aws

デベロッパーガイド

Amazon Managed Streaming for Apache Kafka

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Managed Streaming for Apache Kafka: デベロッパーガイド

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは、Amazon 以外の製品またはサービスに関連して、顧客間で混乱を引き起こす可能性のある方法で、または Amazon を軽蔑または信用を傷つける方法で使用することはできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

ようこそ 1
Amazon MSK とは 1
設定 3
にサインアップする AWS 3
ライブラリとツールをダウンロードする 3
MSK プロビジョニング済み 5
はじめに 5
クラスターを作成する 6
IAM ロールを作成する 7
クライアントマシンを作成する 11
トピックの作成 12
データを生成および消費する 17
メトリクスの表示 18
チュートリアルのリソースを削除する 19
仕組み 19
プロビジョニングされたクラスターを管理する 20
クラスターを作成する 21
クラスターを一覧表示する 31
MSK プロビジョンドクラスターに接続する 33
ブートストラップブローカーを取得する 54
クラスターをモニタリングする 56
クラスターセキュリティを更新する 103
クラスターを展開する 106
ブローカーを削除する 109
クラスターブローカーのサイズを更新する 114
Cruise Control を使用する 117
クラスター設定を更新する 122
Amazon MSK クラスターのブローカーを再起動する 126
クラスターにタグを付ける 128
Amazon MSK クラスターへの移行 130
クラスターを削除 134
主な機能と概念 136
ブローカータイプ 136
ブローカーサイズ 140

ストレージ管理 141
セキュリティ 160
ブローカー設定 230
パッチ適用 293
ブローカーのオフラインおよびクライアントフェイルオーバー 295
Amazon MSK ログ記録 298
メタデータ管理 305
リソース 309
Apache Kafka バージョン 309
Amazon MSK クラスターをトラブルシューティングする 323
ベストプラクティス 333
標準ブローカーのベストプラクティス 333
Express ブローカーのベストプラクティス 342
Apache Kafka クライアントのベストプラクティス 347
MSK サーバーレス 354
MSK サーバーレスクラスターを使用する 355
クラスターを作成する 356
IAM ロールを作成する 357
クライアントマシンを作成するには 359
トピックの作成 361
データの生成と消費 363
リソースの削除 364
設定 365
モニタリング 366
MSK Connect 369
Amazon MSK Connect のメリット 369
入門 371
MSK Connect に必要なリソースを設定する 371
カスタムプラグインを作成する 376
クライアントマシンと Apache Kafka トピックを作成する 377
コネクタの作成 380
MSK クラスターにデータを送信する 381
コネクタを理解する 381
コネクタ容量を理解する 382
コネクタを作成する 383
コネクタを更新する 384

コネクタからの接続 ... 385
カスタムプラグインの作成 ... 385
MSK Connect ワーカーを理解する ... 386
デフォルトのワーカー構成 ... 387
サポートされているワーカー設定プロパティ ... 387
カスタム設定の作成 ... 389
コネクタオフセットの管理 ... 389
設定プロバイダー ... 393
考慮事項 ... 393
カスタムプラグインを作成して S3 にアップロードする ... 394
さまざまなプロバイダーのパラメータとアクセス許可を設定する ... 395
カスタムワーカー設定の作成 ... 399
コネクタを作成する ... 400
IAM ロールとポリシー ... 401
サービス実行ロールを理解する ... 402
ポリシーの例 ... 405
サービス間での混乱した代理問題を防止する ... 407
AWS 管理ポリシー ... 408
サービスリンクロールを使用する ... 412
インターネットアクセスを有効にする ... 414
NAT ゲートウェイの設定 ... 414
プライベート DNS ホスト名を理解する ... 417
VPC DHCP オプションを設定する ... 417
DNS 属性を設定する ... 418
コネクタ作成の失敗を処理する ... 418
セキュリティ ... 419
ロギング ... 419
シークレットがコネクタログに表示されないようにする ... 420
MSK Connect のモニタリング ... 421
例 ... 424
Amazon S3 シンクコネクタを設定する ... 424
EventBridge Kafka シンクコネクタをセットアップする ... 426
Debezium ソースコネクタを使用する ... 432
Amazon MSK Connect への移行 ... 443
Kafka Connect で使用される内部トピックを理解する ... 443
状態の管理 ... 444

ソースコネクタの移行 445
シンクコネクタの移行 446
トラブルシューティング 447
MSK レプリケーター 448
Amazon MSK Replicator の仕組み 449
データレプリケーション 449
メタデータレプリケーション 450
トピック名の設定 452
ソースクラスターとターゲットクラスターを設定する 453
Amazon MSK ソースクラスターを準備する 453
Amazon MSK ターゲットクラスターを準備する 456
チュートリアル: Amazon MSK Replicator を作成する 456
Amazon MSK Replicator を作成する際の考慮事項 457
AWS コンソールでレプリケーターを作成する 461
MSK レプリケータ設定の編集 469
MSK レプリケーターの削除 470
レプリケーションのモニタリング 471
MSK レプリケーターのメトリクス 471
レプリケーションを使用して耐障害性を高める 482
マルチリージョン Apache Kafka アプリケーションを構築する際の考慮事項 482
アクティブ/アクティブとアクティブ/パッシブのクラストポロジの使用 482
アクティブ/パッシブ Kafka クラスターの作成 483
セカンダリリージョンへのフェイルオーバー 483
計画的なフェイルオーバーの実行 484
計画外のフェイルオーバーの実行 485
フェイルバックを実行する 487
アクティブ/アクティブ設定の作成 489
1 つの Amazon MSK クラスターから別の Amazon MSK クラスターに移行する 490
セルフマネージド型の MirrorMaker2 から MSK Replicator に移行する 491
MSK Replicator のトラブルシューティング 491
MSK レプリケーターの状態が CREATING から FAILED に変わります。 491
MSK レプリケーターが CREATING 状態でスタックしているように見える 492
MSK レプリケーターがデータをレプリケートしていない、または一部のデータしかレプリケートしていない 492
メッセージのオフセットがターゲットクラスターとソースクラスターとで異なる 493

MSK Replicator がコンシューマーグループのオフセットを同期していない、またはコンシューマーグループがターゲットクラスターに存在しない ... 494
レプリケーションのレイテンシーが高い、または増加し続けている ... 495
ReplicatorFailure メトリクスの使用 ... 496
MSK レプリケーターの使用に関するベストプラクティス ... 502
Kafka クォータを使用した MSK レプリケータースループットの管理 ... 502
クラスターの保持期間の設定 ... 504
MSK の統合 ... 505
Amazon MSK 用の Athena コネクタ ... 505
Amazon MSK の Redshift 統合 ... 505
Amazon MSK の Firehose 統合 ... 506
EventBridge Pipes にアクセスする ... 506
Express ブローカーと MSK Serverless を備えた Kafka Streams ... 508
Kafka Streams アプリケーションの作成 ... 509
リアルタイムベクトル埋め込みブループリント ... 512
ログ記録とオブザーバビリティ ... 513
リアルタイムベクトル埋め込みブループリントを有効にする前の注意事項 ... 514
ストリーミングデータベクトル化の設計図をデプロイする ... 515
クォータ ... 519
Amazon MSK のクォータ引き上げのリクエスト ... 519
標準ブローカークォータ ... 520
Express ブローカークォータ ... 522
ブローカーサイズ別の高速ブローカースループットスロットル制限 ... 525
Express ブローカーパーティションクォータ ... 526
MSK レプリケータークォータ ... 526
サーバーレスクラスターのクォータ ... 527
MSK Connect クォータ ... 529
ドキュメント履歴 ... 530
... dxlvii

Amazon MSK デベロッパーガイドへようこそ

Amazon Managed Streaming for Apache Kafka デベロッパーガイドへようこそ。このガイドの使用を開始するにあたり、実行しようとしている内容に基づいて次のトピックを参考にしてください。

- Amazon MSK の使用を開始する チュートリアルに従って MSK プロビジョンドクラスターを作成します。
- でプロビジョニングされた MSK の機能について詳しく説明しますMSK プロビジョニングとは。
- MSK Serverless でクラスター容量を管理およびスケーリングすることなく、Apache Kafka を実行します。
- MSK Connect を使用して、Apache Kafka クラスターとの間でデータをストリーミングします。
- MSK レプリケーターを使用して、異なる または同じ の MSK プロビジョンドクラスター間でデータを確実にレプリケートします AWS リージョン。

ハイライト、製品の詳細、価格情報については、Amazon MSK のサービスページを参照してください。

Amazon MSK とは

Amazon Managed Streaming for Apache Kafka (Amazon MSK) は、Apache Kafka を使ってストリーミングデータを処理するアプリケーションの構築および実行を可能にする、フルマネージドサービスです。Amazon MSK は、クラスターの作成、更新、削除などに用いられるコントロールプレーンオペレーションを提供します。データの生成と消費などの、Apache Kafka データプレーンオペレーションを使用することができます。これは、Apache Kafka のオープンソースバージョンを実行します。つまり、パートナーや Apache Kafka コミュニティの既存のアプリケーション、ツール、プラグインが、アプリケーションコードを変更することなくサポートされます。Amazon MSK を使用して、「the section called “サポート対象の Apache Kafka バージョン”」にリストされている任意の Apache Kafka バージョンを使用するクラスターを作成できます。

これらのコンポーネントでは、Amazon MSK のアーキテクチャについて説明します。

- ブローカーノード — Amazon MSK クラスターを作成するときに、Amazon MSK が各アベイラビリティーゾーンに作成するブローカーノードの数を指定します。アベイラビリティーゾーンごとに 1 つのブローカーが最小です。各アベイラビリティーゾーンには、独自の仮想プライベートクラウド (VPC) サブネットがあります。

Amazon MSK Provisioned には、 Amazon MSK Standard ブローカーと の 2 つのブローカータイプがありますAmazon MSK Express ブローカー。MSK Serverless では、MSK がトラフィックの処理に使用されるブローカーノードを管理し、クラスターレベルでのみ Kafka サーバーリソースをプロビジョニングします。

- ZooKeeper ノード – Amazon MSK は、Apache ZooKeeper ノードも作成します。Apache ZooKeeper は、信頼性の高い分散調整を可能にするオープンソースのサーバーです。
- KRaft コントローラー — Apache Kafka コミュニティでは、Apache ZooKeeper に代わって Apache Kafka クラスターのメタデータ管理に使用できる KRaft が開発されました。KRaft モードでは、クラスターメタデータは ZooKeeper ノードではなく Kafka クラスターの一部である Kafka コントローラーのグループ内で伝播されます。KRaft コントローラーは追加料金なしで含まれており、追加のセットアップや管理は必要ありません。
- プロデューサー、コンシューマー、およびトピック作成者 – Amazon MSK では、トピックの作成とデータの生成および消費のために、Apache Kafka データプレーンオペレーションを使用できます。
- クラスターオペレーション SDK の AWS Management Console AWS Command Line Interface 、(AWS CLI) 、または APIs を使用して、コントロールプレーンオペレーションを実行できます。例えば、Amazon MSK クラスターを作成または削除したり、アカウント内のすべてのクラスターを一覧表示したり、クラスターのプロパティを表示したり、クラスター内のブローカーの数とタイプの更新を行うことができます。

Amazon MSK は、クラスターの最も一般的な障害シナリオを検出して自動的に回復します。これにより、プロデューサーおよびコンシューマーアプリケーションは、影響を最小限に抑えながら書き込みおよび読み取り操作を継続できます。Amazon MSK は、ブローカー障害を検出すると、障害を軽減するか、異常または到達不能なブローカーを新しいブローカーに置き換えます。また、可能な場合は、Apache Kafka が複製する必要があるデータを減らすために、古いブローカーからストレージを再利用します。可用性が影響を受けるのは、Amazon MSK が検出と回復に必要とする時間に限定されます。回復後、プロデューサーとコンシューマーのアプリは、障害発生前に使用したものと同じブローカー IP アドレスと通信し続けることができます。

Amazon MSK のセットアップ

Amazon MSK を初めて使用する前に、以下のタスクを完了してください。

タスク

- にサインアップする AWS
- ライブラリとツールをダウンロードする

にサインアップする AWS

にサインアップすると AWS、Amazon MSK AWSを含む のすべてのサービスに Amazon Web Services アカウントが自動的にサインアップされます。料金は、使用するサービスの料金のみが請求されます。

AWS アカウントがすでにある場合は、次のタスクに進んでください。 AWS アカウントをお持ちでない場合は、以下の手順に従ってアカウントを作成してください。

Amazon Web Services アカウントにサインアップするには

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、通話呼び出しまたはテキストメッセージを受け取り、電話キーパッドで検証コードを入力するように求められます。

   にサインアップすると AWS アカウント、 AWS アカウントのルートユーザー が作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、ルートユーザーアクセスが必要なタスクの実行にはルートユーザーのみを使用するようにしてください。

ライブラリとツールをダウンロードする

以下のライブラリとツールは Amazon MSK での作業に役立ちます。

- AWS Command Line Interface (AWS CLI)は Amazon MSK をサポートしています。 AWS CLI を使用すると、コマンドラインから複数の Amazon Web Services を制御し、スクリプトを使用して自

動化できます。 AWS CLI を最新バージョンにアップグレードして、このユーザーガイドに記載されている Amazon MSK 機能がサポートされていることを確認します。 AWS CLIをアップグレードする方法の詳細については、「AWS Command Line Interfaceのインストール」を参照してください。をインストールしたら AWS CLI、設定する必要があります。の設定方法については AWS CLI、「aws configure」を参照してください。

- Amazon Managed Streaming for Kafka API Reference でAmazon MSK がサポートする API オペレーションについて確認できます。
- Go、Java、JavaScript、.NET、Node.js、PHP、Python および Ruby 用の Amazon Web Services SDK には Amazon MSK のサポートとサンプルが含まれてます。

MSK プロビジョニングとは

Amazon MSK プロビジョンドクラスターには、クラスターのパフォーマンスを最適化し、ストリーミングニーズを満たすのに役立つさまざまな機能が用意されています。以下のトピックでは、機能について詳しく説明します。

MSK Provisioned は、Apache Kafka クラスターを手動で設定およびスケーリングできる MSK クラスターデプロイオプションです。これにより、Apache Kafka 環境を強化するインフラストラクチャをさまざまなレベルで制御できます。MSK Provisioned では、Kafka クラスターを構成するインスタンスタイプ、ストレージボリューム (標準ブローカー) 、およびブローカーノードの数を選択できます。データ処理のニーズの変化に応じてブローカーを追加または削除することで、クラスターをスケールすることもできます。この柔軟性により、スループット、保持容量、またはその他のパフォーマンス特性を最大化するかどうかにかかわらず、特定のワークロード要件に合わせてクラスターを最適化できます。MSK Provisioned は、インフラストラクチャ設定オプションに加えて、エンタープライズグレードのセキュリティ、モニタリング、運用上の利点を提供します。これには、Apache Kafka のバージョンアップグレード、暗号化とアクセスコントロールによる組み込みセキュリティ、モニタリング用の Amazon CloudWatch AWS のサービス などの他の との統合などの機能が含まれます。MSK Provisioned には、Standard と Express の 2 つの主要なブローカータイプがあります。

MSK プロビジョンド API の詳細については、「Amazon MSK API リファレンス」を参照してください。

Amazon MSK の使用を開始する

このチュートリアルでは、MSK クラスターを作成し、データを生成および消費し、メトリクスを使用してクラスターのヘルスをモニタリングする方法の例を示します。この例は、MSK クラスターを作成するときに選択できるすべてのオプションを表しているわけではありません。このチュートリアルのさまざまな部分では、簡単にするためにデフォルトのオプションを選択します。これは、MSK クラスターまたはクライアント インスタンスを設定するために機能する唯一のオプションであることを意味するものではありません。

トピック

- ステップ 1: MSK プロビジョンドクラスターを作成する
- ステップ 2: Amazon MSK クラスターでトピックを作成するためのアクセス権を付与する IAM ロールを作成する
- ステップ 3: クライアントマシンを作成する

- ステップ 4: Amazon MSK クラスターにトピックを作成する
- ステップ 5: データを生成および消費する
- ステップ 6: Amazon CloudWatch を使用して Amazon MSK メトリクスを表示する
- ステップ 7: このチュートリアル用に作成された AWS リソースを削除する

ステップ 1: MSK プロビジョンドクラスターを作成する

Amazon MSK の使用を開始するこのステップでは、Amazon MSK プロビジョンドクラスターを作成します。このクラスター AWS Management Console を作成するには、 のクイック作成オプションを使用します。

を使用して Amazon MSK クラスターを作成するには AWS Management Console

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. [クラスターを作成] を選択します。
3. [作成方法] では、[クイック作成] オプションを選択したままにします。[クイック作成] オプションを使用すると、デフォルト設定でクラスターを作成できます。
4. [クラスター名] には、クラスターのわかりやすい名前を入力します。例えば、**`MSKTutorialCluster`**。
5. 一般的なクラスタープロパティの場合は、次の操作を行います。
   a. [クラスタータイプ] に [プロビジョンド] を選択します。
   b. ブローカーで実行する Apache Kafka バージョンを選択します。比較テーブルを表示するには、バージョン互換性の表示を選択します。
   c. ブローカータイプで、スタンダードブローカーまたは Express ブローカーを選択します。
   d. ブローカーサイズを選択します。
6. このチュートリアルの後半で必要になるため、[すべてのクラスター設定] の下の表から次の設定の値をコピーして保存します。
   - VPC
   - サブネット
   - VPC に関連付けられたセキュリティグループ。
7. [クラスターを作成] を選択します。

8. [クラスターの概要] ページでクラスターの [ステータス] を確認します。Amazon MSK がクラスターをプロビジョニングすると、ステータスが [作成中] から [アクティブ] に変わります。ステータスが [アクティブ] の場合、クラスターに接続できます。クラスターのステータスの詳細については、「MSK プロビジョンドクラスターの状態を理解する」を参照してください。

次のステップ

ステップ 2: Amazon MSK クラスターでトピックを作成するためのアクセス権を付与する IAM ロールを作成する

ステップ 2: Amazon MSK クラスターでトピックを作成するためのアクセス権を付与する IAM ロールを作成する

このステップでは、2 つのタスクを実行します。最初のタスクは、クラスターでトピックを作成し、それらのトピックにデータを送信するためのアクセスを許可する IAM ポリシーを作成することです。2 番目のタスクは、IAM ロールを作成し、作成したポリシーをそのロールに関連付けることです。後のステップでは、このロールを引き受けるクライアントマシンを作成し、それを使用してクラスター上にトピックを作成し、そのトピックにデータを送信します。

トピックを作成し、書き込むことを可能にする IAM ポリシーを作成する

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで [Policies] (ポリシー) を選択します。
3. [Create policy] (ポリシーの作成) を選択します。
4. ポリシーエディタで JSON を選択し、エディタウィンドウの JSON を次の JSON に置き換えます。

   次の例では、以下を置き換えます。

   - *region* と、クラスターを作成した AWS リージョン のコード。
   - アカウント ID の例: *123456789012*。 AWS アカウント
   - *MSKTutorialCluster* および *MSKTutorialCluster/7d7131e1-25c5-4e9a-9ac5-ea85bee4da11-14*、クラスターの名前とその ID。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:Connect",
                "kafka-cluster:AlterCluster",
                "kafka-cluster:DescribeCluster"
            ],
            "Resource": [
                "arn:aws:kafka:us-
east-1:123456789012:cluster/MSKTutorialCluster/7d7131e1-25c5-4e9a-9ac5-
ea85bee4da11-14"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:*Topic*",
                "kafka-cluster:WriteData",
                "kafka-cluster:ReadData"
            ],
            "Resource": [
            "arn:aws:kafka:us-east-1:123456789012:topic/MSKTutorialCluster/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:AlterGroup",
                "kafka-cluster:DescribeGroup"
            ],
            "Resource": [
            "arn:aws:kafka:us-east-1:123456789012:group/MSKTutorialCluster/*"
            ]
        }
    ]
```

