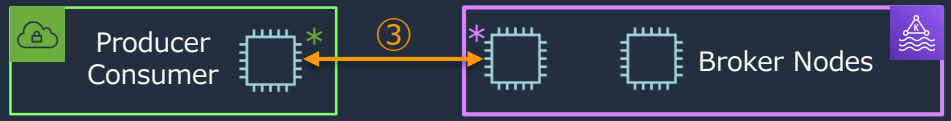
- KMSと連携したサーバー側の暗号化①がデフォルトで有効
(Server Side Encryption)
- クラスター作成時に、
Customer Master Key (CMK) を指定
- 指定しなければ、
MSK が AWS Managed CMK を作成

データ通信の暗号化

- ブローカー同士の通信②はデフォルトで暗号化が有効で、無効化することも可能
- 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で暗号化されたデータのみを許可（デフォルトの設定）
  2. プレーンテキストとTLSで暗号化されたデータの両方を許可
  3. プレーンテキストのデータのみを許可
- TLS 1.2 を利用
- データ通信の暗号化を有効にすると、パフォーマンスは約30%低下

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

TLSクライアント認証



クライアント認証

- MSKクラスターの作成時に、
  ACM Private Certificate Authority
  (ACM PCA) のプライベートCAを指定
- クライアント側に証明書を設定
  1. 秘密鍵を作成し、秘密鍵を使用して
     証明書署名要求（CSR）を作成
  2. ACM PCA で CSR に署名して証明書を発行
  3. ACM PCA が署名した証明書をクライアント
     に設定してブローカーと通信
- Kafka Authorization CLI を利用して、
  トピックに対してアクセス権を設定

ブローカー側のサーバー証明

- クライアントとブローカー間の通信の
  TLS暗号化③を有効化するのみ
  - クライアント認証をするには有効化が必要
- ブローカーは AWS Certificate Manager
  (ACM) の証明書を使用
- Amazon Trust Services を信頼する
  トラストストアは、MSKのブローカーの
  証明書も信頼

https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/msk-authentication.html
https://cwiki.apache.org/confluence/display/KAFKA/Kafka+Authorization+Command+Line+Interface

aws

Amazon MSK によるKafkaクラスターの構成管理

- MSKは、ブローカー、トピック、ZooKeeperノードのデフォルト構成を提供
- カスタム構成を作成し、新規のMSKクラスターを作成することが可能
- カスタム構成で、既存のクラスターを更新することも可能

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

デフォルト構成

設定項目	初期値
num.network.threads	5
num.io.threads	8
num.replica.fetchers	2
socket.send.buffer.bytes	102400
socket.receive.buffer.bytes	102400
socket.request.max.bytes	104857600
num.partitions	1
auto.create.topics.enable	false
default.replication.factor	3
min.insync.replicas	2
unclean.leader.election.enable	true
auto.leader.rebalance.enable	true
allow.everyone.if.no.acl.found	true
zookeeper.set.acl	false

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

aws

カスタム構成で設定可能な項目

設定項目
auto.create.topics.enable
delete.topic.enable
log.cleaner.delete.retention.ms
group.initial.rebalance.delay.ms
group.max.session.timeout.ms
group.min.session.timeout.ms
log.flush.interval.messages
log.flush.interval.ms
log.retention.bytes
log.retention.hours
log.retention.minutes
log.retention.ms
num.partitions

設定項目
max.incremental.fetch.session.cache.slots
log.cleaner.min.cleanable.ratio
offsets.retention.minutes
zookeeper.connection.timeout.ms
unclean.leader.election.enable
min.insync.replicas
message.max.bytes
log.segment.bytes
log.roll.ms
transaction.max.timeout.ms
replica.fetch.max.bytes
compression.type
default.replication.factor

設定項目
message.timestamp.difference.max.ms
message.timestamp.type
num.io.threads
num.network.threads
num.recovery.threads.per.data.dir
num.replica.fetchers
offsets.topic.replication.factor
replica.fetch.response.max.bytes
socket.request.max.bytes
transaction.state.log.min.isr
transaction.state.log.replication.factor
log.cleanup.policy

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

aws

Amazon MSK による構成管理の注意点

- （現時点では）Configurationを削除するAPIはない
  - ✓ AWSアカウントごとに作成できるConfiguration数の上限に注意（デフォルトでは100）
- （現時点では）Configurationを更新するAPIはない
  - ✓ ConfigurationのRevisionを指定してクラスターを更新するが、Revisionを上げる方法はない
- クラスターを更新すると、ブローカーとの順次の接続断が発生する
  - ✓ クラスターのステータスは「更新中」になる
  - ✓ ブローカーはローリングアップグレードされ、順次の接続断が発生する

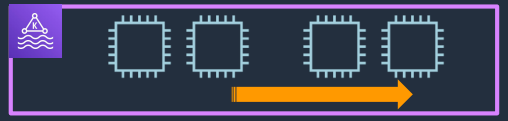
```
$ bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString ¥ --consumer.config client.properties --topic AWSKafkaTutorialTopic --from-beginning
Hello Kafka!
Hello MSK!
[2019-11-10 03:31:54,014] WARN [Consumer clientId=consumer-1, groupId=console-consumer-74932] Connection to node 2 (b-2.awskafkatutorialc.6xjbqy.
c4.kafka.ap-northeast-1.amazonaws.com/10.0.0.214:9094) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
Are you OK?
[2019-11-10 03:33:03,988] WARN [Consumer clientId=consumer-1, groupId=console-consumer-74932] Connection to node 1 (b-1.awskafkatutorialc.6xjbqy.
c4.kafka.ap-northeast-1.amazonaws.com/10.0.1.96:9094) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
We can still consume data
[2019-11-10 03:34:16,763] WARN [Consumer clientId=consumer-1, groupId=console-consumer-74932] Connection to node 3 (b-3.awskafkatutorialc.6xjbqy.
c4.kafka.ap-northeast-1.amazonaws.com/10.0.2.175:9094) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
Done!
Hello updated cluster!
```


aws

クラスターのスケーリング

ストレージの拡張

- マネジメントコンソールまたは AWS CLI を使用して、ブローカーのストレージ容量を拡張可能
- クラスター内の全ブローカーに対する一括での拡張のみ可能
- 容量を増やすことは出来るが、減らすことは出来ない