```
}
```

セキュアポリシーを作成する方法については、「」を参照してくださいthe section called “IAM アクセスコントロール”。

5. [次へ] を選択します。
6. [レビューと作成] ページで、以下の操作を実行します。

   a. ポリシー名には、 などのわかりやすい名前を入力します**msk-tutorial-policy**。

   b. このポリシーで定義されているアクセス許可で、ポリシーで定義されているアクセス許可を確認および/または編集します。

   c. （ オプション) ポリシーを識別、整理、または検索しやすくするには、新しいタグを追加 を選択して、キーと値のペアとしてタグを追加します。たとえば、 **Environment**と のキーと値のペアを使用してポリシーにタグを追加します**Test**。

      タグの使用の詳細については、IAM ユーザーガイドの「 AWS Identity and Access Management リソースのタグ」を参照してください。

7. [Create policy] (ポリシーの作成) を選択します。

IAM ロールを作成し、ポリシーを適用する

1. ナビゲーションペインで、ロールを選択し、ロールの作成を選択します。
2. [信頼されたエンティティを選択] ページで、以下の操作を実行してください：

   a. 信頼できるエンティティタイプ で、AWS のサービス を選択します。

   b. [サービスまたはユースケース] で、 [EC2] を選択します。

   c. [ユースケース] で、[EC2] を選択してください。

3. [次へ] を選択します。
4. [アクセス許可を追加] ページで、以下を実行します。

   a. アクセス許可ポリシーの検索ボックスに、このチュートリアル用に以前に作成したポリシーの名前を入力します。次に、ポリシー名の左側にあるボックスを選択します。

   b. (オプション) アクセス許可の境界を設定します。このアドバンスド機能は、サービスロールで使用できますが、サービスにリンクされたロールではありません。アクセス許可の境界の設定については、IAM ユーザーガイドの「ロールの作成とポリシーのアタッチ (コンソール) 」を参照してください。

5. [次へ] を選択します。
6. [名前を付けて、レビューし、作成する] ページで、以下の操作を実行します。
   a. ロール名には、 などのわかりやすい名前を入力します**msk-tutorial-role**。

      > **Important**
      >
      > ロールに名前を付けるときは、次のことに注意してください。
      >
      > - ロール名は 内で一意である必要があり AWS アカウント、大文字と小文字を区別することはできません。
      >
      >   例えば、**PRODROLE** と **prodrole** の両方の名前でロールを作成することはできません。ロール名がポリシーまたは ARN の一部として使用される場合、ロール名は大文字と小文字が区別されます。ただし、サインインプロセスなど、コンソールにロール名がユーザーに表示される場合、ロール名は大文字と小文字が区別されません。
      > - 他のエンティティがロールを参照する可能性があるため、ロールを作成した後にロール名を編集することはできません。

   b. (オプション) [説明] にロールの説明を入力します。
   c. (オプション) ロールのユースケースとアクセス許可を編集するには、「ステップ 1: 信頼できるエンティティを選択する」または「ステップ 2: アクセス許可を追加する」セクションで、「編集」を選択します。
   d. (オプション) ロールを識別、整理、または検索するには、新しいタグを追加 を選択して、キーと値のペアとしてタグを追加します。たとえば、 **ProductManager**と のキーと値のペアを使用して、ロールにタグを追加します**John**。

      タグの使用の詳細については、IAM ユーザーガイドの「 AWS Identity and Access Management リソースのタグ」を参照してください。
7. ロールを確認したら、[ロールを作成] を選択します。

次のステップ

ステップ 3: クライアントマシンを作成する

ステップ 3: クライアントマシンを作成する

「Amazon MSK の使用を開始する」のこのステップでは、クライアントマシンを作成します。このクライアントマシンを使用して、データを生成および消費するトピックを作成します。簡単にするために、このクライアントマシンを MSK クラスターに関連付けられた VPC に作成します。これにより、クライアントがクラスターに簡単に接続できるようになります。

クライアントマシンを作成するには

1. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。
2. Amazon EC2 コンソールダッシュボードで、[インスタンスを起動] を選択してください。
3. 名前とタグに、名前にクライアントマシンのわかりやすい名前を入力して、簡単に追跡できるようにします。例えば、**MSKTutorialClient**。
4. アプリケーションイメージと OS イメージ (Amazon マシンイメージ) の Amazon マシンイメージ (AMI) で、Amazon Linux 2 AMI (HVM) - カーネル 5.10、SSD ボリュームタイプを選択します。
5. インスタンスタイプの場合、t2.micro のデフォルト選択のままにします。
6. キーペア (ログイン) で、既存のキーペアを選択するか、新しいキーペアを作成します。インスタンスへの接続にキーペアを必要としない場合は、キーペアなしで続行 (非推奨) を選択できます。

   新しいキーペアを作成するには、以下の操作を行います。

   a. 新しいキーペアの作成 を選択します。
   b. [Key Pair Name] (キーペア名) に **MSKKeyPair** と入力します。
   c. キーペアタイプとプライベートキーファイル形式については、デフォルトの選択のままにします。
   d. [キーペアの作成] を選択してください。

   既存のキーペアを使用することもできます。
7. ページを下にスクロールして詳細セクションを展開し、次の操作を行います。

   - IAM インスタンスプロファイルで、クライアントマシンが引き受ける IAM ロールを選択します。

     IAM ロールがない場合は、次の操作を行います。

i. 新しい IAM プロファイルの作成 を選択します。

ii. ステップ 2: IAM ロールを作成する の手順を実行します。

8. [インスタンスを起動] を選択します。
9. [インスタンスの表示] を選択します。次に、[セキュリティグループ] 列で、新しいインスタンスに関連付けられているセキュリティグループを選択します。セキュリティグループの ID をコピーし、後で使用できるように保存します。
10. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
11. ナビゲーションペインで、[Security Groups] (セキュリティグループ) をクリックします。「the section called “クラスターを作成する”」で ID を保存したセキュリティグループを見つけます。
12. Inbound Rules (インバウンドルール) タブで、Edit inbound rules (インバウンドルールの編集) を選択します。
13. [ルールを追加] を選択してください。
14. 新しいルールで、Type (タイプ) 列のAll traffic (すべてのトラフィック) を選択します。[ソース] 列の 2 番目のフィールドで、クライアントマシンのセキュリティグループを選択します。これは、クライアントマシンインスタンスを起動した後に名前を保存したグループです。
15. [Save Rules] (ルールの保存) を選択してください。これで、クラスターのセキュリティグループは、クライアントマシンのセキュリティグループからのトラフィックを受け入れることができます。

次のステップ

ステップ 4: Amazon MSK クラスターにトピックを作成する

ステップ 4: Amazon MSK クラスターにトピックを作成する

Amazon MSK の使用をスタートする のこのステップでは、Apache Kafka クライアントライブラリとツールをクライアントマシンにインストールしてから、トピックを作成します。

> **Warning**
>
> このチュートリアルで使用している Apache Kafka のバージョン番号は例にすぎません。MSK クラスターバージョンと同じバージョンのクライアントを使用することが推奨されます。古いクライアントバージョンでは、特定の機能や重大なバグ修正が行われていない可能性があります。

トピック

- MSK クラスターのバージョンの確認
- クライアントマシンでのトピックの作成

MSK クラスターのバージョンの確認

1. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
2. ナビゲーションバーで、MSK クラスターを作成したリージョンを選択します。
3. MSK クラスターを選択します。
4. クラスターで使用されている Apache Kafka のバージョンをメモします。
5. このチュートリアルの Amazon MSK バージョン番号のインスタンスは、ステップ 3 で取得したバージョンに置き換えてください。

クライアントマシンでのトピックの作成

1. クライアントマシンに接続します。

   a. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。

   b. ナビゲーションペインで、[インスタンス] を選択してください。次に、 で作成したクライアントマシンの名前の横にあるチェックボックスをオンにしますステップ 3: クライアントマシンを作成する。

   c. [Actions (アクション)] を選択して、[Connect (接続)] を選択します。コンソールの指示に従ってクライアントマシンに接続します。

2. Java をインストールし、Kafka バージョン環境変数を設定します。

   a. 次のコマンドを実行して、クライアントマシンに Java をインストールします。

   ```
   sudo yum -y install java-11
   ```

   b. 次のコマンドKAFKA_VERSIONに示すように、MSK クラスターの Kafka バージョンを環境変数 に保存します。この情報は、セットアップ全体で必要になります。

   ```
   export KAFKA_VERSION={KAFKA VERSION}
   ```

   たとえば、バージョン 3.6.0 を使用している場合は、次のコマンドを使用します。

```
export KAFKA_VERSION=3.6.0
```

3. Apache Kafka をダウンロードして抽出します。

   a. 次のコマンドを実行して、Apache Kafka をダウンロードします。

   ```
   wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
   $KAFKA_VERSION.tgz
   ```

   > **Note**
   >
   > 次のリストは、問題が発生した場合に使用できる代替 Kafka ダウンロード情報を示しています。
   >
   > - 接続の問題が発生したり、ミラーサイトを使用したりする場合は、次のコマンドに示すように、Apache ミラーセレクタを試してください。
   >
   >   ```
   >   wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
   >   kafka_2.13-$KAFKA_VERSION.tgz
   >   ```
   >
   > - Apache Kafka ウェブサイトから直接適切なバージョンをダウンロードします。

   b. 前のステップで TAR ファイルをダウンロードしたディレクトリで次のコマンドを実行します。

   ```
   tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
   ```

   c. 新しく作成したディレクトリへのフルパスをKAFKA_ROOT環境変数内に保存します。

   ```
   export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
   ```

4. MSK クラスターの認証を設定します。

   a. Amazon MSK IAM クライアントライブラリの最新バージョンを検索します。このライブラリを使用すると、クライアントマシンは IAM 認証を使用して MSK クラスターにアクセスできます。

   b. 次のコマンドを使用して、 `$KAFKA_ROOT/libs` ディレクトリに移動し、前のステップで見つけた関連する Amazon MSK IAM JAR をダウンロードします。*`{LATEST VERSION}`* は、ダウンロードする実際のバージョン番号に置き換えてください。

```
cd $KAFKA_ROOT/libs
```

```
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-iam-auth-{LATEST VERSION}-all.jar
```

> **Note**
>
> MSK クラスターとやり取りする Kafka コマンドを実行する前に、Java クラスパスに Amazon MSK IAM JAR ファイルを追加する必要がある場合があります。次の例に示すように、 CLASSPATH環境変数を設定します。
>
> ```
> export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-{LATEST VERSION}-all.jar
> ```
>
> これにより、セッション全体の が設定され、以降のすべての Kafka コマンドで JAR を使用CLASSPATHできるようになります。

c. $KAFKA_ROOT/config ディレクトリに移動して、クライアント設定ファイルを作成します。

```
cd $KAFKA_ROOT/config
```

d. 次のプロパティ設定をコピーして、新しいファイルに貼り付けます。**client.properties** という名前でファイルを保存します。

```
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandle
```

5. （オプション) Kafka ツールの Java ヒープサイズを調整します。

   メモリ関連の問題が発生した場合、または多数のトピックやパーティションを使用している場合は、Java ヒープサイズを調整できます。これを行うには、Kafka コマンドを実行する前に KAFKA_HEAP_OPTS環境変数を設定します。

   次の例では、最大ヒープサイズと初期ヒープサイズの両方を 512 メガバイトに設定します。これらの値は、特定の要件と使用可能なシステムリソースに従って調整します。

```
export KAFKA_HEAP_OPTS="-Xmx512M -Xms512M"
```

6. クラスター接続情報を取得します。

   a. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。

   b. クラスターのステータスが [アクティブ] になるまで待ちます。この処理には数分かかることがあります。ステータスが [アクティブ] になったら、クラスター名を選択します。これにより、そのクラスターの概要を含むページに移動します。

   c. View client information (ライアント情報の表示) を選択します。

   d. プライベートエンドポイントの接続文字列をコピーします。

      ブローカーごとに 3 つのエンドポイントを取得します。次のコマンドに示すようにBOOTSTRAP_SERVER、これらの接続文字列の 1 つを環境変数 に保存します。*<bootstrap-server-string>* を接続文字列の実際の値に置き換えます。

      ```
      export BOOTSTRAP_SERVER=<bootstrap-server-string>
      ```

7. 次のコマンドを実行して、 トピックを作成します。

```
$KAFKA_ROOT/bin/kafka-topics.sh --create --bootstrap-server $BOOTSTRAP_SERVER
 --command-config $KAFKA_ROOT/config/client.properties --replication-factor 3 --
partitions 1 --topic MSKTutorialTopic
```

NoSuchFileException client.properties ファイルの を取得した場合は、このファイルが Kafka bin ディレクトリ内の現在の作業ディレクトリに存在することを確認してください。

> **Note**
>
> セッション全体にCLASSPATH環境変数を設定しない場合は、代わりに各 Kafka コマンドに CLASSPATH変数をプレフィックスできます。このアプローチは、その特定のコマンドにのみクラスパスを適用します。
>
> ```
> CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-{LATEST VERSION}-all.jar \
> $KAFKA_ROOT/bin/kafka-topics.sh --create \
> --bootstrap-server $BOOTSTRAP_SERVER \
> --command-config $KAFKA_ROOT/config/client.properties \
> --replication-factor 3 \
> --partitions 1 \
> ```

```
--topic MSKTutorialTopic
```

8. (オプション) トピックが正常に作成されたことを確認します。

   a. コマンドが成功すると、次のメッセージが表示されます。`Created topic MSKTutorialTopic.`

   b. すべてのトピックを一覧表示して、トピックが存在することを確認します。

```
$KAFKA_ROOT/bin/kafka-topics.sh --list --bootstrap-server $BOOTSTRAP_SERVER --
command-config $KAFKA_ROOT/config/client.properties
```

   コマンドが失敗した場合、またはエラーが発生した場合は、トラブルシューティング情報Amazon MSK クラスターをトラブルシューティングするについて「」を参照してください。

9. (オプション) このチュートリアルで使用した環境変数を削除します。

   このチュートリアルの次のステップで環境変数を保持する場合は、このステップをスキップします。それ以外の場合は、次の例に示すように、これらの変数の設定を解除できます。

```
unset KAFKA_VERSION KAFKA_ROOT BOOTSTRAP_SERVER CLASSPATH KAFKA_HEAP_OPTS
```

次のステップ

ステップ 5: データを生成および消費する

ステップ 5: データを生成および消費する

「Amazon MSK の使用を開始する」のこのステップでは、データを生成および消費します。

メッセージを生成および消費するには

1. 次のコマンドを実行して、コンソールプロデューサーを起動します。

```
$KAFKA_ROOT/bin/kafka-console-producer.sh --broker-list $BOOTSTRAP_SERVER --
producer.config $KAFKA_ROOT/config/client.properties --topic MSKTutorialTopic
```

2. 必要なメッセージを入力して、Enter キーを押します。このステップを 2、3 回繰り返します。行を入力して [Enter] キーを押すたびに、その行は個別のメッセージとして Apache Kafka クラスターに送信されます。

3. クライアントマシンへの接続を開いたままにして、そのマシンへ 2 番目の別の接続を新しいウィンドウで開きます。これは新しいセッションであるため、 KAFKA_ROOTおよびBOOTSTRAP_SERVER環境変数を再度設定します。これらの環境変数を設定する方法については、「」を参照してくださいクライアントマシンでのトピックの作成。
4. クライアントマシンへの 2 番目の接続文字列を使用して次のコマンドを実行して、コンソールコンシューマーを作成します。

```
$KAFKA_ROOT/bin/kafka-console-consumer.sh --bootstrap-server $BOOTSTRAP_SERVER --
consumer.config $KAFKA_ROOT/config/client.properties --topic MSKTutorialTopic --
from-beginning
```

   コンソールプロデューサーコマンドを使用したときに、前に入力したメッセージの表示を開始する必要があります。
5. プロデューサーウィンドウにさらにメッセージを入力し、コンシューマーウィンドウに表示されるようにします。

次のステップ

ステップ 6: Amazon CloudWatch を使用して Amazon MSK メトリクスを表示する

ステップ 6: Amazon CloudWatch を使用して Amazon MSK メトリクスを表示する

Amazon MSK の使用をスタートするのこのステップでは、Amazon CloudWatch の Amazon MSK メトリクスを確認します。

CloudWatch で Amazon MSK メトリクスを表示するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. ナビゲーションペインで Metrics (メトリクス) を選択します。
3. [All metrics (すべてのメトリクス)] タブを選択し、[AWS/Kafka] を選択します。
4. ブローカーレベルのメトリクスを表示するには、[Broker ID, Cluster Name (ブローカー ID、クラスター名)] を選択します。クラスターレベルのメトリクスの場合は、[Cluster Name (クラスター名)] を選択します。
5. (オプション) グラフペインで、統計と期間を選択し、これらの設定を使用して CloudWatch アラームを作成します。

次のステップ

ステップ 7: このチュートリアル用に作成された AWS リソースを削除する

ステップ 7: このチュートリアル用に作成された AWS リソースを削除する

「Amazon MSK の使用を開始する」の最後のステップでは、このチュートリアルで作成した MSK クラスターとクライアントマシンを削除します。

を使用してリソースを削除するには AWS Management Console

1. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
2. クラスターの名前を選択します。例えば、MSKTutorialCluster などです。
3. [アクション] を選択してから、[削除] をクリックします。
4. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。
5. クライアントマシン用に作成したインスタンス (例えば **`MSKTutorialClient`**) を選択します。
6. [Instance state] (インスタンスの状態) を選択し、[Terminate instance] (インスタンスの終了) をクリックします。

IAM ポリシーとロールを削除するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで [Roles] (ロール) を選択します。
3. 検索ボックスに、このチュートリアル用に作成した IAM ロールの名前を入力します。
4. ロールを選択します。[ロールの削除] を選択し、削除を確定します。
5. ナビゲーションペインで [Policies] (ポリシー) を選択します。
6. 検索ボックスに、このチュートリアル用に作成したポリシーの名前を入力します。
7. ポリシーを選択すると、その概要ページが開きます。ポリシーの[Summary] (概要) ページで [Delete policy] (ポリシーの削除) を選択します。
8. [Delete] (削除) をクリックします。

Amazon MSK: 仕組み

Amazon MSK は、フルマネージド型の Apache Kafka サービスであり、Apache Kafka を使用してストリーミングデータを処理するアプリケーションを簡単に構築および実行できます。このガイドで

は、開発者が Amazon MSK の仕組みと、アプリケーションで効果的に使用する方法を理解するのに役立つ情報を提供します。

大まかに言うと、Amazon MSK は、 によってプロビジョニングおよび運用されるフルマネージド Apache Kafka クラスターを提供します AWS。つまり、EC2 インスタンスのプロビジョニング、ネットワーク設定の設定、Kafka ブローカーの管理、継続的なメンテナンスタスクの実行について心配する必要はありません。代わりに、アプリケーションの構築に集中し、Amazon MSK にインフラストラクチャを処理させることができます。Amazon MSK は、必要なコンピューティング、ストレージ、ネットワークリソースを自動的にプロビジョニングし、自動スケーリング、高可用性、フェイルオーバーなどの機能を提供して、Kafka クラスターの信頼性と可用性を確保します。このガイドでは、Amazon MSK の主要なコンポーネントと、それを使用してストリーミングデータアプリケーションを構築する方法について説明します。

プロビジョニングされたクラスターを管理する

Amazon MSK クラスターは、お客様のアカウントで作成できる最も基本的な Amazon MSK のリソースです。このセクションの各トピックでは、Amazon MSK における一般的なオペレーションの実行方法について説明します。MSK クラスターで実行できる全オペレーションのリストについては、次を参照してください。

- AWS Management Console
- Amazon MSK API リファレンス
- Amazon MSK CLI コマンドリファレンス

トピック

- MSK プロビジョンドクラスターを作成する
- Amazon MSK クラスターを一覧表示する
- Amazon MSK プロビジョンドクラスターに接続する
- Amazon MSK クラスターのブートストラップブローカーを取得する
- Amazon MSK プロビジョンドクラスターをモニタリングする
- Amazon MSK クラスターのセキュリティ設定を更新する
- Amazon MSK クラスター内のブローカーの数を拡張する
- Amazon MSK クラスターからブローカーを削除する
- Amazon MSK クラスター内の標準ブローカーのストレージスループットをプロビジョニングする

- Amazon MSK クラスターのブローカーサイズを更新する
- LinkedIn の Cruise Control for Apache Kafka を Amazon MSK で使用する
- Amazon MSK クラスターの設定を更新する
- Amazon MSK クラスターのブローカーを再起動する
- Amazon MSK クラスターをタグ付ける
- Amazon MSK クラスターへの移行
- Amazon MSK プロビジョンドクラスターを削除する

MSK プロビジョンドクラスターを作成する

> **Important**
>
> クラスターの作成後に MSK プロビジョンドクラスターの VPC を変更することはできません。

MSK プロビジョンドクラスターを作成する前に、 Amazon Virtual Private Cloud (VPC) を持ち、その VPC 内にサブネットを設定する必要があります。

米国西部 (北カリフォルニア) リージョンのスタンダードブローカーの場合、2 つの異なるアベイラビリティーゾーンに 2 つのサブネットが必要です。Amazon MSK が使用可能なその他のリージョンでは、サブネットを 2 つまたは 3 つ指定できます。サブネットはすべて、異なるアベイラビリティーゾーンに存在している必要があります。Express ブローカーの場合、3 つの異なるアベイラビリティーゾーンに 3 つのサブネットが必要です。MSK プロビジョンドクラスターを作成すると、Amazon MSK は指定したサブネットにブローカーノードを均等に分散します。

トピック

- を使用して MSK プロビジョンドクラスターを作成する AWS Management Console
- を使用してプロビジョニングされた Amazon MSK クラスターを作成する AWS CLI
- を使用してカスタム Amazon MSK 設定で MSK プロビジョンドクラスターを作成する AWS CLI
- Amazon MSK API を使用して MSK プロビジョンドクラスターを作成する

を使用して MSK プロビジョンドクラスターを作成する AWS Management Console

このトピックの手順では、 のカスタム作成オプションを使用して MSK プロビジョンドクラスターを作成する一般的なタスクについて説明します AWS Management Console。で使用できる他のオプションを使用して AWS Management Console、以下を作成することもできます。

- サーバーレスクラスター
- クイック作成オプションを使用した MSK プロビジョンドクラスター

このトピックの手順

- ステップ 1: クラスターの初期セットアップと設定
- ステップ 2: ストレージとクラスターの設定を構成する
- ステップ 3: ネットワーク設定を構成する
- ステップ 4: セキュリティ設定を構成する
- ステップ 5: モニタリングオプションを設定する
- ステップ 6: クラスター設定を確認する

ステップ 1: クラスターの初期セットアップと設定

1. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
2. Create cluster (クラスターの作成) を選択します。
3. [クラスターの作成方法] で、[カスタム作成] を選択します。
4. クラスター名には、一意で 64 文字以下の名前を指定します。
5. [クラスタータイプ] に [プロビジョンド] を選択します。
6. Apache Kafka バージョンでは、ブローカーで実行するバージョンを選択します。各 Apache Kafka バージョンでサポートされている Amazon MSK 機能の比較を確認するには、「バージョンの互換性を表示」を選択します。
7. ブローカーセクションで、次の操作を行います。

    a. ブローカータイプで、次のいずれかのオプションを選択します。

    - Express ブローカー: フルマネージド仮想ストレージを備えた高性能でスケーラブルなブローカー。要求の厳しい高スループットアプリケーションには、このブローカータイプを選択します。

- 標準ブローカー: フル設定制御を備えた従来の Kafka ブローカー。スループット要件が中程度の汎用ワークロードには、このブローカータイプを選択します。

これらのブローカータイプの詳細については、「」を参照してくださいAmazon MSK ブローカータイプ。

b. ブローカーサイズでは、クラスターのコンピューティング、メモリ、ストレージのニーズに基づいて、クラスターに使用するサイズを選択します。

c. ゾーンの数 で、ブローカーが分散AWS アベイラビリティーゾーンされる の数を選択します。

Express ブローカーでは、可用性を高めるために 3 つのアベイラビリティーゾーンが必要です。

d. ゾーンあたりのブローカーについては、Amazon MSK が各アベイラビリティーゾーンに作成するブローカーの数を指定します。最小値はアベイラビリティーゾーンごとに 1、最大値は ZooKeeper ベースのクラスターの場合クラスターごとに 30、KRaft ベースのクラスターの場合クラスターごとに 60 です。

ステップ 2: ストレージとクラスターの設定を構成する

この手順では、すべてのブローカーでデータストレージのニーズを設定し、ストレージモードを指定する方法について説明します。これにより、ワークロードのニーズに基づいてデータストレージ要件を定義できます。さらに、この手順では、ブローカーの動作を制御するクラスター設定について説明します。これらの設定には、ブローカー設定、デフォルトのトピック設定、階層型ストレージポリシーが含まれます。

1. ブローカータイプを標準として選択した場合は、ストレージセクションで次の操作を行います。

a. Storage で、クラスターに必要なストレージの初期量を選択します。クラスターの作成後にストレージ容量を減らすことはできません。

b. (オプション) 選択したブローカーサイズ (インスタンスサイズ) に応じて、ブローカーごとにプロビジョンドストレージスループットを指定することもできます。このオプションを使用すると、各ブローカーの Amazon EBS ボリュームに専用の入出力 (I/O) パフォーマンスを割り当てることができます。

このオプションを有効にするには、x86 の場合 kafka.m5.4xlarge 以上、Graviton ベースのインスタンスの場合 kafka.m7g.2xlarge 以上のブローカーサイズ (インスタンスサイズ) を選択します。次に、プロビジョニングされたストレージスループットを有効にするチェック

ボックスをオンにします。このチェックボックスをオンにすると、1 秒あたり最低 250 MiB のスループットを手動で設定できます。これは、I/O を多用するワークロードや、高速で予測可能なストレージパフォーマンスを必要とするアプリケーションに役立ちます。詳細については、「???」を参照してください。

c. クラスターストレージモードでは、クラスター内でデータを保存および管理する方法を指定します。このオプションは、ブローカーに使用されるストレージのタイプと設定を決定します。以下のオプションのいずれかを選択してください：

- EBS ストレージのみ: すべてのトピックデータを各ブローカーにアタッチされた Amazon Elastic Block Store (Amazon EBS) ボリュームにローカルに保存します。一貫したパフォーマンスニーズと最近のメッセージへの迅速なアクセスのために、このモードを選択します。
- 階層型ストレージと EBS ストレージ: ローカル Amazon EBS データと Amazon S3 の大規模なデータセットのリモートでコスト効率の高いストレージを組み合わせます。このモードでは、Amazon EBS ストレージコストを削減し、より長いデータ保持をサポートし、手動による介入なしでストレージを自動的にスケーリングします。データを低コストで長期間保持する場合や、ストレージの大幅な増加が予想される場合は、このモードを選択します。

> **Note**
>
> Express ブローカーのストレージを管理する必要はありません。

2. クラスター設定では、次のいずれかのオプションを指定してクラスターの動作を定義します。

- Amazon MSK デフォルト設定: 汎用ユースケース用に最適化された事前定義された設定のセットが含まれています。クラスターのセットアップとデプロイをすばやく行うには、このオプションを選択します。Amazon MSK の設定については、「Amazon MSK プロビジョンド設定」を参照してください。
- カスタム設定: 独自のブローカーとトピックの設定を指定できます。リストから既存のカスタム設定を選択するか、新しいカスタム設定を作成できます。特定のパフォーマンス調整、セキュリティ設定など、ブローカーの微調整されたコントロールには、このオプションを選択します。

3. [Next] (次へ) をクリックして先に進みます。

ステップ 3: ネットワーク設定を構成する

ネットワーク設定は、クラスターを AWS インフラストラクチャ内にデプロイする方法を定義します。これには、ネットワーク、可用性、アクセスを制御する VPC、アベイラビリティーゾーンとサブネット、セキュリティグループが含まれます。

1. ネットワーキングの場合は、次の操作を行います。

   a. クラスターに使用する VPC を選択します。

   b. 以前に選択したアベイラビリティーゾーンの数に基づいて、ブローカーがデプロイするアベイラビリティーゾーンとサブネットを指定します。

      米国西部 (北カリフォルニア) リージョンのスタンダードブローカーの場合、2 つの異なるアベイラビリティーゾーンに 2 つのサブネットが必要です。Amazon MSK が使用可能なその他のリージョンでは、サブネットを 2 つまたは 3 つ指定できます。サブネットはすべて、異なるアベイラビリティーゾーンに存在している必要があります。

      Express ブローカーの場合、3 つの異なるアベイラビリティーゾーンに 3 つのサブネットが必要です。

      MSK プロビジョンドクラスターを作成すると、MSK は指定したサブネットにブローカーノードを均等に分散します。

   c. Amazon EC2 のセキュリティグループでは、クラスターへのアクセスを許可する 1 つ以上のセキュリティグループを選択または作成します。これらの Amazon EC2 セキュリティグループは、ブローカーへのインバウンドトラフィックとアウトバウンドトラフィックを制御します。たとえば、クライアントマシンのセキュリティグループなどです。

      共有されるセキュリティグループを指定する場合は、そのセキュリティグループを使用するアクセス許可があることを確認する必要があります。具体的には、`ec2:DescribeSecurityGroups` アクセス許可が必要です。詳細については、「MSK クラスターへの接続」を参照してください。

2. [Next] (次へ) をクリックして先に進みます。

ステップ 4: セキュリティ設定を構成する

1. セキュリティ設定セクションで、次の操作を行います。

   - Kafka クラスターへのクライアントアクセスを制御するには、次の認証および認可方法を 1 つ以上選択します。

- 認証されていないアクセス: クライアントが認証情報を指定せずにクラスターにアクセスできるようにします。この方法はセキュリティリスクであり、セキュリティのベストプラクティスに準拠していない可能性があります。詳細については、「msk-unrestricted-access-check」を参照してください。
- IAM ロールベースの認証: AWS IAM ユーザー/ロールを使用したクライアント認証と認可を有効にします。この方法では、IAM ポリシーによるクラスターアクセスをきめ細かく制御できます。この方法は、すでに で実行されているアプリケーションに推奨されます AWS。
- SASL/SCRAM 認証: クライアントは認証 AWS Secrets Manager のために に保存されているユーザー名とパスワードの認証情報を提供する必要があります。Amazon MSK は、Secrets Manager からこれらの認証情報を取得し、ユーザーを安全に認証します。

  クラスターの認証に関するサインイン認証情報を設定するには、まず Secrets Manager で Secret リソースを作成します。次に、サインイン認証情報をそのシークレットに関連付けます。このアクセスコントロール方法の詳細については、「」を参照してくださいAmazon MSK クラスターの SASL/SCRAM 認証を設定する。
- AWS Certificate Manager (ACM) による TLS クライアント認証: デジタル証明書を使用したクライアントとブローカー間の相互認証を有効にします。 AWS Private Certificate Authority （ AWS Private CA) は、 AWS アカウント クラスターと同じ または異なる で設定する必要があります。

  mTLS AWS Private CAを実装するときは、MSK クラスターごとに独立した を使用することを強くお勧めします。これにより、PCAs によって署名された TLS 証明書は 1 つの MSK クラスターでのみ認証されるため、厳格なアクセスコントロールが維持されます。

2. 暗号化で、保管中のデータの暗号化に使用する KMS キーの種類を選択します。詳細については、「the section called “保管中の Amazon MSK 暗号化”」を参照してください。

   保管中のデータを暗号化すると保存データの整合性が保護され、転送中のデータを暗号化すると転送中のネットワークモニタリングからデータの機密性が保護されます。
3. [Next] (次へ) をクリックして先に進みます。

ステップ 5: モニタリングオプションを設定する

この手順では、ブローカーメトリクスをセットアップし、ブローカーログを収集して配信する方法について説明します。これらの設定により、クラスターの状態、パフォーマンス、問題のトラブル

シューティングを監視および分析できます。詳細については、「the section called “クラスターをモニタリングする”」を参照してください。

1. このクラスターの Amazon CloudWatch メトリクスでは、次のいずれかのモニタリングレベルを選択します。各モニタリングレベルで収集されたメトリクスは、可視化とアラートのために CloudWatch と統合されています。

   a. 基本モニタリング: クラスターレベルの必須メトリクスのセットを追加料金なしで提供します。このレベルは、一般的なモニタリングが必要なほとんどのユースケースに適しています。

   b. ブローカーレベルのモニタリングの強化: 詳細なブローカーメトリクスを追加料金で提供します。このレベルには、基本的なモニタリングと、他のブローカーの階層型ストレージメトリクスのバイトイン/アウト、読み取り/書き込みオペレーションの合計時間など、より詳細なブローカーメトリクスが含まれます。このレベルのメトリクスに対して料金が発生しますが、基本レベルのメトリクスは引き続き無料です。

   c. トピックレベルのモニタリングの強化: 個々のトピックのメトリクスを追加料金で提供します。このレベルを選択すると、ブローカー全体のトピックパフォーマンスをより詳細に確認できます。このレベルには、ブローカーレベルのモニタリングの強化と、指定されたトピックの階層型ストレージメトリクスや 1 秒あたりに受信されたメッセージ数などのトピックレベルのメトリクスが含まれます。

   d. 拡張パーティションレベルのモニタリング: パーティションあたりのメトリクスの最も詳細なビューを追加料金で提供します。このレベルを選択すると、ブローカー間で各トピック内の各パーティションのメトリクスをキャプチャして、最も詳細なモニタリングを取得できます。このレベルには、強化されたトピックレベルのモニタリングと、オフセットラグメトリクスなどのきめ細かなパーティション固有のメトリクスが含まれます。

   これらの各モニタリングレベルで Standard および Express ブローカータイプで使用できるメトリクスの詳細については、標準ブローカーの CloudWatch メトリクス「」および「」を参照してくださいExpress ブローカーの CloudWatch メトリクス。

2. （オプション) JMX Exporter、Node Exporter、またはその両方を使用してメトリクスを Prometheus 形式でエクスポートする場合は、Prometheus でオープンモニタリングを有効にするを選択します。このオプションの詳細については、「Prometheus でモニタリングする」を参照してください。

3. （オプション) トラブルシューティングと監査 AWS のサービス のためにブローカーログをさまざまな に配信するように MSK クラスターを設定するには、次のオプションを 1 つ以上選択

します。Amazon MSK は、これらの宛先リソースがまだ存在しない場合、それらを作成しません。詳細については、「ブローカーログ」を参照してください。

- Amazon CloudWatch Logs への配信: クラスタリング、検索、視覚化機能を使用して CloudWatch にログを送信します。を離れることなく、ログをクエリおよび分析できます AWS Management Console。
- Amazon S3 に配信: 長期的なアーカイブとバッチ分析のために、ログをファイルとして Amazon S3 バケットに保存します。
- Amazon Data Firehose への配信: Amazon OpenSearch Service への自動配信のために Firehose にログを送信し、リアルタイムのトラブルシューティングを行います。

4. (オプション) クラスターの識別、整理、検索を行うには、新しいタグの追加を選択して、キーと値のペアとしてタグを追加します。たとえば、 **`Load testing`**と のキーと値のペアを使用してクラスターにタグを追加します**`Test`**。

   クラスターでのタグの使用の詳細については、「」を参照してくださいAmazon MSK クラスターをタグ付ける。

5. [Next] (次へ) をクリックして先に進みます。

ステップ 6: クラスター設定を確認する

1. クラスターの設定を確認します。

   編集または前 を選択して、以前に指定した設定を変更するか、前のコンソール画面に戻ります。

2. [クラスターを作成] を選択します。
3. クラスターの詳細ページのクラスター概要セクションで、このクラスターのステータスを確認します。Amazon MSK がクラスターをプロビジョニングすると、ステータスが [作成中] から [アクティブ] に変わります。ステータスが [アクティブ] の場合、クラスターに接続できます。クラスターのステータスの詳細については、「MSK プロビジョンドクラスターの状態を理解する」を参照してください。

を使用してプロビジョニングされた Amazon MSK クラスターを作成する AWS CLI

1. 次の JSON をコピーして、ファイルに保存します。ファイルを `brokernodegroupinfo.json` と名付けます。JSON のサブネット ID を、サブネットに対応する値に置き換えます。これらの

サブネットは、異なるアベイラビリティーゾーンに存在している必要があります。*"Security-Group-ID"* を、クライアント VPC の 1 つ以上のセキュリティグループの ID に置き換えます。これらのセキュリティグループに関連付けられたクライアントは、クラスターへのアクセスを取得します。共有されたセキュリティグループを指定する場合は、そのセキュリティグループに対するアクセス許可があることを確認する必要があります。具体的には、ec2:DescribeSecurityGroups アクセス許可が必要です。例については、「Amazon EC2: 特定の VPC に関連付けられた Amazon EC2 セキュリティグループの管理をプログラムによりコンソールで許可する」を参照してください。最後に、 AWS CLI がインストールされているコンピュータに、更新された JSON ファイルを保存します。

```
{
  "InstanceType": "kafka.m5.large",
  "ClientSubnets": [
    "Subnet-1-ID",
    "Subnet-2-ID"
  ],
  "SecurityGroups": [
    "Security-Group-ID"
  ]
}
```

> **Important**
>
> Express ブローカーの場合、3 つの異なるアベイラビリティーゾーンに 3 つのサブネットが必要です。また、ストレージ関連のプロパティを定義する必要はありません。
> 米国西部 (北カリフォルニア) リージョンのスタンダードブローカーの場合、2 つの異なるアベイラビリティーゾーンに 2 つのサブネットが必要です。Amazon MSK が使用可能なその他のリージョンでは、サブネットを 2 つまたは 3 つ指定できます。サブネットはすべて、異なるアベイラビリティーゾーンに存在している必要があります。クラスターを作成すると、Amazon MSK により、指定したサブネット上にブローカーノードが均等に割り当てられます。

2. brokernodegroupinfo.json ファイルを保存したディレクトリで次の AWS CLI コマンドを実行し、*Your-Cluster-Name*」を任意の名前に置き換えます。*"Monitoring-Level"* では、DEFAULT、PER_BROKER、または PER_TOPIC_PER_BROKER の 3 つの値のうちいずれかを指定できます。これらの、モニタリングの 3 つの異なるレベルについて詳しくは、「???」を参照してください enhanced-monitoring パラメータはオプションです。create-cluster コマンドでこのパラメータを指定しない場合、DEFAULT レベルのモニタリングを取得します。

```
aws kafka create-cluster --cluster-name "Your-Cluster-Name" --broker-node-group-
info file://brokernodegroupinfo.json --kafka-version "2.8.1" --number-of-broker-
nodes 3 --enhanced-monitoring "Monitoring-Level"
```

コマンドの出力は以下の JSON のようになります。