ブローカー数の拡張

- マネジメントコンソールまたは AWS CLI を使用して、ブローカーのノード数を拡張可能
- AZあたりのブローカー数を指定して拡張
- 数を増やすことは出来るが、減らすことは出来ない

ブローカーサイズの拡張

現時点では、ブローカーのスケールアップは未サポート
（今後の機能追加でサポート予定）

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://aws.amazon.com/jp/msk/faqs/

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
    1. ブローカーへのネットワーク接続の損失
    2. ブローカーノードの障害

- 障害を検出すると、正常でない（到達不能な）ブローカーを新しいブローカーに置換
    - ✓ 可能であれば、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が検出と回復を完了するのに必要な時間に制限される
    - ✓ 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できる

- Amazon MSK は、99.9%の可用性（SLA）を提供

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

aws

Amazon MSK のベストプラクティス

1. **クラスターを適切なサイズに設定する**
   - ✓ MSK Sizing and Pricing スプレッドシートを利用してサイズを決定
     https://amazonmsk.s3.amazonaws.com/MSK_Sizing_Pricing.xlsx
2. **ディスク容量を監視する**
   - ✓ KafkaDataLogsDiskUsed メトリクスを監視し、85%を超えたらいずれかの方法で対処
     1. ブローカーのストレージ容量を増やす
     2. メッセージの保存期間またはログサイズを減らす
     3. 未使用のトピックを削除する
3. **Kafka の Data Retention パラメーターを指定する**
   - ✓ ディスクの空き容量を定期的に解放するために、保持期間を明示的に指定
   - ✓ log.retention.hours などの設定により、クラスターレベルまたはトピックレベルで、Data Retentionパラメーターを指定
4. **MSK の管理外にあるブローカーを追加しない**
5. **データ通信の暗号化を有効にする**
6. **パーティションの再割り当てを行う**
   - ✓ パーティション再割り当てツール kafka-reassign-partitions.sh を使用
   - ✓ 新しいブローカーの追加後にはパーティションを再割り当てしてリバランスを実施

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

aws

Amazon MSK の料金体系

ブローカーインスタンスの料金

- ブローカーインスタンスの使用量に応じて時間単位で発生 (1秒ごとに請求)
- 料金は、ブローカーインスタンスのサイズや、アクティブなブローカーの数によって異なる

ブローカーストレージの料金

- クラスターにプロビジョニングしたストレージ容量に応じて課金
- 時間単位のブローカーあたりの GB 数を加算した値を、月の使用時間数で割って算出

データ転送料金

- クライアントとクラスターとの間で送受信されたデータ量に対して、通常のAWSデータ転送料金が発生
- ブローカー間、ZooKeeperノードおよびブローカー間のデータ転送に料金は無料

拡張モニタリング

- 拡張モニタリングを有効化した場合には、利用量に応じた課金が Amazon CloudWatch に発生

https://aws.amazon.com/jp/msk/pricing/
https://aws.amazon.com/jp/cloudwatch/pricing/

Amazon MSK が提供する機能のまとめ

- ✓ Kafkaクラスターの作成、更新、削除の自動化
- ✓ ZooKeeperによるクラスター管理の隠蔽
- ✓ 複数AZに跨ったデプロイメントの自動最適化
- ✓ Amazon CloudWatch と連携した監視機能
- ✓ 保存データとデータ通信の暗号化
- ✓ TLSクライアント認証
- ✓ タグ付けとタグベースのIAM設定
- ✓ Kafkaクラスターの構成管理
- ✓ ストレージのスケーリング
- ✓ ブローカーのスケールアウト
- ✓ 99.9%の可用性（SLA）
- ✓ PCI や HIPAA BAA のコンプライアンス認証

現時点では出来ないこと
（今後の機能追加でサポート予定）

- クラスターのバージョンアップ
- ブローカーのスケールアップ

aws

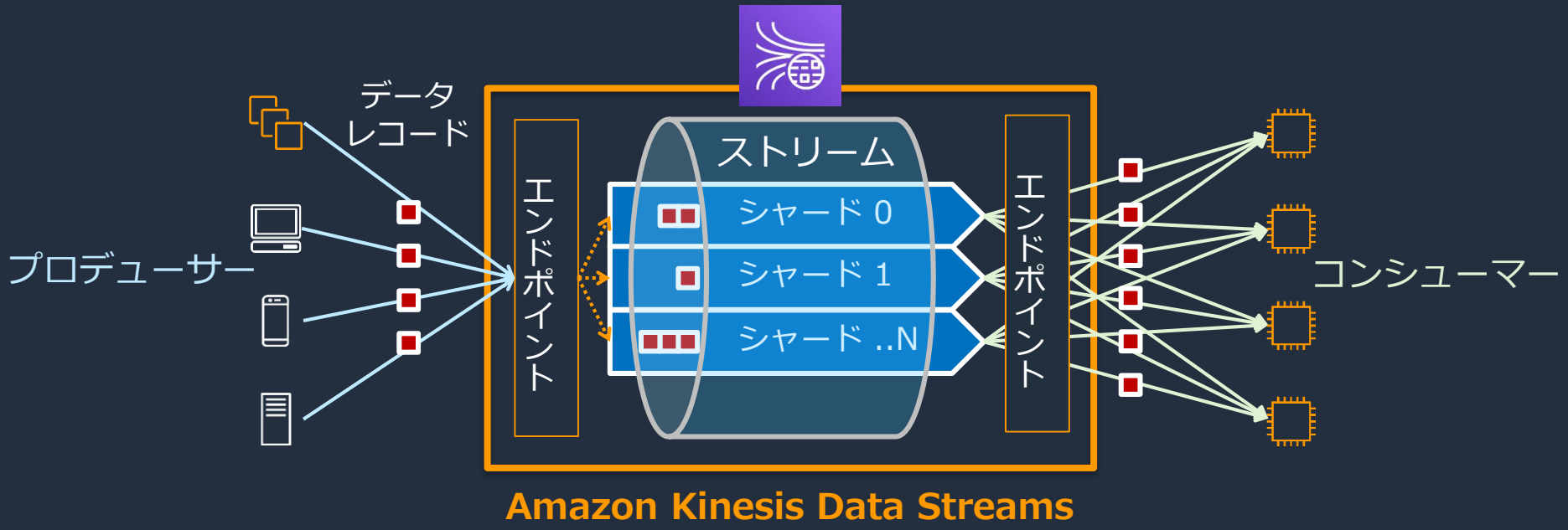
本日のアジェンダ

1. Amazon Managed Streaming for Kafka (Amazon MSK) とは
2. Apache Kafka の概要
3. Amazon MSK の機能
    - Amazon MSK を動かすまで
    - Amazon MSK として提供している機能
    - Amazon MSK の運用
4. Amazon MSK のユースケース
    - Amazon Kinesis との使い分け
    - Apache Kafka の周辺ツールとの組み合わせ
5. まとめ

aws

Amazon Kinesis Data Streams

大量のストリームをリアルタイムで収集して処理するためのデータストリーミングサービス



Amazon Kinesis Data Streams のエコシステム

プロデューサー（データ送信側）

AWS SDK

Kinesis Producer Library

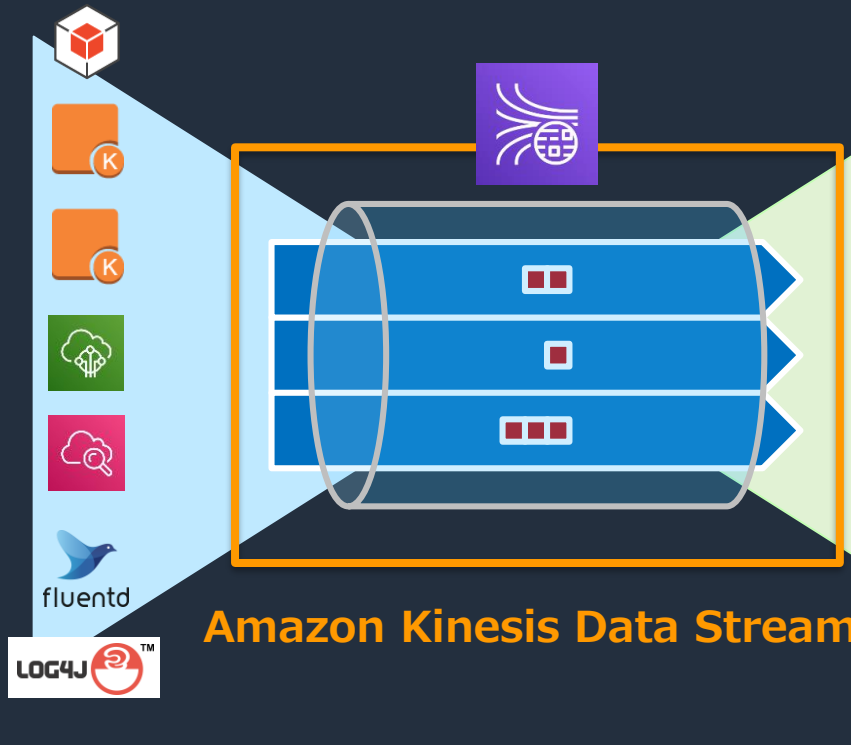
Kinesis Agent

AWS IoT Core

CloudWatch Events
CloudWatch Logs

Fluentd

Kinesis Log4j Appender
…



コンシューマー（データ処理側）

AWS SDK

Kinesis Client Library

Kinesis Data Firehose

Kinesis Data Analytics

AWS Lambda

Amazon EMR

Flink

Apache Flink
Apache Storm
…

Amazon MSK と Amazon Kinesis Data Streams の特徴

Amazon MSK

- オープンソース互換のAPIで送受信
- Kafkaのエコシステムを利用可能
- クラスターをプロビジョング
- シームレスなスケーリングが困難
- オンプレミスからの移行が容易