```
{
    "ClusterArn": "...",
    "ClusterName": "AWSKafkaTutorialCluster",
    "State": "CREATING"
}
```

> **Note**
>
> create-cluster コマンドにより、1 つ以上のサブネットが、サポートされていないアベイラビリティーゾーンに属しているというエラーが返される場合があります。この場合、エラーはどのアベイラビリティーゾーンがサポートされていないかを示します。サポートされていないアベイラビリティーゾーンを使用しないサブネットを作成して、create-cluster コマンドを再度実行してください。

3. クラスターで他のアクションを実行するために必要になるため、ClusterArn キーの値を保存します。
4. 次のコマンドを実行して、クラスターの STATE を確認します。Amazon MSK がクラスターをプロビジョニングすると、STATE 値が CREATING から ACTIVE に変わります。状態が ACTIVE の場合、クラスターに接続できます。クラスターのステータスの詳細については、「MSK プロビジョンドクラスターの状態を理解する」を参照してください。

```
aws kafka describe-cluster --cluster-arn <your-cluster-ARN>
```

を使用してカスタム Amazon MSK 設定で MSK プロビジョンドクラスターを作成する AWS CLI

カスタム Amazon MSK 設定とその作成方法については、「the section called “ブローカー設定”」を参照してください。

1. ファイルに次の JSON を保存し、*configuration-arn* をクラスターの作成に使用する設定の ARN に置き換えます。

```
{
    "Arn": configuration-arn,
    "Revision": 1
}
```

2. create-cluster コマンドを実行し、configuration-info オプションを使用して、前の手順で保存した JSON ファイルを指定します。以下に例を示します。

```
aws kafka create-cluster --cluster-name ExampleClusterName --broker-node-group-
info file://brokernodegroupinfo.json --kafka-version "2.8.1" --number-of-broker-
nodes 3 --enhanced-monitoring PER_TOPIC_PER_BROKER --configuration-info file://
configuration.json
```

次に、このコマンドを実行した後の正常な応答の例を示します。

```
{
    "ClusterArn": "arn:aws:kafka:us-east-1:123456789012:cluster/
CustomConfigExampleCluster/abcd1234-abcd-dcba-4321-a1b2abcd9f9f-2",
    "ClusterName": "CustomConfigExampleCluster",
    "State": "CREATING"
}
```

Amazon MSK API を使用して MSK プロビジョンドクラスターを作成する

Amazon MSK API を使用すると、自動インフラストラクチャプロビジョニングまたはデプロイスクリプトの一部として MSK プロビジョンドクラスターをプログラムで作成および管理できます。

API を使用して MSK プロビジョンドクラスターを作成するには、CreateCluster」を参照してください。

Amazon MSK クラスターを一覧表示する

Amazon MSK クラスターのブートストラップブローカーを取得するには、クラスターの Amazon リソースネーム (ARN) が必要です。クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。「the section called “ブートストラップブローカーを取得する”」を参照してください。

トピック

- を使用してクラスターを一覧表示する AWS Management Console
- を使用してクラスターを一覧表示する AWS CLI
- API を使用してクラスターを一覧表示する

を使用してクラスターを一覧表示する AWS Management Console

Amazon MSK クラスターのブートストラップブローカーを取得するには、クラスターの Amazon リソースネーム (ARN) が必要です。クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。「the section called “ブートストラップブローカーを取得する”」を参照してください。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. このテーブルには、このアカウントにある現在のリージョンのすべてのクラスターが表示されます。詳細を表示するには、クラスターの名前を選択します。

を使用してクラスターを一覧表示する AWS CLI

Amazon MSK クラスターのブートストラップブローカーを取得するには、クラスターの Amazon リソースネーム (ARN) が必要です。クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。「the section called “ブートストラップブローカーを取得する”」を参照してください。

```
aws kafka list-clusters
```

API を使用してクラスターを一覧表示する

Amazon MSK クラスターのブートストラップブローカーを取得するには、クラスターの Amazon リソースネーム (ARN) が必要です。クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。「the section called “ブートストラップブローカーを取得する”」を参照してください。

API を使用してクラスターを一覧表示するには、「ListClusters」を参照してください。

Amazon MSK プロビジョンドクラスターに接続する

デフォルトでは、クライアントはクラスターと同じ VPC 内にある場合にのみ、MSK プロビジョンドクラスターにアクセスできます。Kafka クライアントと MSK プロビジョンドクラスター間のすべての通信はデフォルトでプライベートであり、ストリーミングデータがインターネットを通過することはありません。クラスターと同じ VPC にあるクライアントから MSK プロビジョンドクラスターに接続するには、クラスターのセキュリティグループに、クライアントのセキュリティグループからのトラフィックを受け入れるインバウンドルールがあることを確認します。これらのルールの設定については、セキュリティグループルールを参照してください。クラスターと同じ VPC にある Amazon EC2 インスタンスからクラスターにアクセスする方法の例については、「the section called "はじめに"」を参照してください。

> **Note**
>
> KRaft メタデータモードと MSK Express ブローカーでは、オープンモニタリングとパブリックアクセスの両方を有効にすることはできません。

クラスターの VPC の外部にあるクライアントから MSK プロビジョンドクラスターに接続するには、「クラスターの VPC の AWS 内外からのアクセス」を参照してください。

トピック

- MSK プロビジョンドクラスターへのパブリックアクセスを有効にする
- AWS クラスターの VPC 内外からのアクセス

MSK プロビジョンドクラスターへのパブリックアクセスを有効にする

Amazon MSK では、Apache Kafka 2.6.0 以降のバージョンを実行している MSK プロビジョンドクラスターのブローカーへのパブリックアクセスを有効にするオプションがあります。セキュリティ上の理由から、MSK クラスターの作成中に公開アクセスをオンにすることはできません。ただし、既存のクラスターを更新して、公開アクセスできるようにすることができます。また、新しいクラスターを作成してから更新して、公開アクセスできるようにすることもできます。

MSK クラスターへのパブリックアクセスは追加料金なしで有効にできますが、クラスターに出入りするデータ転送には標準データ転送 AWS コストが適用されます。料金については、Amazon EC2 オンデマンド料金をご覧ください。

> **Note**
>
> SASL/SCRAM または mTLS アクセスコントロールメソッドを使用している場合は、まずクラスターに Apache Kafka ACLs を設定する必要があります。次に、クラスターの設定を更新して、 `allow.everyone.if.no.acl.found`プロパティを false に設定します。クラスターの設定を更新する方法については、「the section called “ブローカー設定オペレーション”」を参照してください。

MSK プロビジョンドクラスターへのパブリックアクセスを有効にするには、クラスターが次の条件をすべて満たしていることを確認します。

- クラスターに関連付けられているサブネットは公開である必要があります。各パブリックサブネットにはパブリック IPv4 アドレスが関連付けられており、パブリック IPv4 アドレスは Amazon VPC の料金ページに示すように料金が設定されます。つまり、サブネットには、インターネットゲートウェイが接続されたルートテーブルが関連付けられている必要があります。インターネットゲートウェイを作成してアタッチする方法については、Amazon VPC ユーザーガイドの「インターネットゲートウェイを使用して VPC インターネットアクセスを有効にする」を参照してください。
- 認証されていないアクセス制御をオフにし、次のアクセス制御方法の少なくとも 1 つをオンにする必要があります : SASL/IAM、SASL/SCRAM、mTLS。クラスターのアクセス制御方式を更新する方法については、「the section called “クラスターセキュリティを更新する”」を参照してください。
- クラスター内の暗号化をオンにする必要があります。「オン」は、クラスターを作成するときのデフォルト設定です。暗号化をオフにして作成されたクラスターのクラスター内で暗号化をオンにすることはできません。したがって、クラスター内の暗号化をオフにして作成されたクラスターの公開アクセスをオンにすることはできません。
- ブローカーとクライアント間のプレーンテキストトラフィックは、オフにする必要があります。オンになっている場合にオフにする方法については、「the section called “クラスターセキュリティを更新する”」を参照してください。
- IAM アクセスコントロールを使用していて、認可ポリシーを適用したり、認可ポリシーを更新したりする場合は、「」を参照してくださいthe section called “IAM アクセスコントロール”。Apache Kafka ACL の詳細については、「the section called “Apache Kafka ACL”」を参照してください。

MSK クラスターが上記の条件を満たすことを確認したら、 AWS Management Console、 AWS CLI、または Amazon MSK API を使用してパブリックアクセスを有効にできます。クラスターへの公開アクセスをオンにすると、そのクラスターの公開ブートストラップブローカー文字列を取得できます。クラスターのブートストラップブローカーを取得する方法については、「the section called "ブートストラップブローカーを取得する"」を参照してください。

Important

公開アクセスをオンにすることに加えて、クラスターのセキュリティグループに IP アドレスからの公開アクセスを許可するインバウンド TCP ルールがあることを確認してください。これらのルールを可能な限り制限することをお勧めします。セキュリティグループとインバウンドルールの詳細については、「Amazon VPC ユーザーガイド」のVPC のセキュリティグループを参照してください。ポート番号については、「the section called "ポート情報"」を参照してください。クラスターのセキュリティグループを変更する方法については、「the section called "セキュリティグループの変更"」を参照してください。

Note

次の手順を使用して公開アクセスをオンにしても、クラスターにアクセスできない場合は、「the section called "パブリックアクセスが有効になっているクラスターにアクセスできない"」を参照してください。

コンソールを使用して公開アクセスをオンにする

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. クラスターのリストで、公開アクセスをオンにするクラスターを選択します。
3. [プロパティ] タブを選択し、[ネットワーク設定] セクションを探します。
4. Edit public access (公開アクセスの編集) を選択します。

を使用してパブリックアクセスを有効にする AWS CLI

1. 次の AWS CLI コマンドを実行し、*ClusterArn* と *Current-Cluster-Version* を ARN とクラスターの最新バージョンに置き換えます。クラスターの最新バージョンを検索するに

は、DescribeCluster オペレーションまたは describe-cluster AWS CLI コマンドを使用します。サンプルのバージョンは KTVPDKIKX0DER です。

```
aws kafka update-connectivity --cluster-arn ClusterArn --current-
version Current-Cluster-Version --connectivity-info '{"PublicAccess": {"Type":
 "SERVICE_PROVIDED_EIPS"}}'
```

この update-connectivity コマンドの出力は、次の JSON の例のようになります。

```
{
    "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
abcdefab-1234-abcd-5678-cdef0123ab01-2",
    "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef"
}
```

> **Note**
>
> パブリックアクセスを無効にするには、同様の AWS CLI コマンドを使用しますが、代わりに次の接続情報を使用します。
>
> ```
> '{"PublicAccess": {"Type": "DISABLED"}}'
> ```

2. update-connectivity オペレーションの結果を取得するには、*ClusterOperationArn*を update-connectivity コマンドの出力で取得した ARN に置き換えて、次のコマンドを実行します。

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

この describe-cluster-operation コマンドの出力は、次の JSON の例のようになります。

```
{
    "ClusterOperationInfo": {
        "ClientRequestId": "982168a3-939f-11e9-8a62-538df00285db",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2019-06-20T21:08:57.735Z",
```

```
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "UPDATE_COMPLETE",
        "OperationType": "UPDATE_CONNECTIVITY",
        "SourceClusterInfo": {
            "ConnectivityInfo": {
                "PublicAccess": {
                    "Type": "DISABLED"
                }
            }
        },
        "TargetClusterInfo": {
            "ConnectivityInfo": {
                "PublicAccess": {
                    "Type": "SERVICE_PROVIDED_EIPS"
                }
            }
        }
    }
}
```

`OperationState` の値が `UPDATE_IN_PROGRESS` の場合は、しばらく待ってから再度 `describe-cluster-operation` コマンドを実行します。

Amazon MSK API を使用して公開アクセスをオンにする

- API を使用してクラスターへの公開アクセスをオンまたはオフにするには、UpdateConnectivityを参照してください。

> **Note**
>
> セキュリティ上の理由から、Amazon MSK は Apache ZooKeeper ノードや KRaft コントローラーノードへの公開アクセスを許可していません。

AWS クラスターの VPC 内外からのアクセス

クラスターの Amazon VPC 内外から MSK クラスターに接続する AWS には、次のオプションがあります。

Amazon VPC ピアリング

クラスターの VPC とは異なる VPC から MSK クラスターに接続するには、2 つの VPC 間にピアリング接続を作成できます。VPC ピアリングについては、Amazon VPC ピアリングガイドを参照してください。

AWS Direct Connect

AWS Direct Connect は、標準の 1 ギガビットまたは 10 ギガビットイーサネット光ファイバーケーブル AWS を介して、オンプレミスネットワークを にリンクします。ケーブルの一端はルーターに接続され、もう一端は AWS Direct Connect ルーターに接続されます。この接続を使用すると、 AWS クラウドと Amazon VPC への仮想インターフェイスを直接作成し、ネットワークパス内のインターネットサービスプロバイダーをバイパスできます。詳細については、「AWS Direct Connect」を参照してください。

AWS Transit Gateway

AWS Transit Gateway は、VPCsとオンプレミスネットワークを単一のゲートウェイに接続できるサービスです。 AWS Transit Gatewayの使用方法については、AWS Transit Gatewayを参照してください。

VPN 接続

次のトピックで説明されている VPN 接続オプションを使用して、MSK クラスターの VPC をリモートネットワークおよびユーザーに接続できます : VPN接続。

REST プロキシ

クラスターの Amazon VPC 内で実行されているインスタンスに REST プロキシをインストールできます。REST プロキシを使用すると、プロデューサーとコンシューマーは HTTP API リクエストを介してクラスターと通信できます。

複数リージョンのマルチ VPC 接続

次のドキュメントでは、異なる リージョンに存在する複数の VPC の接続オプションについて説明します : 複数リージョンのマルチ VPC 接続。

単一リージョンのマルチ VPC プライベート接続

Amazon Managed Streaming for Apache Kafka (Amazon MSK) クラスターのマルチ VPC プライベート接続 (AWS PrivateLink を使用) は、さまざまな仮想プライベートクラウド (VPCs) と AWS アカウントでホストされている Kafka クライアントをより迅速に Amazon MSK クラスターに接続できる機能です。

「クロスアカウントクライアントの単一リージョンマルチ VPC 接続」を参照してください。

EC2-Classic Networking は廃止されました

Amazon MSK は、Amazon EC2-Classic Networking で実行される Amazon EC2 インスタンスをサポートしなくなりました。

「EC2-Classic Networking は販売終了になります — 準備方法はこちら」を参照してください。

単一リージョンの Amazon MSK マルチ VPC プライベート接続

Amazon Managed Streaming for Apache Kafka (Amazon MSK) クラスターのマルチ VPC プライベート接続 (AWS PrivateLink を使用) は、さまざまな仮想プライベートクラウド (VPCs) と AWS アカウントでホストされている Kafka クライアントをより迅速に Amazon MSK クラスターに接続できる機能です。

マルチ VPC プライベート接続は、マルチ VPC 接続とクロスアカウント接続のネットワークインフラストラクチャを簡素化するマネージドソリューションです。クライアントは、ネットワーク内のすべてのトラフィックを維持しながら、PrivateLink 経由で Amazon MSK クラスターに接続できます AWS 。Amazon MSK クラスターのマルチ VPC プライベート接続は、Amazon MSK が利用可能なすべての AWS リージョンで使用できます。

トピック

- マルチ VPC プライベート接続とは
- マルチ VPC プライベート接続の利点
- マルチ VPC プライベート接続の要件と制限
- マルチ VPC プライベート接続の使用を開始する
- クラスターの認可スキームの更新
- Amazon MSK クラスターへのマネージド VPC 接続の拒否
- Amazon MSK クラスターへのマネージド VPC 接続の削除
- マルチ VPC プライベート接続のアクセス許可

マルチ VPC プライベート接続とは

Amazon MSK のマルチ VPC プライベート接続は、異なる仮想プライベートクラウド (VPCs) と AWS アカウントでホストされている Apache Kafka クライアントを MSK クラスターに接続できるようにする接続オプションです。

Amazon MSK はクラスターポリシーによりクロスアカウントアクセスを簡素化します。これらのポリシーにより、クラスター所有者は他の AWS アカウントに MSK クラスターへのプライベート接続を確立するためのアクセス許可を付与できます。

マルチ VPC プライベート接続の利点

他の接続ソリューションと比べ、マルチ VPC プライベート接続にはいくつかの利点があります。

- AWS PrivateLink 接続ソリューションの運用管理を自動化します。
- 接続している VPC 全体で重複する IP を使用できるため、重複しない IP、複雑なピアリング、および他の VPC 接続ソリューションに関連付けられたルーティングテーブルを維持する必要がなくなります。

MSK クラスターのクラスターポリシーを使用して、MSK クラスターへのクロスアカウントプライベート接続を設定するアクセス許可 AWS を持つアカウントを定義します。クロスアカウント管理者は、適切なロールまたはユーザーにアクセス許可を委任できます。IAM クライアント認証と併用すると、クラスターポリシーを使用して、接続クライアントの Kafka データプレーンのアクセス許可を細かく定義することもできます。

マルチ VPC プライベート接続の要件と制限

マルチ VPC プライベート接続を実行するには、次の MSK クラスター要件に注意してください。

- マルチ VPC プライベート接続は、Apache Kafka 2.7.1 以上でのみサポートされています。MSK クラスターで使用するクライアントが、クラスターと互換性のある Apache Kafka バージョンを実行していることを確認してください。
- マルチ VPC プライベート接続は、IAM、TLS、および SASL/SCRAM の認証タイプをサポートします。認証されていないクラスターは、マルチ VPC プライベート接続を使用できません。
- SASL/SCRAM または mTLS アクセスコントローラー方式を使用している場合は、クラスターに Apache Kafka ACL を設定する必要があります。まず、クラスターの Apache Kafka ACL を設定します。次に、クラスターの設定を更新して、クラスターのプロパティ `allow.everyone.if.no.acl.found` を false に設定します。クラスターの設定を更新する方法については、「the section called “ブローカー設定オペレーション”」を参照してください。IAM アクセス制御を使用している場合、認可ポリシーを適用したり、認可ポリシーを更新したりする場合は、「the section called “IAM アクセスコントロール”」を参照してください。Apache Kafka ACL の詳細については、「the section called “Apache Kafka ACL”」を参照してください。
- マルチ VPC プライベート接続は、t3.small インスタンスタイプをサポートしていません。

- マルチ VPC プライベート接続は、 AWS リージョン間ではサポートされず、同じリージョン内の AWS アカウントでのみサポートされます。
- マルチ VPC プライベート接続を設定するには、クラスターサブネットと同じ数のクライアントサブネットが必要です。また、クライアントサブネットとクラスターサブネットのアベイラビリティーゾーン IDs が同じであることを確認する必要があります。
- Amazon MSK は、ZooKeeper ノードへのマルチ VPC プライベート接続をサポートしていません。

マルチ VPC プライベート接続の使用を開始する

トピック

- ステップ 1: アカウント A の MSK クラスターで、クラスターの IAM 認証スキームのマルチ VPC 接続をオンにする
- ステップ 2: クラスターポリシーを MSK クラスターにアタッチする
- ステップ 3: クライアント管理の VPC 接続を設定するためのクロスアカウントユーザーの操作

このチュートリアルでは、マルチ VPC 接続を使用して Apache Kafka クライアントをクラスターの内部から MSK クラスターにプライベートに接続する方法の例として AWS、一般的なユースケースを使用します。このプロセスでは、クロスアカウントユーザーは、必要なクライアントアクセス許可を含め、MSK マネージド VPC 接続と設定をクライアントごとに作成する必要があります。また、このプロセスでは、MSK クラスター所有者が MSK クラスターで PrivateLink 接続を有効にし、クラスターへのアクセスを制御する認証スキームを選択する必要もあります。

このチュートリアルの各部では、この例に適用するオプションをこちらで選択しています。これは、MSK クラスターまたはクライアント インスタンスを設定するために機能する唯一のオプションであることを意味するものではありません。

このユースケースのネットワーク構成は次のとおりです。

- クロスアカウントユーザー (Kafka クライアント) と MSK クラスターは同じ AWS ネットワーク/リージョン内にありますが、アカウントは異なります。
  - アカウント A の MSK クラスター
  - アカウント B の Kafka クライアント
- クロスアカウントユーザーは IAM 認証スキームを使用して MSK クラスターにプライベート接続します。

このチュートリアルでは、Apache Kafka バージョン 2.7.1 以上で作成された MSK クラスターがプロビジョニング済みであることを前提としています。MSK クラスターは、設定プロセスを開始する前に ACTIVE 状態になっている必要があります。データの損失やダウンタイムが生じないようにするため、マルチ VPC プライベート接続を使用してクラスターに接続するクライアントは、クラスターと互換性のある Apache Kafka バージョンを使用する必要があります。

次の図は、別の AWS アカウントのクライアントに接続された Amazon MSK マルチ VPC 接続のアーキテクチャを示しています。

ステップ 1: アカウント A の MSK クラスターで、クラスターの IAM 認証スキームのマルチ VPC 接続をオンにする

MSK クラスター所有者は、クラスターが作成されて ACTIVE 状態になった後に、MSK クラスターの設定を行う必要があります。

クラスター所有者は、クラスター上でアクティブになるすべての認証スキームについて、ACTIVE なクラスターでマルチ VPC プライベート接続を有効にします。これは、UpdateSecurity API または MSK コンソールを使用して実行できます。IAM、SASL/SCRAM、および TLS 認証スキームは、マルチ VPC プライベート接続をサポートします。マルチ VPC プライベート接続は、認証されていないクラスターでは有効にできません。

このユースケースでは、IAM 認証スキームを使用するようにクラスターを設定します。

> **Note**
>
> SASL/SCRAM 認証スキームを使用するように MSK クラスターを設定する場合、Apache Kafka ACL プロパティ "`allow.everyone.if.no.acl.found=false`" は必須です。「Apache Kafka ACL」を参照してください。

マルチ VPC プライベート接続の設定を更新すると、Amazon MSK はブローカーノードのローリングリブートを開始し、ブローカー設定を更新します。この処理の完了には 30 分以上かかる場合があります。接続が更新されている間、クラスターに他の更新を行うことはできません。

コンソールを使用してアカウント A のクラスターで選択した認証スキームのマルチ VPC を有効にする

1. https://console.aws.amazon.com/msk/ で、クラスターのあるアカウントの Amazon MSK コンソールを開きます。

2. ナビゲーションペインの [MSK クラスター] で [クラスター] を選択し、アカウントのクラスターのリストを表示します。
3. マルチ VPC プライベート接続を設定するクラスターを選択します。クラスターは ACTIVE 状態である必要があります。
4. クラスターの [プロパティ] タブを選択し、[ネットワーク設定] に移動します。
5. [編集] ドロップダウンメニューを選択し、[マルチ VPC 接続をオンにする] を選択します。
6. このクラスターで有効にする認証タイプを 1 つ以上選択します。このユースケースでは、[IAM ロールベースの認証] を選択します。
7. [変更を保存] を選択します。

Example : クラスターでマルチ VPC プライベート接続の認証スキームを有効にする UpdateConnectivity API

MSK コンソールの代わりに、UpdateConnectivity API を使用してマルチ VPC プライベート接続を有効にし、ACTIVE なクラスターで認証スキームを設定できます。次の例は、クラスターで IAM 認証スキームが有効になっていることを示しています。