Amazon Kinesis Data Streams

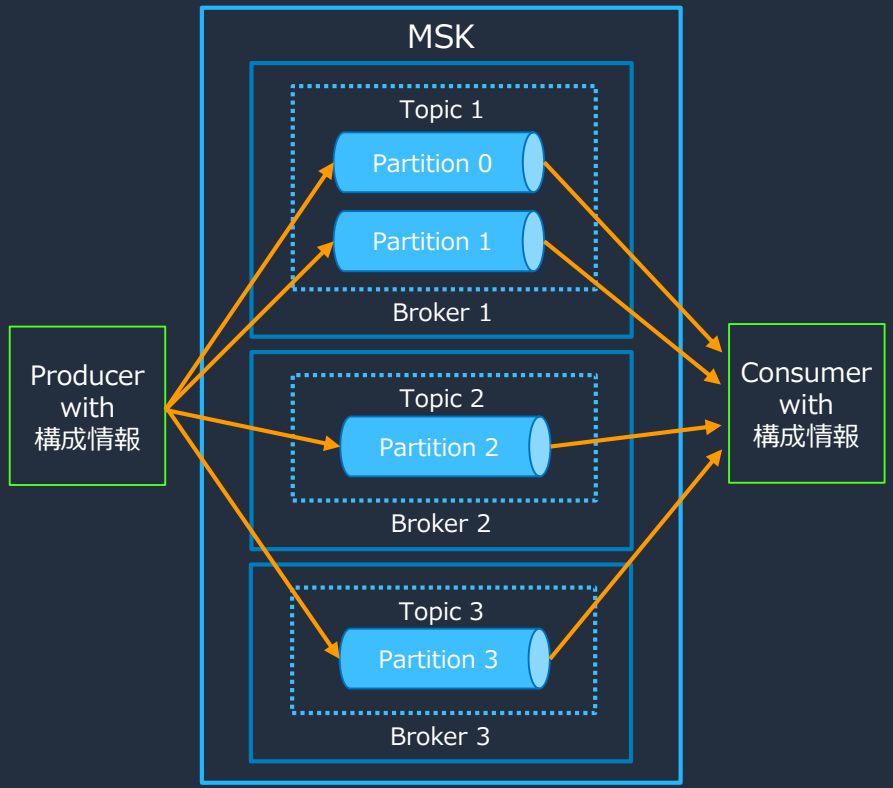
- AWSが提供するAPIで送受信
- 他のAWSサービスとの深い統合
- スループットをプロビジョニング
- シームレスなスケーリングが可能
- インフラを意識しないサーバーレス構成

用語の対応関係

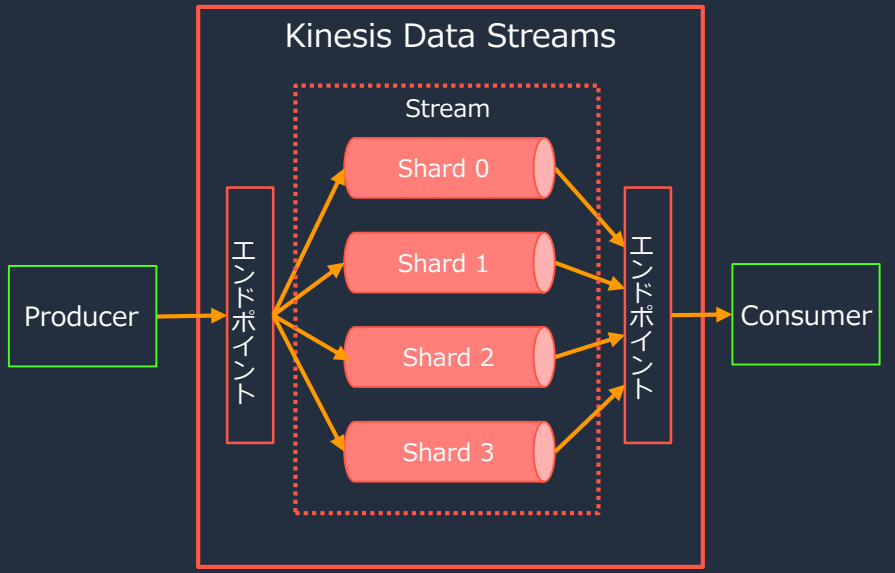
Apache Kafka (Amazon MSK)	Amazon Kinesis Data Streams
クラスター	（サーバーレスなので存在しない）
ブローカー	（サーバーレスなので存在しない）
トピック	ストリーム
パーティション	シャード
プロデューサー	プロデューサー
コンシューマー	コンシューマー
オフセット	チェックポイント
リーダーレプリカ フォロワーレプリカ ハイウォーターマーク	（内部で3AZに冗長化して隠蔽）

アーキテクチャの比較

Amazon MSK

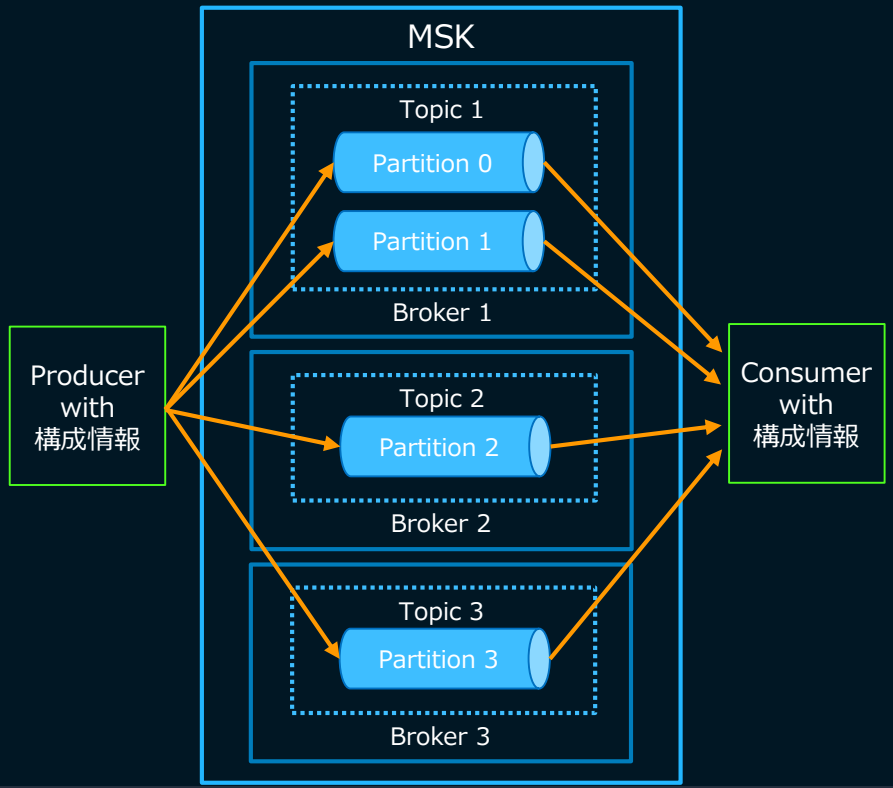


Amazon Kinesis Data Streams



アーキテクチャの比較

Amazon MSK

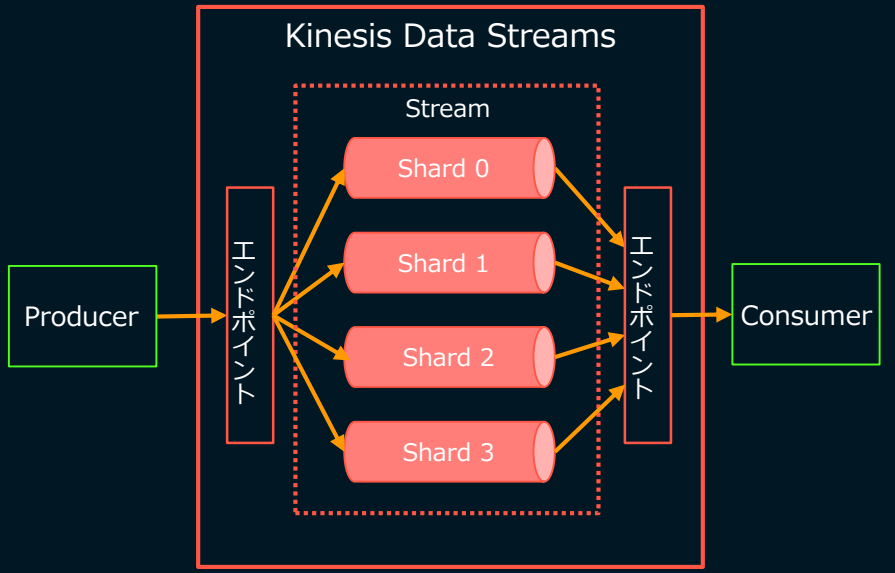


- プロデューサーとコンシューマーが接続先のブローカーを選択して通信するため、シームレスなスケールが難しい
- クラスターをプロビジョニングするためリソースの利用効率を管理する必要があるが、トピック毎にパーティション割当を管理できる

アーキテクチャの比較

- プロデューサーとコンシューマーは APIのエンドポイントと通信するため、シームレスにスケールすることができる
- シャード数を増やすことでスケールし、それ以外のリソースをプロビジョニングする必要はない（サーバーレス）

Amazon Kinesis Data Streams



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

- ✓ Kafka MirrorMaker
- ✓ Kafka Connect
- ✓ Kafka Streams

Apache Kafka の周辺ツールとフレームワーク

- ✓ Schema Registry
- ✓ Rest Proxy
- ✓ Burrow

Jarファイルのロードが必要なツール

- Confluent Control Center
- Auto Data Balancer (Confluent)
- uReplicator (Uber)
- Cruise Control (LinkedIn)

aws

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

- ✓ Kafka MirrorMaker
- ✓ Kafka Connect
- ✓ Kafka Streams

Apache Kafka の周辺ツールとフレームワーク

- ✓ Schema Registry
- ✓ Rest Proxy
- ✓ Burrow

Jarファイルのロードが必要なツール

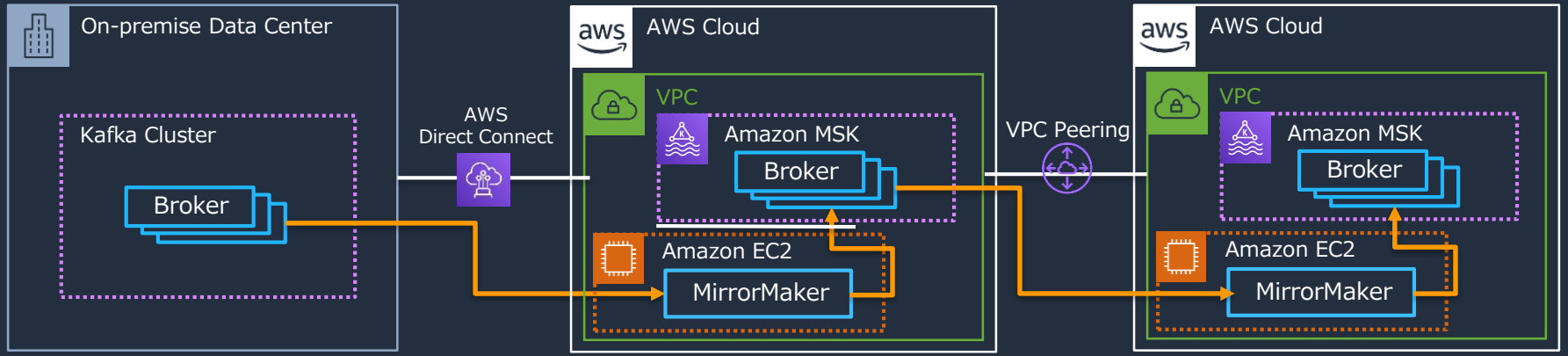
- Confluent Control Center
- Auto Data Balancer (Confluent)
- uReplicator (Uber)
- Cruise Control (LinkedIn)

aws

Kafka MirrorMaker

Kafkaのクラスター間をレプリーケーションするツール

- MirrorMaker に、ソースのクラスターを読み取るコンシューマーとしての設定と、ターゲットのクラスターに書き込むプロデューサーとしての設定を入力
- レプリケーションに問題が発生した場合に備えて、MirrorMaker はターゲット側で動かす
- Amazon MSK のドキュメントにある **“MirrorMaker 1.0 Best Practices”** も参照