```
{
  "currentVersion": "K3T4TT2Z381HKD",
  "connectivityInfo": {
    "vpcConnectivity": {
      "clientAuthentication": {
        "sasl": {
          "iam": {
            "enabled": TRUE
            }
        }
      }
    }
  }
}
```

Amazon MSK は、プライベート接続に必要なネットワークインフラストラクチャを作成します。また、Amazon MSK は、プライベート接続を必要とする認証タイプごとに、ブートストラップブローカーエンドポイントの新しいセットも作成します。プレーンテキスト認証スキームはマルチ VPC プライベート接続をサポートしていないことに注意してください。

ステップ 2: クラスターポリシーを MSK クラスターにアタッチする

クラスター所有者は、マルチ VPC プライベート接続を有効にする MSK クラスターにクラスターポリシー (リソースベースのポリシーとも呼ばれます) をアタッチできます。クラスターポリシーは、別のアカウントからクラスターにアクセスするアクセス許可をクライアントに付与します。クラスターポリシーを編集するには、MSK クラスターにアクセスするアクセス許可を必要とするアカウントのアカウント ID が必要です。「Amazon MSK と IAM の連携の仕組み」を参照してください。

クラスター所有者は、クラスターポリシーを MSK クラスターにアタッチすることにより、アカウント B のクロスアカウントユーザーがクラスターのブートストラップブローカーを取得し、アカウント A の MSK クラスターで次のアクションを許可することを許可する必要があります。

- CreateVpcConnection
- GetBootstrapBrokers
- DescribeCluster
- DescribeClusterV2

Example

参考までに、MSK コンソールの IAM ポリシーエディタに表示されるデフォルトポリシーと同様の、基本的なクラスターポリシーの JSON の例を以下に示します。次のポリシーは、クラスター、トピック、およびグループレベルのアクセス許可を付与します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "123456789012"
      },
      "Action": [
        "kafka:CreateVpcConnection",
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka-cluster:*"
```

```
      ],
      "Resource": "arn:aws:kafka:us-east-1:111122223333:cluster/testing/
de8982fa-8222-4e87-8b20-9bf3cdfa1521-2"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "123456789012"
      },
      "Action": "kafka-cluster:*",
      "Resource": "arn:aws:kafka:us-east-1:111122223333:topic/testing/*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "123456789012"
      },
      "Action": "kafka-cluster:*",
      "Resource": "arn:aws:kafka:us-east-1:111122223333:group/testing/*"
    }
  ]
}
```

クラスターポリシーを MSK クラスターにアタッチする

1. Amazon MSK コンソールの [MSK クラスター] で、[クラスター] を選択します。
2. [セキュリティ設定] まで下にスクロールし、[クラスターポリシーの編集] を選択します。
3. コンソールの [クラスターポリシーの編集] 画面で、[マルチ VPC 接続の基本ポリシー] を選択します。
4. [アカウント ID] フィールドに、このクラスターにアクセスするアクセス許可を必要とする各アカウントのアカウント ID を入力します。入力した ID は、表示されたポリシー JSON 構文に自動的にコピーされます。この例のクラスターポリシーでは、アカウント ID は 123456789012 です。
5. [変更を保存] を選択します。

クラスターポリシー API については、「Amazon MSK のリソースベースのポリシー」を参照してください。

ステップ 3: クライアント管理の VPC 接続を設定するためのクロスアカウントユーザーの操作

MSK クラスターとは別のアカウントのクライアント間にマルチ VPC プライベート接続を設定するには、クロスアカウントユーザーがクライアントのマネージド VPC 接続を作成します。この手順を繰り返すことで、複数のクライアントを MSK クラスターに接続できます。このユースケースでは、クライアントを 1 つだけ設定します。

クライアントは、サポートされている認証スキームである、IAM、SASL/SCRAM、または TLS を使用できます。各マネージド VPC 接続に関連付けることができる認証スキームは 1 つのみです。クライアント認証スキームは、クライアントが接続する MSK クラスターで設定する必要があります。

このユースケースでは、アカウント B のクライアントが IAM 認証スキームを使用するようにクライアント認証スキームを設定します。

前提条件

このプロセスには、以下の項目が必要です。

- アカウント A の MSK クラスターでアクションを実行するアクセス許可をアカウント B のクライアントに付与する、以前に作成したクラスターポリシー。
- アカウント B のクライアントにアタッチされた、`kafka:CreateVpcConnection`、`ec2:CreateTags`、`ec2:CreateVPCEndpoint`、`ec2:Descri` のアクションのアクセス許可を付与する ID ポリシー。

Example

参考までに、基本的なクライアント ID ポリシーの JSON の例を以下に示します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kafka:CreateVpcConnection",
        "ec2:CreateTags",
        "ec2:CreateVPCEndpoint",
```

```
        "ec2:DescribeVpcAttribute"
      ],
      "Resource": "*"
    }
  ]
}
```

アカウント B のクライアントのマネージド VPC 接続を作成するには

1. クラスター管理者から、アカウント B のクライアントの接続先となるアカウント A の MSK クラスターのクラスター ARN を取得します。クラスター ARN は、後で使用するため書き留めておきます。
2. クライアントアカウント B の MSK コンソールで、[マネージド VPC 接続] を選択し、[接続の作成] を選択します。
3. [接続設定] ペインで、クラスター ARN をクラスター ARN テキストフィールドに貼り付け、[検証] を選択します。
4. アカウント B のクライアントの [認証タイプ] を選択します。このユースケースでは、クライアント VPC 接続を作成するときに IAM を選択します。
5. クライアントの [VPC] を選択します。
6. 少なくとも 2 つのアベイラビリティーゾーンと、関連付けられたサブネットを選択します。アベイラビリティーゾーン IDs は、 AWS マネジメントコンソールクラスターの詳細から、または DescribeCluster API または describe-cluster AWS CLI コマンドを使用して取得できます。クライアントサブネットに指定するゾーン ID は、クラスターサブネットの ID と一致する必要があります。サブネットの値が見つからない場合は、まず MSK クラスターと同じゾーン ID でサブネットを作成します。
7. この VPC 接続のセキュリティグループを選択します。デフォルトのセキュリティグループを使用できます。セキュリティグループの詳細については、「セキュリティグループを使用してリソースへのトラフィックを制御する」を参照してください。
8. [接続の作成] を選択します。
9. クロスアカウントユーザーの MSK コンソール ([クラスター] の詳細 > [マネージド VPC 接続]) から新しいブートストラップブローカー文字列のリストを取得するには、[クラスター接続文字列] の下に表示されているブートストラップブローカー文字列を確認します。クライアントアカウント B からブートストラップブローカーのリストを表示するには、GetBootstrapBrokers API を呼び出すか、コンソールクラスターの詳細でブートストラップブローカーのリストを表示します。

10. VPC 接続に関連付けられたセキュリティグループを次のように更新します。

    a. PrivateLink VPC のインバウンドルールを設定して、アカウント B ネットワークからの IP 範囲のすべてのトラフィックを許可します。

    b. (オプション) MSK クラスターへの [アウトバウンドルール] 接続を設定します。VPC コンソールで [セキュリティグループ] を選択し、[アウトバウンドルールの編集] を行い、ポート範囲 14001 ~ 14100 の [カスタム TCP トラフィック] のルールを追加します。マルチ VPC ネットワークロードバランサーは、14001 ~ 14100 のポート範囲をリッスンしています。「Network Load Balancer」を参照してください。

11. マルチ VPC プライベート接続用の新しいブートストラップブローカーを使用してアカウント A の MSK クラスターに接続するように、アカウント B のクライアントを設定します。「データを生成および消費する」を参照してください。

認可が完了すると、Amazon MSK は指定された VPC および認可スキームごとにマネージド VPC 接続を作成します。選択したセキュリティグループは各接続に関連付けられます。このマネージド VPC 接続は、ブローカーにプライベート接続するように Amazon MSK によって設定されます。新しいブートストラップブローカーのセットを使用して、Amazon MSK クラスターにプライベート接続できます。

クラスターの認可スキームの更新

マルチ VPC プライベート接続は、複数の認可スキーム (SASL/SCRAM、IAM、および TLS) をサポートしています。クラスター所有者は、1 つ以上の認証スキームのプライベート接続を有効/無効にできます。このアクションを実行するには、クラスターが ACTIVE 状態である必要があります。

Amazon MSK コンソールを使用して認証スキームを有効にするには

1. 編集するクラスターの AWS Management Consoleで Amazon MSK コンソールを開きます。
2. ナビゲーションペインの [MSK クラスター] で [クラスター] を選択し、アカウントのクラスターのリストを表示します。
3. 編集するクラスターを選択します。クラスターは ACTIVE 状態である必要があります。
4. クラスターの [プロパティ] タブを選択し、[ネットワーク設定] に移動します。
5. [編集] ドロップダウンメニューを選択し、[マルチ VPC 接続をオンにする] を選択して新しい認証スキームを有効にします。
6. このクラスターで有効にする認証タイプを 1 つ以上選択します。
7. [選択をオンにする] を選択します。

新しい認証スキームを有効にするときは、新しい認証スキーム用に新しいマネージド VPC 接続を作成し、新しい認証スキーム固有のブートストラップブローカーを使用するようにクライアントを更新する必要もあります。

Amazon MSK コンソールを使用して認証スキームを無効にするには

> **Note**
>
> 認証スキームのマルチ VPC プライベート接続を無効にすると、マネージド VPC 接続を含むすべての接続関連インフラストラクチャが削除されます。

認証スキームのマルチ VPC プライベート接続を無効にすると、クライアント側の既存の VPC 接続が INACTIVE に変わり、クラスター側の Privatelink インフラストラクチャ (クラスター側のマネージド VPC 接続を含む) が削除されます。クロスアカウントユーザーは、非アクティブな VPC 接続のみを削除できます。クラスターでプライベート接続を再度有効にした場合、クロスアカウントユーザーはクラスターへの新しい接続を作成する必要があります。

1. AWS Management Console で Amazon MSK コンソールを開きます。
2. ナビゲーションペインの [MSK クラスター] で [クラスター] を選択し、アカウントのクラスターのリストを表示します。
3. 編集するクラスターを選択します。クラスターは ACTIVE 状態である必要があります。
4. クラスターの [プロパティ] タブを選択し、[ネットワーク設定] に移動します。
5. [編集] ドロップダウンメニューを選択し、[マルチ VPC 接続をオフにする] を選択します (認証スキームを無効にします)。
6. このクラスターで無効にする認証タイプを 1 つ以上選択します。
7. [選択をオフにする] を選択します。

Example API で認証スキームを有効/無効にするには

MSK コンソールの代わりに、UpdateConnectivity API を使用してマルチ VPC プライベート接続を有効にし、ACTIVE なクラスターで認証スキームを設定できます。次の例は、クラスターで SASL/SCRAM 認証スキームと IAM 認証スキームが有効になっていることを示しています。

新しい認証スキームを有効にするときは、新しい認証スキーム用に新しいマネージド VPC 接続を作成し、新しい認証スキーム固有のブートストラップブローカーを使用するようにクライアントを更新する必要もあります。

認証スキームのマルチ VPC プライベート接続を無効にすると、クライアント側の既存の VPC 接続が INACTIVE に変わり、クラスター側の Privatelink インフラストラクチャ (マネージド VPC 接続を含む) が削除されます。クロスアカウントユーザーは、非アクティブな VPC 接続のみを削除できます。クラスターでプライベート接続を再度有効にした場合、クロスアカウントユーザーはクラスターへの新しい接続を作成する必要があります。

```
Request:
{
  "currentVersion": "string",
  "connnectivityInfo": {
    "publicAccess": {
      "type": "string"
    },
    "vpcConnectivity": {
      "clientAuthentication": {
        "sasl": {
          "scram": {
            "enabled": TRUE
          },
          "iam": {
            "enabled": TRUE
          }
        },
        "tls": {
          "enabled": FALSE
        }
      }
    }
  }
}

Response:
{
  "clusterArn": "string",
  "clusterOperationArn": "string"
}
```

Amazon MSK クラスターへのマネージド VPC 接続の拒否

クラスター管理者アカウントの Amazon MSK コンソールから、クライアント VPC 接続を拒否できます。拒否するには、クライアント VPC 接続が AVAILABLE 状態である必要があります。クラスターへの接続が許可されなくなったクライアントからのマネージド VPC 接続を拒否することが必要

な場合があります。新しいマネージド VPC 接続がクライアントに接続されないようにするには、クラスターポリシーでクライアントへのアクセスを拒否します。拒否された接続は、接続の所有者によって削除されるまで料金が発生します。「Amazon MSK クラスターへのマネージド VPC 接続の削除」を参照してください。

MSK コンソールを使用してクライアント VPC 接続を拒否するには

1. AWS Management Console で Amazon MSK コンソールを開きます。
2. ナビゲーションペインで [クラスター] を選択し、[ネットワーク設定] > [クライアント VPC 接続] リストまでスクロールします。
3. 拒否する接続を選択し、[クライアント VPC 接続を拒否する] を選択します。
4. 選択したクライアント VPC 接続を拒否することを確認します。

API を使用してマネージド VPC 接続を拒否するには、`RejectClientVpcConnection` API を使用します。

Amazon MSK クラスターへのマネージド VPC 接続の削除

クロスアカウントユーザーは、クライアントアカウントコンソールから MSK クラスターのマネージド VPC 接続を削除できます。クラスター所有者ユーザーはマネージド VPC 接続を所有していないため、クラスター管理者アカウントから接続を削除することはできません。VPC 接続を削除すると、料金は発生しなくなります。

MSK コンソールを使用してマネージド VPC 接続を削除するには

1. クライアントアカウントから、AWS Management Consoleで Amazon MSK コンソールを開きます。
2. ナビゲーションペインで、[マネージド VPC 接続] を選択します。
3. 接続リストから、削除する接続を選択します。
4. VPC 接続を削除することを確認します。

API を使用してマネージド VPC 接続を削除するには、`DeleteVpcConnection` API を使用します。

マルチ VPC プライベート接続のアクセス許可

このセクションでは、マルチ VPC プライベート接続機能を使用するクライアントとクラスターに必要なアクセス許可の概要を示します。マルチ VPC プライベート接続では、クライアント管理者

が、MSK クラスターへのマネージド VPC 接続を持つ各クライアントに対するアクセス許可を作成する必要があります。また、MSK クラスター管理者が MSK クラスターで PrivateLink 接続を有効にし、クラスターへのアクセスを制御する認証スキームを選択する必要もあります。

クラスターの認証タイプとトピックのアクセス許可

MSK クラスターで有効になっている認証スキームのマルチ VPC プライベート接続機能を有効にします。「マルチ VPC プライベート接続の要件と制限」を参照してください。SASL/SCRAM 認証スキームを使用するように MSK クラスターを設定する場合、Apache Kafka ACL プロパティ `allow.everyone.if.no.acl.found=false` は必須です。クラスターに Apache Kafka ACL を設定した後、クラスターの設定を更新して、クラスターのプロパティ `allow.everyone.if.no.acl.found` を false に設定します。クラスターの設定を更新する方法については、「ブローカー設定オペレーション」を参照してください。

クロスアカウントクラスターポリシーのアクセス許可

Kafka クライアントが MSK クラスターとは異なる AWS アカウントにある場合は、クライアントルートユーザーにクロスアカウント接続を許可するクラスターベースのポリシーを MSK クラスターにアタッチします。MSK コンソールの IAM ポリシーエディタ (クラスターの [セキュリティ設定] > [クラスターポリシーの編集]) を使用してマルチ VPC クラスターポリシーを編集するか、以下の API を使用してクラスターポリシーを管理できます。

PutClusterPolicy

クラスターポリシーをクラスターにアタッチします。この API を使用して、指定した MSK クラスターポリシーを作成または更新できます。ポリシーを更新する場合、リクエストペイロードには currentVersion フィールドが必要です。

GetClusterPolicy

クラスターにアタッチされているクラスターポリシードキュメントの JSON テキストを取得します。

DeleteClusterPolicy

クラスターポリシーを削除します。

MSK コンソールの IAM ポリシーエディタに表示されるポリシーと同様の、基本的なクラスターポリシーの JSON の例を以下に示します。次のポリシーは、クラスター、トピック、およびグループレベルのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "AWS": [
                "123456789012"
            ]
        },
        "Action": [
            "kafka-cluster:*",
            "kafka:CreateVpcConnection",
            "kafka:GetBootstrapBrokers",
            "kafka:DescribeCluster",
            "kafka:DescribeClusterV2"
        ],
        "Resource": [
            "arn:aws:kafka:us-east-1:123456789012:cluster/testing/
de8982fa-8222-4e87-8b20-9bf3cdfa1521-2",
            "arn:aws:kafka:us-east-1:123456789012:topic/testing/*",
            "arn:aws:kafka:us-east-1:123456789012:group/testing/*"
        ]
    }]
}
```