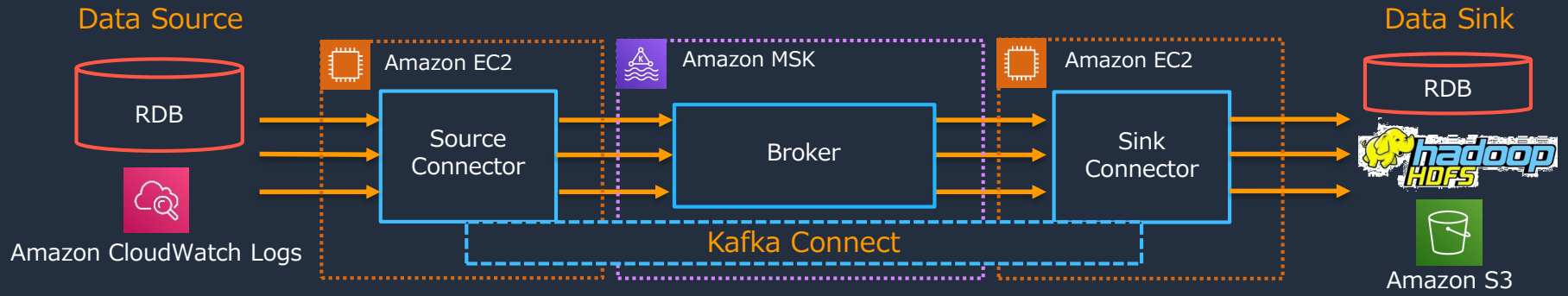


https://kafka.apache.org/documentation/#basic_ops_mirror_maker
https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/migration.html

Kafka Connect

Kafkaと周辺のシステムを接続するためのフレームワーク

- Kafka のブローカーと接続する部分を Connector と呼ぶ
  - ✓ プロデューサー側を Source Connector、コンシューマー側を Sink Connector と呼ぶ
- 様々なプラグインが公開されており、接続先に合ったプラグインを実装せずに利用可能
  - ✓ Source の例：RDB (JDBC)、Amazon CloudWatch Logs、Amazon DynamoDB …
  - ✓ Sink の例：RDB (JDBC) 、HDFS、Elasticsearch、Amazon S3、AWS Lambda …

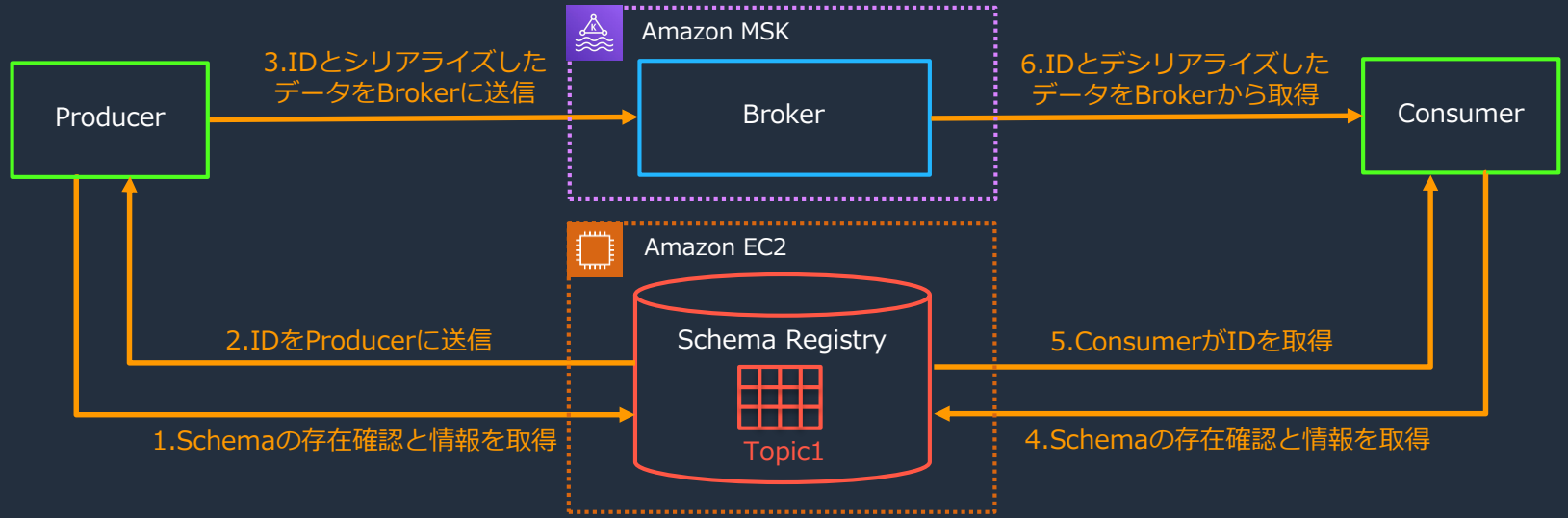


https://kafka.apache.org/documentation/#connect
https://docs.confluent.io/current/connect/index.html