MSK クラスターへのマルチ VPC プライベート接続のクライアントアクセス許可

Kafka クライアントと MSK クラスターの間にマルチ VPC プライベート接続をセットアップするには、クライアントでの `kafka:CreateVpcConnection`、`ec2:CreateTags`、および `ec2:CreateVPCEndpoint` のアクションのアクセス許可を付与する ID ポリシーがクライアントにアタッチされている必要があります。参考までに、基本的なクライアント ID ポリシーの JSON の例を以下に示します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "kafka:CreateVpcConnection",
        "ec2:CreateTags",
        "ec2:CreateVPCEndpoint"
      ],
      "Resource": "*"
    }
  ]
}
```

ポート情報

Amazon MSK がクライアントマシンと通信できるように、以下のポート番号を使用します。

- プレーンテキストでブローカーと通信するには、ポート 9092 を使用します。
- TLS 暗号化を使用してブローカーと通信するには、内部からのアクセスにポート 9094 AWS を使用し、パブリックアクセスにポート 9194 を使用します。
- SASL/SCRAM を使用してブローカーと通信するには、内部からのアクセスにポート 9096 AWS を使用し、パブリックアクセスにポート 9196 を使用します。
- を使用するように設定されたクラスター内のブローカーと通信するにはthe section called “IAM アクセスコントロール”、内部からのアクセスにポート 9098 AWS を使用し、パブリックアクセスにポート 9198 を使用します。
- TLS 暗号化を使用して Apache ZooKeeper と通信するには、ポート 2182 を使用します。Apache ZooKeeper ノードは、デフォルトでポート 2181 を使用します。

Amazon MSK クラスターのブートストラップブローカーを取得する

ブートストラップブローカーとは、Apache Kafka クライアントで Amazon MSK クラスターに接続するために使用できるブローカーのリストを指します。このリストにクラスター内のすべてのブローカーが含まれているとは限りません。ブートストラップブローカーは AWS Management Console、、 AWS CLI、または Amazon MSK API を使用して取得できます。

トピック

- を使用してブートストラップブローカーを取得する AWS Management Console
- を使用してブートストラップブローカーを取得する AWS CLI

- API を使用してブートストラップブローカーを取得する

を使用してブートストラップブローカーを取得する AWS Management Console

このプロセスでは、 を使用してクラスターのブートストラップブローカーを取得する方法について説明します AWS Management Console。ブートストラップブローカーとは、Apache Kafka クライアントがクラスターに接続するための出発点として使用できるブローカーのリストを指します。このリストには、クラスター内のすべてのブローカーが含まれているとは限りません。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. このテーブルには、このアカウントにある現在のリージョンのすべてのクラスターが表示されます。説明を表示するには、クラスターの名前を選択します。
3. [Cluster summary] (クラスターの概要) ページで、[View client information] (クライアント情報の表示) を選択します。これにより、ブートストラップブローカーと Apache ZooKeeper 接続文字列が表示されます。

を使用してブートストラップブローカーを取得する AWS CLI

*ClusterArn*をクラスターの作成時に取得したAmazon リソースネーム (ARN) に置き換えて、次のコマンドを実行します。クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。詳細については、「the section called “クラスターを一覧表示する”」を参照してください。

```
aws kafka get-bootstrap-brokers --cluster-arn ClusterArn
```

the section called “IAM アクセスコントロール” を使用するMSK クラスターでは、コマンドの出力は以下の JSON の例のようになります。

```
{
    "BootstrapBrokerStringSaslIam": "b-1.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9098,b-2.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9098"
}
```

次の例は、パブリックアクセスを有効にしたクラスターでのブートストラップブローカーを示しています。パブリックアクセスBootstrapBrokerStringPublicSaslIamには を使用し、内部からのアクセスには BootstrapBrokerStringSaslIam文字列を使用します AWS。