Schema Registry

メッセージに対して一元化されたスキーマ管理とシリアライズ機構を提供

- Avro Schemaでスキーマを定義し、REST APIでスキーマを登録／取得
- スキーマに応じた自動的なシリアライゼーションを提供
- スキーマをバージョン管理し、変更時の変化にも対応



https://docs.confluent.io/current/schema-registry/index.html

Apache Kafka のユースケース

1. **メッセージング**
   - ✓ ActiveMQ や RabbitMQ などの代替として、メッセージブローカーとして利用
2. **ウェブサイトのアクティビティ追跡**
   - ✓ サイトアクティビティからユーザーのアクティビティ追跡パイプラインを構築
3. **メトリクス**
   - ✓ 分散したアプリケーションから運用監視データを集約して集計や統計処理を実施
4. **ログの集約**
   - ✓ ログのイベントデータをメッセージのストリームとして利用
5. **ストリーム処理**
   - ✓ 複数のステップで構成されるパイプラインでストリームデータを処理
   - ✓ Kafka Streams や Apache Flink などのストリーム処理ツールと合わせて利用
6. **イベントソーシング**
   - ✓ 状態の変化が時系列なレコードのシーケンスとして記録されるアプリケーションのバックエンドとして利用
7. **コミットログ**
   - ✓ Apache BookKeeper のような、分散システムの外部コミットログとして利用

https://kafka.apache.org/uses

aws

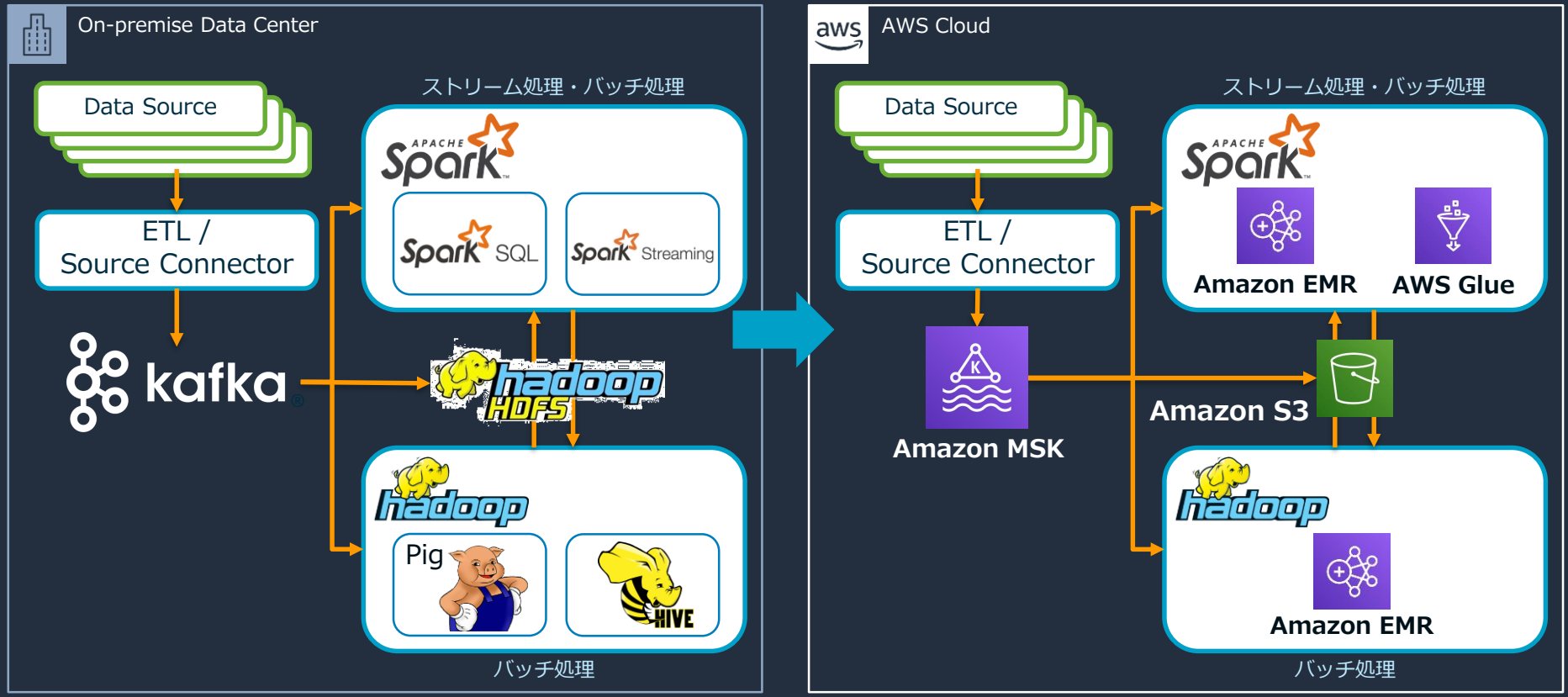
Amazon MSK のユースケース

1. **オンプレミスにある既存の Apache Kafka クラスターの移行**
2. **Amazon EC2 上に構築されている Apache Kafka クラスターの移行**
3. **Apache Kafka の周辺ツールを利用している場合、または利用したい場合**