```
{
    "BootstrapBrokerStringPublicSaslIam": "b-2-public.myTestCluster.v4ni96.c2.kafka-
beta.us-east-1.amazonaws.com:9198,b-1-public.myTestCluster.v4ni96.c2.kafka-
beta.us-east-1.amazonaws.com:9198,b-3-public.myTestCluster.v4ni96.c2.kafka-beta.us-
east-1.amazonaws.com:9198",
    "BootstrapBrokerStringSaslIam": "b-2.myTestCluster.v4ni96.c2.kafka-
beta.us-east-1.amazonaws.com:9098,b-1.myTestCluster.v4ni96.c2.kafka-beta.us-
east-1.amazonaws.com:9098,b-3.myTestCluster.v4ni96.c2.kafka-beta.us-
east-1.amazonaws.com:9098"
}
```

ブートストラップブローカー文字列には、MSK クラスターをデプロイするアベイラビリティーゾーン全体の 3 つのブローカーが含まれている必要があります (2 つのブローカーのみが使用可能である場合を除く)。

API を使用してブートストラップブローカーを取得する

API を使用してブートストラップブローカーを取得するには、「GetBootstrapBrokers」を参照してください。

Amazon MSK プロビジョンドクラスターをモニタリングする

Amazon MSK は、Amazon MSK プロビジョンドクラスターのステータスのモニタリングに役立ついくつかの方法があります。

- Amazon MSK は、Apache Kafka メトリクスを収集し、それらを表示できる Amazon CloudWatch に送信します。Amazon MSK が表示するものを含む Apache Kafka メトリクスの詳細については、Apache Kafka ドキュメントのモニタリングを参照してください。
- オープンソースのモニタリングアプリケーションである Prometheus を使用して MSK クラスターをモニタリングすることもできます。Prometheus の詳細については、Prometheus のドキュメントの「概要」を参照してください。Prometheus で MSK プロビジョンドクラスターをモニタリングする方法については、「」を参照してくださいthe section called “Prometheus でモニタリングする”。
- (標準ブローカーのみ) Amazon MSK は、プロビジョンドクラスターがストレージ容量の制限に近づいたときにストレージ容量アラートを自動的に送信することで、ディスクストレージ容量をモニタリングするのに役立ちます。アラートには、検出された問題に対処するための最善の手順に関する推奨事項も記載されています。これにより、ディスク容量の問題が深刻化する前に特定し、迅速に解決できます。Amazon MSK は、これらのアラートを Amazon MSK コンソール、Amazon EventBridge、および AWS アカウントの E メール連絡先に自動的に送信します。 AWS Health

Dashboardストレージ容量アラートの引き上げの詳細については、Amazon MSK ストレージ容量アラートを使用する を参照してください。

トピック

- CloudWatch を使用して Amazon MSK メトリクスを表示する
- CloudWatch で標準ブローカーをモニタリングするための Amazon MSK メトリクス
- CloudWatch で Express ブローカーをモニタリングするための Amazon MSK メトリクス
- Prometheus で MSK プロビジョンドクラスターをモニタリングする
- コンシューマーラグをモニタリングする
- Amazon MSK ストレージ容量アラートを使用する

CloudWatch を使用して Amazon MSK メトリクスを表示する

CloudWatch コンソール、コマンドライン、または CloudWatch API を使用して、Amazon MSK のメトリクスをモニタリングできます。次の手順は、これらのさまざまなメソッドを使用してメトリクスにアクセスする方法を示しています。

CloudWatch コンソールを使用してメトリクスにアクセスするには

にサインイン AWS Management Console し、https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。

1. ナビゲーションペインで [Metrics (メトリクス)] を選択してください。
2. All metrics (すべてのメトリクス) タブを選択してから、AWS/Kafkaを選択します。
3. トピックレベルのメトリクスを表示するには、ブローカーレベルのメトリクスの場合は、[Topic, Broker ID, Cluster Name (トピック、ブローカー ID、クラスター名)] を選択し、クラスターレベルのメトリクスの場合は [Broker ID, Cluster Name (ブローカー ID、クラスター名)]を選択して、[Cluster Name (クラスター名)] を選択します。
4. (オプション) グラフペインで、統計と期間を選択し、これらの設定を使用して CloudWatch アラームを作成します。

を使用してメトリクスにアクセスするには AWS CLI

list-metrics および get-metric-statistics コマンドを使用します。

CloudWatch CLI を使用してメトリクスにアクセスするには

mon-list-metrics コマンドと mon-get-stats コマンドを使用します。

CloudWatch API を使用してメトリクスにアクセスするには

ListMetrics オペレーションと GetMetricStatistics オペレーションを使用します。

CloudWatch で標準ブローカーをモニタリングするための Amazon MSK メトリクス

Amazon MSK は Amazon CloudWatch と統合されているため、MSK Standard ブローカーの CloudWatch メトリクスを収集、表示、分析できます。MSK プロビジョンドクラスター用に設定したメトリクスは、自動的に収集され、1 分間隔で CloudWatch にプッシュされます。MSK プロビジョンドクラスターのモニタリングレベルは、DEFAULT、、PER_BROKERPER_TOPIC_PER_BROKER、または のいずれかに設定できますPER_TOPIC_PER_PARTITION。次のセクションの表は、各モニタリングレベルから利用できるすべてのメトリクスを示しています。

> **Note**
>
> CloudWatch モニタリング用の一部の Amazon MSK メトリクスの名前は、バージョン 3.6.0 以降で変更されています。これらのメトリクスをモニタリングするには、新しい名前を使用してください。名前が変更されたメトリクスの場合、以下の表にはバージョン 3.6.0 以降で使用されていた名前と、その後にバージョン 2.8.2.tiered の名前が続きます。

DEFAULT レベルのメトリクスは無料です。他のメトリクスの料金については、Amazon CloudWatch の料金ページで説明されています。

DEFAULT レベルモニタリング

次の表で説明するメトリクスは、DEFAULT モニタリングレベルで使用できます。これらは無料です。

名前	表示可能なタイミング	ディメンション	説明
ActiveControllerCount	クラスターが ACTIVE 状態になった後。	クラスター名	クラスターごとに 1 つのコントローラーだけをアクティブにする必要があります。

名前	表示可能なタイミング	ディメンション	説明
BurstBalance	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	クラスター内の EBS ボリュームの入出力バーストクレジットの残高。これを使用して、レイテンシーまたはスループットの低下を調査します。 ボリュームのベースラインパフォーマンスが最大バーストパフォーマンスより高い場合、EBS ボリュームの BurstBalance は報告されません。詳細については、「I/O クレジットおよびバーストパフォーマンス」を参照してください。
BytesInPerSec	トピックを作成した後。	クラスター名、ブローカー ID、トピック	クライアントから受信した 1 秒あたりのバイト数。このメトリクスは、ブローカーごとおよびトピックごとに利用できます。
BytesOutPerSec	トピックを作成した後。	クラスター名、ブローカー ID、トピック	クライアントに送信された 1 秒あたりのバイト数。このメトリクスは、ブローカーごとおよびトピックごとに利用できます。

名前	表示可能なタイミング	ディメンション	説明
ClientConnectionCount	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID、クライアント認証	アクティブな認証済みクライアント接続の数。
ConnectionCount	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	アクティブな認証済み接続、未認証接続、およびブローカー間接続の数。
CPUCreditBalance	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーの起動後に蓄積した獲得 CPU クレジットの数。クレジットは獲得後にクレジット残高に蓄積され、消費されるとクレジット残高から削除されます。CPU クレジット残高が不足した場合、クラスターのパフォーマンスに悪影響を与える可能性があります。CPU ロードを軽減するためのステップを実行できます。例えば、クライアント要求の数を減らしたり、ブローカータイプを M5 ブローカータイプに更新したりできます。

名前	表示可能なタイミング	ディメンション	説明
CpuIdle	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	CPU アイドル時間の割合。
CpuIoWait	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	保留中のディスクオペレーション中の CPU アイドル時間の割合 (%)。
CpuSystem	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	カーネルスペースの CPU の割合。
CpuUser	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ユーザースペースの CPU の割合。

名前	表示可能なタイミング	ディメンション	説明
GlobalPartitionCount	クラスターが ACTIVE 状態になった後。	クラスター名	クラスター内のすべてのトピック全体のパーティション数 (レプリカを除く)。GlobalPartitionCount にはレプリカが含まれていないため、トピックのレプリケーション係数が 1 より大きい場合、PartitionCount 値の合計が GlobalPartitionCount より大きくなる可能性があります。
GlobalTopicCount	クラスターが ACTIVE 状態になった後。	クラスター名	クラスター内のすべてのブローカーのトピックの合計数。
EstimatedMaxTimeLag	コンシューマーグループがトピックから消費した後。	クラスター名、コンシューマーグループ、トピック	MaxOffsetLag を排出するための推定時間 (秒単位)。
KafkaAppLogsDiskUsed	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	アプリケーションログに使用されるディスク領域の割合。

名前	表示可能なタイミング	ディメンション	説明
KafkaData LogsDiskUsed (Cluster Name, Broker ID ディメンション)	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	データログに使用されるディスク容量の割合。
LeaderCount	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	レプリカを含まない、ブローカーごとのパーティションのリーダーの総数。
MaxOffsetLag	コンシューマーグループがトピックから消費した後。	クラスター名、コンシューマーグループ、トピック	トピック内のすべてのパーティションにおける最大オフセットラグ。
MemoryBuffered	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーのバッファーメモリーのサイズ (バイト単位)。

名前	表示可能なタイミング	ディメンション	説明
MemoryCached	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーのキャッシュメモリーのサイズ (バイト単位)。
MemoryFree	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーが使用可能な空きメモリのサイズ (バイト単位) 。
HeapMemoryAfterGC	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ガベージコレクション後に使用されている合計ヒープメモリの割合 (%)。
MemoryUsed	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーに使用されているメモリのサイズ (バイト単位) 。

名前	表示可能なタイミング	ディメンション	説明
MessagesInPerSec	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーの 1 秒あたりの受信メッセージ数。
NetworkRxDropped	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ドロップされた受信パッケージの数。
NetworkRxErrors	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーのネットワーク受信エラーの数。
NetworkRxPackets	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーによって受信されたパケットの数。

名前	表示可能なタイミング	ディメンション	説明
NetworkTx Dropped	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ドロップされた送信パッケージの数。
NetworkTx Errors	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーのネットワーク送信エラーの数。
NetworkTx Packets	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーによって送信されたパケットの数。
OfflinePa rtitionsCount	クラスターが ACTIVE 状態になった後。	クラスター名	クラスター内でオフラインになっているパーティションの合計数。
PartitionCount	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	レプリカを含む、ブローカーごとのトピックパーティションの総数。

名前	表示可能なタイミング	ディメンション	説明
ProduceTotalTimeMsMean	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ミリ秒単位の平均生成時間。
RequestBytesMean	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーのリクエストバイトの平均数。
RequestTime	リクエストスロットリングが適用された後。	クラスター名、ブローカー ID	ブローカーネットワークおよび I/O スレッドでリクエストを処理するのに費やされた平均時間 (ミリ秒)。
RootDiskUsed	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーが使用するルートディスクの割合。

名前	表示可能なタイミング	ディメンション	説明
SumOffsetLag	コンシューマーグループがトピックから消費した後。	クラスター名、コンシューマーグループ、トピック	トピック内のすべてのパーティションの集計オフセットラグ。
SwapFree	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーで使用可能なスワップメモリのサイズ (バイト単位) 。
SwapUsed	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーに使用されているスワップメモリのサイズ (バイト単位) 。
TrafficShaping	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ネットワーク割り当てを超えたためにシェイピングされた (ドロップされた、またはキューに入れられた) パケットの数を示す高レベルのメトリクス。PER_BROKER メトリクスを使用すると、より詳細な情報を利用できます。

名前	表示可能なタイミング	ディメンション	説明
UnderMinIsrPartitionCount	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーの minIsr 未満パーティションの数。
UnderReplicatedPartitions	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーのレプリケートされていないパーティションの数。
UserPartitionExists	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーにユーザー所有のパーティションが存在することを示すブールメトリクス。値 1 は、ブローカーにパーティションが存在することを示します。
ZooKeeperRequestLatencyMsMean	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ZooKeeper ベースのクラスターの場合。ブローカーからの Apache ZooKeeper リクエストの平均レイテンシー (ミリ秒)。

名前	表示可能なタイミング	ディメンション	説明
ZooKeeper SessionState	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ZooKeeper ベースのクラスターの場合。ブローカーの ZooKeeper セッションの接続状態は、次のいずれかになります。NOT_CONNECTED: '0.0'、ASSOCIATING: '0.1'、CONNECTING: '0.5'、CONNECTEDREADONLY: '0.8'、CONNECTED: '1.0'、CLOSED: '5.0'、AUTH_FAILED: '10.0'。

PER_BROKER レベルモニタリング

モニタリングレベルを PER_BROKER に設定すると、すべての DEFAULT レベルメトリクスに加えて、次の表で説明するメトリクスが表示されます。次の表に示すメトリクスに対して料金をお支払いいただきますが、DEFAULT レベルメトリクスは引き続き無料です。この表のメトリクスには、クラスター名、ブローカー ID のディメンションがあります。

名前	表示可能なタイミング	説明
BwInAllowanceExceeded	クラスターが ACTIVE 状態になった後。	インバウンド集約帯域幅がブローカーの最大値を超えたために形成されたパケットの数。
BwOutAllowanceExceeded	クラスターが ACTIVE 状態になった後。	アウトバウンド集約帯域幅がブローカーの最大値を超えたために形成されたパケットの数。
ConntrackAllowanceExceeded	クラスターが ACTIVE 状態になった後。	接続追跡がブローカーの最大値を超えたために形成されたパケットの数。接続追跡は、確立された各接続を追跡するセキュリティグループに関連してお

名前	表示可能なタイミング	説明
		り、リターンパケットが期待どおりに配信されるようにします。
ConnectionCloseRate	クラスターが ACTIVE 状態になった後。	リスナーごとの 1 秒あたりに閉じられた接続の数。この数はリスナーごとに集計され、クライアントリスナー用にフィルタリングされます。
ConnectionCreationRate	クラスターが ACTIVE 状態になった後。	リスナーごとに 1 秒あたりに確立された新しい接続の数。この数はリスナーごとに集計され、クライアントリスナー用にフィルタリングされます。
CpuCreditUsage	クラスターが ACTIVE 状態になった後。	ブローカーで消費された CPU クレジットの数。CPU クレジット残高が不足した場合、クラスターのパフォーマンスに悪影響を与える可能性があります。CPU ロードを軽減するためのステップを実行できます。例えば、クライアント要求の数を減らしたり、ブローカータイプを M5 ブローカータイプに更新したりできます。
FetchConsumerLocalTimeMsMean	プロデューサー/コンシューマーがいる後。	コンシューマーのリクエストがリーダーで処理される平均時間 (ミリ秒) 。
FetchConsumerRequestQueueTimeMsMean	プロデューサー/コンシューマーがいる後。	コンシューマーのリクエストがリクエストキューで待機する平均時間 (ミリ秒) 。
FetchConsumerResponseQueueTimeMsMean	プロデューサー/コンシューマーがいる後。	コンシューマーのリクエストが応答キューで待機する平均時間 (ミリ秒) 。

名前	表示可能なタイミング	説明
FetchConsumerResponseSendTimeMsMean	プロデューサー/コンシューマーがいる後。	コンシューマーが応答を送信するための平均時間（ミリ秒）。
FetchConsumerTotalTimeMsMean	プロデューサー/コンシューマーがいる後。	コンシューマーがブローカーからデータを取得するのに費やす平均合計時間（ミリ秒）。
FetchFollowerLocalTimeMsMean	プロデューサー/コンシューマーがいる後。	リーダーでフォロワーのリクエストが処理される平均時間（ミリ秒）。
FetchFollowerRequestQueueTimeMsMean	プロデューサー/コンシューマーがいる後。	フォロワーのリクエストがリクエストキューで待機する平均時間（ミリ秒）。
FetchFollowerResponseQueueTimeMsMean	プロデューサー/コンシューマーがいる後。	フォロワーの要求が応答キューで待機する平均時間（ミリ秒）。
FetchFollowerResponseSendTimeMsMean	プロデューサー/コンシューマーがいる後。	フォロワーが応答を送信するまでの平均時間 (ミリ秒)。
FetchFollowerTotalTimeMsMean	プロデューサー/コンシューマーがいる後。	フォロワーがブローカーからデータを取得するのに費やす平均合計時間（ミリ秒）。
FetchMessageConversionsPerSec	トピックを作成した後。	ブローカーの 1 秒あたりのフェッチメッセージ変換回数。
FetchThrottleByteRate	帯域幅スロットリングが適用された後。	1 秒あたりのスロットルバイト数。
FetchThrottleQueueSize	帯域幅スロットリングが適用された後。	スロットルキュー内のメッセージ数。

名前	表示可能なタイミング	説明
FetchThrottleTime	帯域幅スロットリングが適用された後。	フェッチスロットルの平均時間（ミリ秒単位）。
IAMNumberOfConnectionRequests	クラスターが ACTIVE 状態になった後。	1 秒あたりの IAM 認証リクエスト数。
IAMTooManyConnections	クラスターが ACTIVE 状態になった後。	試行された 100 を超える接続の数。0 は、接続数が制限内であることを意味します。>0 の場合、スロットル制限を超えているため、接続数を減らす必要があります。
NetworkProcessorAvgIdlePercent	クラスターが ACTIVE 状態になった後。	ネットワークプロセッサがアイドル状態の平均時間の割合。
PpsAllowanceExceeded	クラスターが ACTIVE 状態になった後。	双方向 PPS がブローカーの最大値を超えたために形成されたパケットの数。
ProduceLocalTimeMsMean	クラスターが ACTIVE 状態になった後。	リーダーでリクエストが処理される平均時間 (ミリ秒単位)。
ProduceMessageConversionsPerSec	トピックを作成した後。	ブローカーの 1 秒あたりの生成メッセージ変換回数。
ProduceMessageConversionsTimeMsMean	クラスターが ACTIVE 状態になった後。	メッセージ形式の変換に費やされた平均時間（ミリ秒）。
ProduceRequestQueueTimeMsMean	クラスターが ACTIVE 状態になった後。	リクエストメッセージがキューに費やした平均時間（ミリ秒）。

名前	表示可能なタイミング	説明
ProduceResponseQueueTimeMsMean	クラスターが ACTIVE 状態になった後。	応答メッセージがキューに費やした平均時間（ミリ秒）。
ProduceResponseSendTimeMsMean	クラスターが ACTIVE 状態になった後。	応答メッセージの送信に費やされた平均時間（ミリ秒）。
ProduceThrottleByteRate	帯域幅スロットリングが適用された後。	1 秒あたりのスロットルバイト数。
ProduceThrottleQueueSize	帯域幅スロットリングが適用された後。	スロットルキュー内のメッセージ数。
ProduceThrottleTime	帯域幅スロットリングが適用された後。	平均生成スロットル時間（ミリ秒単位）。
ProduceTotalTimeMsMean	クラスターが ACTIVE 状態になった後。	ミリ秒単位の平均生成時間。
RemoteFetchBytesPerSec (RemoteBytesInPerSec in v2.8.2.tiered)	プロデューサー/コンシューマーが出現した後。	コンシューマーフェッチへのレスポンスで階層型ストレージから転送された合計バイト数。このメトリクスには、ダウンストリームのデータ転送トラフィックの一因となるすべてのトピックパーティションが含まれます。カテゴリ: トラフィックとエラー率。これは KIP-405 メトリクスです。

名前	表示可能なタイミング	説明
RemoteCopyBytesPerSec (RemoteBytesOutPerSec in v2.8.2.tiered)	プロデューサー/コンシューマーが出現した後。	階層型ストレージに転送された合計バイト数 (ログセグメント、インデックス、その他の補助ファイルからのデータを含む)。このメトリクスには、アップストリームのデータ転送トラフィックの一因となるすべてのトピックパーティションが含まれます。カテゴリ: トラフィックとエラー率。これは KIP-405 メトリクスです。
RemoteLogManagerTasksAvgIdlePercent	クラスターが ACTIVE 状態になった後。	リモートログマネージャーがアイドル状態であった時間の割合 (%) の平均。リモートログマネージャーは、ブローカーから階層型ストレージにデータを転送します。カテゴリ: 内部アクティビティ。これは KIP-405 メトリクスです。
RemoteLogReaderAvgIdlePercent	クラスターが ACTIVE 状態になった後。	リモートログリーダーがアイドル状態であった時間の割合 (%) の平均。リモートログリーダーは、コンシューマーフェッチへのレスポンスで、リモートストレージからブローカーにデータを転送します。カテゴリ: 内部アクティビティ。これは KIP-405 メトリクスです。
RemoteLogReaderTaskQueueSize	クラスターが ACTIVE 状態になった後。	階層型ストレージからの読み取りを担当し、スケジュール待ちのタスクの数。カテゴリ: 内部アクティビティ。これは KIP-405 メトリクスです。

名前	表示可能なタイミング	説明
RemoteFetchErrorsPerSec (RemoteReadErrorPerSec in v2.8.2.tiered)	クラスターが ACTIVE 状態になった後。	指定されたブローカーがコンシューマーフェッチへのレスポンスでデータを取得するために階層型ストレージに送信した読み取りリクエストに対するレスポンスでの合計エラー率。このメトリクスには、ダウンストリームのデータ転送トラフィックの一因となるすべてのトピックパーティションが含まれます。カテゴリ: トラフィックとエラー率。これは KIP-405 メトリクスです。
RemoteFetchRequestsPerSec (RemoteReadRequestsPerSec in v2.8.2.tiered)	クラスターが ACTIVE 状態になった後。	指定されたブローカーがコンシューマーフェッチへのレスポンスでデータを取得するために階層型ストレージに送信した読み取りリクエストの合計数。このメトリクスには、ダウンストリームのデータ転送トラフィックの一因となるすべてのトピックパーティションが含まれます。カテゴリ: トラフィックとエラー率。これは KIP-405 メトリクスです。
RemoteCopyErrorsPerSec (RemoteWriteErrorPerSec in v2.8.2.tiered)	クラスターが ACTIVE 状態になった後。	指定されたブローカーがデータをアップストリームに転送するために階層型ストレージに送信した書き込みリクエストに対するレスポンスでの合計エラー率。このメトリクスには、アップストリームのデータ転送トラフィックの一因となるすべてのトピックパーティションが含まれます。カテゴリ: トラフィックとエラー率。これは KIP-405 メトリクスです。

名前	表示可能なタイミング	説明
RemoteLogSizeBytes	クラスターが ACTIVE 状態になった後。	リモート階層に保存されているバイト数。 このメトリクスは、Amazon MSK の Apache Kafka バージョン 3.7.x の階層型ストレージクラスターで使用できます。
ReplicationBytesInPerSec	トピックを作成した後。	他のブローカーから受信した 1 秒あたりのバイト数。
ReplicationBytesOutPerSec	トピックを作成した後。	他のブローカーに送信された 1 秒あたりのバイト数。
RequestExemptFromThrottleTime	リクエストスロットリングが適用された後。	ブローカーネットワークおよび I/O スレッドで、スロットリングから除外されたリクエストを処理するのに費やされた平均時間 (ミリ秒)。
RequestHandlerAvgIdlePercent	クラスターが ACTIVE 状態になった後。	リクエストハンドラーのスレッドがアイドル状態の平均時間の割合。
RequestThrottleQueueSize	リクエストスロットリングが適用された後。	スロットルキュー内のメッセージ数。
RequestThrottleTime	リクエストスロットリングが適用された後。	リクエストスロットルの平均時間 (ミリ秒単位) 。
TcpConnections	クラスターが ACTIVE 状態になった後。	SYN フラグが設定された着信および発信 TCP セグメントの数を表示します。

名前	表示可能なタイミング	説明
RemoteCopyLagBytes (TotalTierBytesLag in v2.8.2.tiered)	トピックを作成した後。	ブローカーで階層化の対象になっているが、まだ階層型ストレージに転送されていないデータの合計バイト数。このメトリクスは、アップストリームのデータ転送の効率を示します。ラグが大きくなると、階層型ストレージに保持されないデータの量が増えます。カテゴリ: アーカイブラグ。これは KIP-405 メトリクスではありません。
TrafficBytes	クラスターが ACTIVE 状態になった後。	クライアント (プロデューサーとコンシューマー) とブローカー間のネットワークトラフィックを全体のバイト数で表示します。ブローカー間のトラフィックは報告されません。
VolumeQueueLength	クラスターが ACTIVE 状態になった後。	指定された期間内に完了するのを待機している読み取りおよび書き込みオペレーション要求の数。
VolumeReadBytes	クラスターが ACTIVE 状態になった後。	指定された期間に読み取られたバイト数。
VolumeReadOps	クラスターが ACTIVE 状態になった後。	指定された期間内の読み取りオペレーションの数。
VolumeTotalReadTime	クラスターが ACTIVE 状態になった後。	指定された期間内に完了したすべての読み取りオペレーションに費やされた合計秒数。
VolumeTotalWriteTime	クラスターが ACTIVE 状態になった後。	指定された期間内に完了したすべての書き込みオペレーションに費やされた合計秒数。

名前	表示可能なタイミング	説明
VolumeWriteBytes	クラスターが ACTIVE 状態になった後。	指定された期間に書き込まれたバイト数。
VolumeWriteOps	クラスターが ACTIVE 状態になった後。	指定された期間内の書き込みオペレーションの数。

PER_TOPIC_PER_BROKER レベルモニタリング

モニタリングレベルを PER_TOPIC_PER_BROKER に設定すると、PER_BROKER および DEFAULT レベルのすべてのメトリクスに加えて、次の表で説明するメトリクスを取得します。DEFAULT レベルメトリクスのみが無料です。この表のメトリクスには、クラスター名、ブローカー ID、トピックのディメンションがあります。

> **Important**
>
> Apache Kafka 2.4.1 以降のバージョンを使用する Amazon MSK クラスターの場合、次の表のメトリクスは、値が初めてゼロ以外になった後にのみ表示されます。たとえば、BytesInPerSec を表示するには、1 つ以上のプロデューサーが最初にクラスターにデータを送信する必要があります。

名前	表示可能なタイミング	説明
FetchMessageConversionsPerSec	トピックを作成した後。	1 秒あたりに変換されたフェッチ済みメッセージの数。
MessagesInPerSec	トピックを作成した後。	1 秒あたりに受信したメッセージ数。
ProduceMessageConversionsPerSec	トピックを作成した後。	生成されたメッセージの 1 秒あたりの変換回数。

名前	表示可能なタイミング	説明
RemoteFetchBytesPerSec (RemoteBytesInPerSec in v2.8.2.tiered)	トピックを作成して、そのトピックが生成/消費中。	指定されたトピックとブローカーに対するコンシューマーフェッチへのレスポンスで階層型ストレージから転送されたバイト数。このメトリクスには、指定されたブローカーでのダウンストリームのデータ転送トラフィックの一因となるトピックのすべてのパーティションが含まれます。カテゴリ: トラフィックとエラー率。これは KIP-405 メトリクスです。
RemoteCopyBytesPerSec (RemoteBytesOutPerSec in v2.8.2.tiered)	トピックを作成して、そのトピックが生成/消費中。	指定されたトピックとブローカーで階層型ストレージに転送されたバイト数。このメトリクスには、指定されたブローカーでのアップストリームのデータ転送トラフィックの一因となるトピックのすべてのパーティションが含まれます。カテゴリ: トラフィックとエラー率。これは KIP-405 メトリクスです。
RemoteFetchErrorsPerSec (RemoteReadErrorPerSec in v2.8.2.tiered)	トピックを作成して、そのトピックが生成/消費中。	指定されたブローカーが指定されたトピックでのコンシューマーフェッチへのレスポンスでデータを取得するために階層型ストレージに送信した読み取りリクエストに対するレスポンスでのエラー率。このメトリクスには、指定されたブローカーでのダウンストリームのデータ転送トラフィックの一因となるトピックのすべてのパーティションが含まれます。カテゴリ: トラフィックとエラー率。これは KIP-405 メトリクスです。

名前	表示可能なタイミング	説明
RemoteFetchRequestsPerSec (RemoteReadRequestsPerSec in v2.8.2.tiered)	トピックを作成して、そのトピックが生成/消費中。	指定されたブローカーが指定されたトピックでのコンシューマーフェッチへのレスポンスでデータを取得するために階層型ストレージに送信した読み取りリクエストの数。このメトリクスには、指定されたブローカーでのダウンストリームのデータ転送トラフィックの一因となるトピックのすべてのパーティションが含まれます。カテゴリ: トラフィックとエラー率。これは KIP-405 メトリクスです。
RemoteCopyErrorsPerSec (RemoteWriteErrorPerSec in v2.8.2.tiered)	トピックを作成して、そのトピックが生成/消費中。	指定されたブローカーがデータをアップストリームに転送するために階層型ストレージに送信した書き込みリクエストに対するレスポンスでのエラー率。このメトリクスには、指定されたブローカーでのアップストリームのデータ転送トラフィックの一因となるトピックのすべてのパーティションが含まれます。カテゴリ: トラフィックとエラー率。これは KIP-405 メトリクスです。
RemoteLogSizeBytes	トピックを作成した後。	リモート階層に保存されているバイト数。 このメトリクスは、Amazon MSK の Apache Kafka バージョン 3.7.x の階層型ストレージクラスターで使用できます。

PER_TOPIC_PER_PARTITION レベルモニタリング

モニタリングレベルを PER_TOPIC_PER_PARTITION に設定すると、PER_TOPIC_PER_BROKER、PER_BROKER、および DEFAULT レベルのすべてのメトリクスに加えて、次の表で説明するメトリクスが取得されます。DEFAULT レベルメトリクスのみが無料です。この表のメトリクスには、コンシューマーグループ、トピック、パーティションのディメンションがあります。

名前	表示可能なタイミング	説明
EstimatedTimeLag	コンシューマーグループがトピックから消費した後。	パーティションオフセットラグを排出するための推定時間 (秒単位)。
OffsetLag	コンシューマーグループがトピックから消費した後。	オフセット数のパーティションレベルのコンシューマーラグ。

MSK プロビジョンドクラスターの状態を理解する

次の表は、MSK プロビジョンドクラスターの可能な状態と、その意味を示しています。特に指定がない限り、MSK プロビジョンドクラスターの状態は、スタンダードブローカータイプと Express ブローカータイプの両方に適用されます。この表では、MSK プロビジョンドクラスターがこれらの状態のいずれかである場合に実行できるアクションと実行できないアクションについても説明します。クラスターの状態を確認するには、 AWS Management Consoleにアクセスします。describe-cluster-v2 コマンドまたは DescribeClusterV2 オペレーションを使用して、プロビジョニングされたクラスターを記述することもできます。クラスターの説明に、クラスターの状態が含まれます。

MSK プロビジョンドクラスターの状態	意味と実行可能なアクション
ACTIVE	データを生成および消費できます。クラスターで Amazon MSK API および AWS CLI オペレーションを実行することもできます。
CREATING	Amazon MSK はプロビジョニングされたクラスターをセットアップしています。クラスターが ACTIVE 状態になるまで待ってから、クラスターを使用してデータを生成または消費したり、Amazon MSK API または AWS CLI オペレーションを実行したりできます。

MSK プロビジョンドクラスターの状態	意味と実行可能なアクション
DELETING	プロビジョニングされたクラスターは削除中です。クラスターを使用してデータを生成したり消費したりすることはできません。また、Amazon MSK API または AWS CLI オペレーションを実行することはできません。
FAILED	プロビジョニングされたクラスターの作成または削除プロセスに失敗しました。クラスターを使用してデータを生成したり消費したりすることはできません。クラスターを削除することはできますが、Amazon MSK API を実行したり、クラスターに対してオペレーション AWS CLI を更新したりすることはできません。
HEALING	Amazon MSK は、異常なブローカーの交換などの内部オペレーションを実行しています。例えば、ブローカーが応答しない場合があります。プロビジョンドクラスターを使用してデータを生成および消費することはできます。ただし、クラスターが ACTIVE 状態に戻るまで、Amazon MSK API を実行したり、クラスターでオペレーション AWS CLI を更新したりすることはできません。

MSK プロビジョンドクラスターの状態	意味と実行可能なアクション
MAINTENANCE	(標準ブローカーのみ) Amazon MSK はクラスターで定期的なメンテナンスオペレーションを実行しています。このようなメンテナンスオペレーションには、セキュリティパッチの適用が含まれます。引き続き、クラスターを使用してデータを生成し、消費することができます。ただし、クラスターが ACTIVE AWS 状態に戻るまで、クラスターで Amazon MSK API または CLI 更新オペレーションを実行することはできません。Express ブローカーのメンテナンス中、クラスターの状態は ACTIVE のままです。「パッチ適用」を参照してください。
REBOOTING_BROKER	Amazon MSK はブローカーを再起動しています。プロビジョンドクラスターを使用してデータを生成および消費することはできます。ただし、クラスターが ACTIVE 状態に戻るまで、Amazon MSK API を実行したり、クラスターでオペレーション AWS CLI を更新したりすることはできません。
UPDATING	ユーザー主導の Amazon MSK API または AWS CLI オペレーションは、プロビジョニングされたクラスターを更新しています。プロビジョンドクラスターを使用してデータを生成および消費することはできます。ただし、クラスターが ACTIVE 状態に戻るまで、追加の Amazon MSK API または AWS CLI 更新オペレーションを実行することはできません。

CloudWatch で Express ブローカーをモニタリングするための Amazon MSK メトリクス

Amazon MSK は CloudWatch と統合されているため、MSK Express ブローカーの CloudWatch メトリクスを収集、表示、分析できます。MSK プロビジョンドクラスター用に設定したメトリクスは、自動的に収集され、1 分間隔で CloudWatch にプッシュされます。MSK プロビジョンドクラスターのモニタリングレベルは、DEFAULT、、PER_BROKERPER_TOPIC_PER_BROKER、または のいずれかに設定できますPER_TOPIC_PER_PARTITION。次のセクションの表は、各モニタリングレベルから利用できるメトリクスを示しています。

DEFAULT レベルのメトリクスは無料です。他のメトリクスの料金については、Amazon CloudWatch の料金ページで説明されています。

DEFAULT Express ブローカーのレベルモニタリング

次の表に示すメトリクスは、DEFAULTモニタリングレベルで無料で利用できます。

名前	表示可能なタイミング	ディメンション	説明
ActiveControllerCount	クラスターが ACTIVE 状態になった後。	クラスター名	クラスターごとに 1 つのコントローラーだけをアクティブにする必要があります。
BytesInPerSec	トピックを作成した後。	クラスター名、ブローカー ID、トピック	クライアントから受信した 1 秒あたりのバイト数。このメトリクスは、ブローカーごとおよびトピックごとに利用できます。
BytesOutPerSec	トピックを作成した後。	クラスター名、ブローカー ID、トピック	クライアントに送信された 1 秒あたりのバイト数。このメトリクスは、ブローカーごとおよびト

名前	表示可能なタイミング	ディメンション	説明
			ピックごとに利用できます。
ClientConnectionCount	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID、クライアント認証	アクティブな認証済みクライアント接続の数。
ConnectionCount	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	アクティブな認証済み接続、未認証接続、およびブローカー間接続の数。
CpuIdle	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	CPU アイドル時間の割合。
CpuSystem	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	カーネルスペースの CPU の割合。
CpuUser	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ユーザースペースの CPU の割合。

名前	表示可能なタイミング	ディメンション	説明
GlobalPartitionCount	クラスターが ACTIVE 状態になった後。	クラスター名	クラスター内のすべてのトピック全体のパーティション数 (レプリカを除く)。GlobalPartitionCount にはレプリカが含まれていないため、トピックのレプリケーション係数が より大きいGlobalPartitionCount 場合よりもPartitionCount 値の合計が高くなる可能性があります1。
GlobalTopicCount	クラスターが ACTIVE 状態になった後。	クラスター名	クラスター内のすべてのブローカーのトピックの合計数。
EstimatedMaxTimeLag	コンシューマーグループがトピックから消費した後。	コンシューマーグループ、トピック	MaxOffsetLag を排出するための推定時間 (秒単位)。
LeaderCount	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	レプリカを含まない、ブローカーごとのパーティションのリーダーの総数。
MaxOffsetLag	コンシューマーグループがトピックから消費した後。	コンシューマーグループ、トピック	トピック内のすべてのパーティションにわたる最大オフセットラグ。

名前	表示可能なタイミング	ディメンション	説明
MemoryBuffered	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーのバッファーメモリーのサイズ (バイト単位)。
MemoryCached	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーのキャッシュメモリーのサイズ (バイト単位)。
MemoryFree	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーが使用可能な空きメモリのサイズ (バイト単位) 。
MemoryUsed	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーに使用されているメモリのサイズ (バイト単位) 。
MessagesInPerSec	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーの 1 秒あたりの受信メッセージ数。
NetworkRxDropped	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ドロップされた受信パッケージの数。
NetworkRxErrors	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーのネットワーク受信エラーの数。
NetworkRxPackets	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーによって受信されたパケットの数。
NetworkTxDropped	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ドロップされた送信パッケージの数。

名前	表示可能なタイミング	ディメンション	説明
NetworkTxErrors	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーのネットワーク送信エラーの数。
NetworkTxPackets	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーによって送信されたパケットの数。
PartitionCount	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	レプリカを含む、ブローカーごとのトピックパーティションの総数。
ProduceTotalTimeMsMean	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ミリ秒単位の平均生成時間。
RequestBytesMean	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーのリクエストバイトの平均数。
RequestTime	リクエストスロットリングが適用された後。	クラスター名、ブローカー ID	ブローカーネットワークおよび I/O スレッドでリクエストを処理するのに費やされた平均時間 (ミリ秒)。
ストレージ使用量	クラスターが ACTIVE 状態になった後。	クラスター名	レプリカを除く、クラスター内のすべてのパーティションで使用されるストレージの合計。

名前	表示可能なタイミング	ディメンション	説明
SumOffsetLag	コンシューマーグループがトピックから消費した後。	コンシューマーグループ、トピック	トピックのすべてのパーティションの集計されたオフセットラグ。
UserPartitionExists	クラスターが ACTIVE 状態になった後。	クラスター名、ブローカー ID	ブローカーにユーザー所有のパーティションが存在することを示すブールメトリクス。値 1 は、ブローカーにパーティションが存在することを示します。

PER_BROKER Express ブローカーのレベルモニタリング

モニタリングレベルを PER_BROKER に設定すると、すべての DEFAULT レベルメトリクスに加えて、次の表で説明するメトリクスが表示されます。次の表のメトリクスに対して料金が発生しますが、DEFAULTレベルメトリクスは引き続き無料です。この表のメトリクスには、クラスター名、ブローカー ID のディメンションがあります。

PER_BROKER モニタリングレベルから利用可能な追加のメトリクス

名前	表示可能なタイミング	説明
ConnectionCloseRate	クラスターが ACTIVE 状態になった後。	リスナーごとの 1 秒あたりに閉じられた接続の数。この数はリスナーごとに集計され、クライアントリスナー用にフィルタリングされます。
ConnectionCreationRate	クラスターが ACTIVE 状態になった後。	リスナーごとに 1 秒あたりに確立された新しい接続の数。この数はリスナーごとに集計され、クライアントリスナー

名前	表示可能なタイミング	説明
		用にフィルタリングされます。
FetchConsumerLocalTimeMsMean	プロデューサー/コンシューマーがいる後。	コンシューマーのリクエストがリーダーで処理される平均時間（ミリ秒）。
FetchConsumerRequestQueueTimeMsMean	プロデューサー/コンシューマーがいる後。	コンシューマーのリクエストがリクエストキューで待機する平均時間（ミリ秒）。
FetchConsumerResponseQueueTimeMsMean	プロデューサー/コンシューマーがいる後。	コンシューマーのリクエストが応答キューで待機する平均時間（ミリ秒）。
FetchConsumerResponseSendTimeMsMean	プロデューサー/コンシューマーがいる後。	コンシューマーが応答を送信するための平均時間（ミリ秒）。
FetchConsumerTotalTimeMsMean	プロデューサー/コンシューマーがいる後。	コンシューマーがブローカーからデータを取得するのに費やす平均合計時間（ミリ秒）。
FetchFollowerLocalTimeMsMean	プロデューサー/コンシューマーがいる後。	リーダーでフォロワーのリクエストが処理される平均時間（ミリ秒）。
FetchFollowerRequestQueueTimeMsMean	プロデューサー/コンシューマーがいる後。	フォロワーのリクエストがリクエストキューで待機する平均時間（ミリ秒）。
FetchFollowerResponseQueueTimeMsMean	プロデューサー/コンシューマーがいる後。	フォロワーの要求が応答キューで待機する平均時間（ミリ秒）。

名前	表示可能なタイミング	説明
FetchFollowerResponseSendTimeMsMean	プロデューサー/コンシューマーがいる後。	フォロワーが応答を送信するまでの平均時間 (ミリ秒)。
FetchFollowerTotalTimeMsMean	プロデューサー/コンシューマーがいる後。	フォロワーがブローカーからデータを取得するのに費やす平均合計時間 (ミリ秒) 。
FetchThrottleByteRate	帯域幅スロットリングが適用された後。	1 秒あたりのスロットルバイト数。
FetchThrottleQueueSize	帯域幅スロットリングが適用された後。	スロットルキュー内のメッセージ数。
FetchThrottleTime	帯域幅スロットリングが適用された後。	フェッチスロットルの平均時間 (ミリ秒単位) 。
IAMNumberOfConnectionRequests	クラスターが ACTIVE 状態になった後。	1 秒あたりの IAM 認証リクエスト数。
IAMTooManyConnections	クラスターが ACTIVE 状態になった後。	100 を超えて試行された接続の数。 は、接続数が制限内である`0`ことを意味します。の場合`>0`、スロットル制限を超えているため、接続数を減らす必要があります。
NetworkProcessorAvgIdlePercent	クラスターが ACTIVE 状態になった後。	ネットワークプロセッサがアイドル状態の平均時間の割合。
ProduceLocalTimeMsMean	クラスターが ACTIVE 状態になった後。	リーダーでリクエストが処理される平均時間 (ミリ秒単位)。
ProduceRequestQueueTimeMsMean	クラスターが ACTIVE 状態になった後。	リクエストメッセージがキューに費やした平均時間 (ミリ秒) 。

名前	表示可能なタイミング	説明
ProduceResponseQueueTimeMsMean	クラスターが ACTIVE 状態になった後。	応答メッセージがキューに費やした平均時間 (ミリ秒) 。
ProduceResponseSendTimeMsMean	クラスターが ACTIVE 状態になった後。	応答メッセージの送信に費やされた平均時間 (ミリ秒) 。
ProduceThrottleByteRate	帯域幅スロットリングが適用された後。	1 秒あたりのスロットルバイト数。
ProduceThrottleQueueSize	帯域幅スロットリングが適用された後。	スロットルキュー内のメッセージ数。
ProduceThrottleTime	帯域幅スロットリングが適用された後。	平均生成スロットル時間 (ミリ秒単位) 。
ProduceTotalTimeMsMean	クラスターが ACTIVE 状態になった後。	ミリ秒単位の平均生成時間。
ReplicationBytesInPerSec	トピックを作成した後。	他のブローカーから受信した 1 秒あたりのバイト数。
ReplicationBytesOutPerSec	トピックを作成した後。	他のブローカーに送信された 1 秒あたりのバイト数。
RequestExemptFromThrottleTime	リクエストスロットリングが適用された後。	ブローカーネットワークおよび I/O スレッドで、スロットリングから除外されたリクエストを処理するのに費やされた平均時間 (ミリ秒)。
RequestHandlerAvgIdlePercent	クラスターが ACTIVE 状態になった後。	リクエストハンドラーのスレッドがアイドル状態の平均時間の割合。
RequestThrottleQueueSize	リクエストスロットリングが適用された後。	スロットルキュー内のメッセージ数。

名前	表示可能なタイミング	説明
RequestThrottleTime	リクエストスロットリングが適用された後。	リクエストスロットルの平均時間（ミリ秒単位）。
TcpConnections	クラスターが ACTIVE 状態になった後。	SYN フラグが設定された着信および発信 TCP セグメントの数を表示します。
TrafficBytes	クラスターが ACTIVE 状態になった後。	クライアント (プロデューサーとコンシューマー) とブローカー間のネットワークトラフィックを全体のバイト数で表示します。ブローカー間のトラフィックは報告されません。

PER_TOPIC_PER_PARTITION Express ブローカーのレベルモニタリング

モニタリングレベルを に設定するとPER_TOPIC_PER_PARTITION、、、および DEFAULTレベルのすべてのメトリクスに加えてPER_TOPIC_PER_BROKERPER_BROKER、次の表で説明されているメトリクスを取得します。DEFAULT レベルのメトリクスのみが無料です。この表のメトリクスには、コンシューマーグループ、トピック、パーティションのディメンションがあります。

PER_PARTITION モニタリングレベルから利用できる追加のメトリクス

名前	表示可能なタイミング	説明
EstimatedTimeLag	コンシューマーグループがトピックから消費した後。	パーティションオフセットラグを排出するための推定時間 (秒単位)。
OffsetLag	コンシューマーグループがトピックから消費した後。	オフセット数のパーティションレベルのコンシューマーラグ。

PER_TOPIC_PER_BROKER Express ブローカーのレベルモニタリング

モニタリングレベルを に設定するとPER_TOPIC_PER_BROKER、 PER_BROKERおよび DEFAULTレベルのすべてのメトリクスに加えて、次の表で説明されているメトリクスを取得します。DEFAULT レベルのメトリクスのみが無料です。この表のメトリクスには、クラスター名、ブローカー ID、トピックのディメンションがあります。

> **Important**
>
> 次の表のメトリクスは、値が初めてゼロ以外になった後にのみ表示されます。例えば、BytesInPerSec を表示するには、1 つ以上のプロデューサーが最初にクラスターにデータを送信する必要があります。

PER_TOPIC_PER_BROKER モニタリングレベルから利用できる追加のメトリクス

名前	表示可能なタイミング	説明
MessagesInPerSec	トピックを作成した後。	1 秒あたりに受信したメッセージ数。

Prometheus で MSK プロビジョンドクラスターをモニタリングする

MSK プロビジョンドクラスターは、時系列メトリクスデータのオープンソースモニタリングシステムである Prometheus でモニタリングできます。Prometheus のリモート書き込み機能を使用して、このデータを Amazon Managed Service for Prometheus に公開できます。また、Prometheus 形式のメトリクスと互換性のあるツールや、Datadog、Lenses、New Relic、Sumo ロジックなどの Amazon MSK Open Monitoring と統合するツールを使用することもできます。オープンモニタリングは無料ですが、アベイラビリティーゾーン間のデータ転送には料金がかかります。

Prometheus の詳細については、Prometheus のドキュメントを参照してください。

Prometheus の使用の詳細については、「Amazon Managed Service for Prometheus と Amazon Managed Grafana を使用した Amazon MSK の運用上のインサイトの強化」を参照してください。

> **Note**
>
> KRaft メタデータモードと MSK Express ブローカーでは、オープンモニタリングとパブリックアクセスの両方を有効にすることはできません。

新しい MSK プロビジョンドクラスターでオープンモニタリングを有効にする

この手順では、、 AWS Management Console、 AWS CLIまたは Amazon MSK API を使用して、新しい MSK クラスターでオープンモニタリングを有効にする方法について説明します。

の使用 AWS Management Console

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. [Monitoring (モニタリング)] セクションで、[Enable open monitoring with Prometheus (Prometheus でオープンモニタリングを有効にする)] の横にあるチェックボックスをオンにします。
3. ページのすべてのセクションに必要な情報を入力し、使用可能なすべてのオプションを確認します。
4. Create cluster (クラスターの作成) を選択します。

の使用 AWS CLI

- create-cluster コマンドを呼び出し、`open-monitoring` オプションを指定します。`JmxExporter`、`NodeExporter`、またはその両方を有効にします。`open-monitoring` を指定した場合、2 つのエクスポーターを同時に無効にすることはできません。

API の使用

- CreateCluster オペレーションを呼び出し、`OpenMonitoring` を指定します。`jmxExporter`、`nodeExporter`、またはその両方を有効にします。`OpenMonitoring` を指定した場合、2 つのエクスポーターを同時に無効にすることはできません。

既存の MSK プロビジョンドクラスターでオープンモニタリングを有効にする

オープンモニタリングを有効にするには、MSK プロビジョンドクラスターが `ACTIVE`状態であることを確認します。

の使用 AWS Management Console

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。

2. 更新するクラスターの名前を選択します。これにより、そのクラスターの詳細を含むページが表示されます。
3. [プロパティ] タブで、下にスクロールして [モニタリング] セクションを見つけます。
4. [編集] を選択します。
5. [Enable open monitoring with Prometheus (Prometheus でオープンモニタリングを有効にする)] の横にあるチェックボックスをオンにします。
6. [Save changes] (変更の保存) をクリックします。

の使用 AWS CLI

- update-monitoring コマンドを呼び出し、その `open-monitoring` オプションを指定します。`JmxExporter`、`NodeExporter`、またはその両方を有効にします。`open-monitoring` を指定した場合、2 つのエクスポーターを同時に無効にすることはできません。

API の使用

- UpdateMonitoring オペレーションを呼び出し、`OpenMonitoring` を指定します。`jmxExporter`、`nodeExporter`、またはその両方を有効にします。`OpenMonitoring` を指定した場合、2 つのエクスポーターを同時に無効にすることはできません。

Amazon EC2 インスタンスで Prometheus ホストを設定する

この手順では、prometheus.yml ファイルを使用して Prometheus ホストを設定する方法について説明します。

1. Prometheus サーバーを https://prometheus.io/download/#prometheus から Amazon EC2 インスタンスにダウンロードします。
2. ダウンロードしたファイルをディレクトリに展開し、そのディレクトリに移動します。
3. 次の内容で、`prometheus.yml` という名前のファイルを作成します。