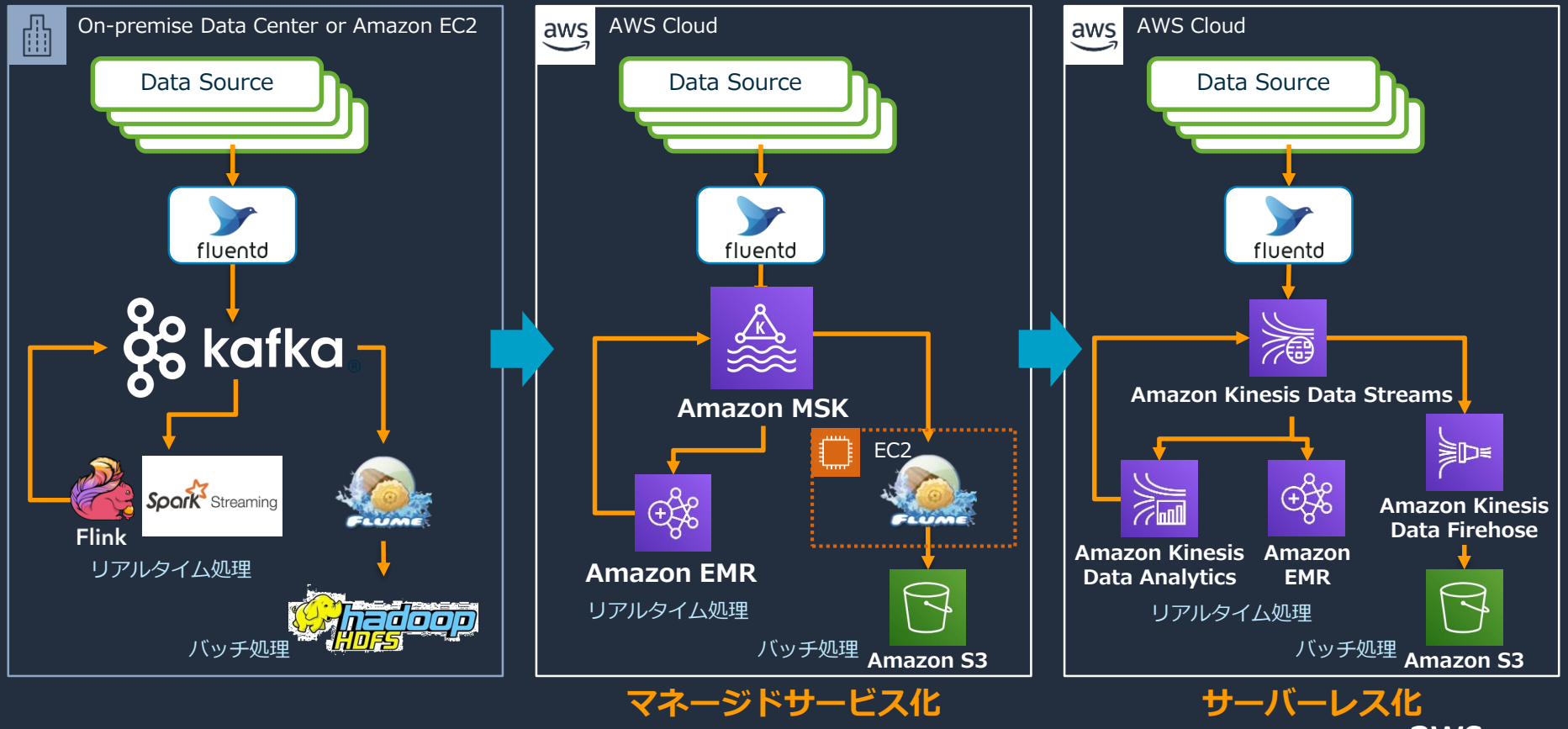
それ以外の場合には、基本的に Amazon Kinesis Data Streams の利用がお勧め

- ✓ サーバーレスのメリットが得られるため、多くの場合ではスケーラビリティとコストの観点でメリットがある
- ✓ AWS Lambda など他のAWSサービスとの連携機能を活用して、プロデューサーとコンシューマーを含めてサーバレスなアーキテクチャを設計できる

オンプレミス上のストリーム処理+バッチ処理の移行



ストリーム・パイプラインのアーキテクチャと移行プラン



本日のアジェンダ

1. Amazon Managed Streaming for Kafka (Amazon MSK) とは
2. Apache Kafka の概要
3. Amazon MSK の機能
    - Amazon MSK を動かすまで
    - Amazon MSK として提供している機能
    - Amazon MSK の運用
4. Amazon MSK のユースケース
    - Amazon Kinesis との使い分け
    - Apache Kafka の周辺ツールとの組み合わせ
5. まとめ

aws

まとめ

- **Amazon MSK は、フルマネージドで可用性が高くセキュアな Apache Kafka サービス**
  - ✓ Kafkaクラスターの作成、更新、削除をはじめ、クラスターの管理機能を提供
  - ✓ 複数AZに自動的にデプロイして可用性を確保し、障害復旧の自動化も内包
  - ✓ データプレーンの操作は、Apache Kafka の API をそのまま使用可能
  - ✓ Kafka のエコシステムの大部分とも互換性を保持
- **Amazon MSK では現在は未サポートの機能もあるが、今後も拡張予定**
  - ✓ クラスターのアップグレード
  - ✓ ブローカーインスタンスのスケールアップ
- **Amazon Kinesis と上手く使い分けることをお勧め**
  - ✓ Amazon MSK の主なユースケースは、既存のKafkaクラスターの移行
  - ✓ Amazon MSK はフルマネージドではあるが、サーバーレスではない

aws

Q&A

お答えできなかったご質問については
AWS Japan Blog 「https://aws.amazon.com/jp/blogs/news/」にて
後日掲載します。

AWS の日本語資料の場所「AWS 資料」で検索

https://amzn.to/JPArchive

AWS Well-Architected 個別技術相談会

毎週”W-A個別技術相談会”を実施中

- AWSのソリューションアーキテクト(SA)に対策などを相談することも可能
- 申込みはイベント告知サイトから

(https://aws.amazon.com/jp/about-aws/events/)

AWS イベント で[検索]

ご視聴ありがとうございました

AWS 公式 Webinar
https://amzn.to/JPWebina

過去資料
https://amzn.to/JPArchive