```
# file: prometheus.yml
# my global config
global:
  scrape_interval:     60s

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
```

```
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped
 from this config.
  - job_name: 'prometheus'
    static_configs:
    # 9090 is the prometheus server port
    - targets: ['localhost:9090']
  - job_name: 'broker'
    file_sd_configs:
    - files:
      - 'targets.json'
```

4. ListNodes オペレーションを使用して、クラスターのブローカーのリストを取得します。
5. 以下の JSON を使用した `targets.json` という名前のファイルを作成します。*broker_dns_1*、*broker_dns_2*、残りのブローカーの DNS 名を、前の手順でブローカー用に取得した DNS 名に置き換えます。前のステップで取得したブローカーのすべてを含めます。Amazon MSK は、JMX Exporter にポート 11001 を使用し、Node Exporter にポート 11002 を使用します。

   ZooKeeper mode targets.json

```
[
  {
    "labels": {
      "job": "jmx"
    },
    "targets": [
      "broker_dns_1:11001",
      "broker_dns_2:11001",
      .
      .
      .
      "broker_dns_N:11001"
    ]
  },
  {
    "labels": {
      "job": "node"
    },
    "targets": [
      "broker_dns_1:11002",
      "broker_dns_2:11002",
```

```
      .
      .
      .
      "broker_dns_N:11002"
    ]
  }
]
```

KRaft mode targets.json

```
[
  {
    "labels": {
      "job": "jmx"
    },
    "targets": [
      "broker_dns_1:11001",
      "broker_dns_2:11001",
      .
      .
      .
      "broker_dns_N:11001",
      "controller_dns_1:11001",
      "controller_dns_2:11001",
      "controller_dns_3:11001"
    ]
  },
  {
    "labels": {
      "job": "node"
    },
    "targets": [
      "broker_dns_1:11002",
      "broker_dns_2:11002",
      .
      .
      .
      "broker_dns_N:11002"
    ]
  }
]
```

> **Note**
>
> KRaft コントローラーから JMX メトリクスをスクレイプするには、コントローラーの DNS 名を JSON ファイルにターゲットとして追加します。例えば `controller_dns_1:11001` です。ここで、`controller_dns_1` を実際のコントローラー DNS 名に置き換えます。

6. Amazon EC2 インスタンスで Prometheus サーバーをスタートするには、Prometheus ファイルを抽出して `prometheus.yml` と `targets.json` を保存したディレクトリで次のコマンドを実行します。

```
./prometheus
```

7. 前のステップで Prometheus を実行した Amazon EC2 インスタンスの IPv4 パブリック IP アドレスを見つけます。このパブリック IP アドレスは、次のステップで必要になります。
8. Prometheus ウェブ UI にアクセスするには、Amazon EC2 インスタンスにアクセスできるブラウザを開き、*Prometheus-Instance-Public-IP*`:9090` に移動します。ここで、*Prometheus-Instance-Public-IP*は、前の手順で取得したパブリック IP アドレスです。

Prometheus メトリクスを使用する

Apache Kafka から JMX に出力されるすべてのメトリクスは、Prometheus のオープンモニタリングを使用してアクセスできます。Apache Kafka のメトリクスについては、Apache Kafka のドキュメントの「モニタリング」を参照してください。Apache Kafkaメトリクスに加えて、コンシューマラグメトリクスも JMX MBean 名 `kafka.consumer.group:type=ConsumerLagMetrics` のポート 11001 で利用できます。Prometheus Node Exporter を使用して、ポート 11002 でブローカーの CPU およびディスクメトリクスを取得することもできます。

Prometheus メトリクスを Amazon Managed Service for Prometheus に保存する

Amazon Managed Service for Prometheus は Prometheus 互換のモニタリングおよびアラートサービスであり、Amazon MSK クラスターのモニタリングに使用できます。これは、メトリクスの取り込み、ストレージ、クエリ、アラートを自動的にスケーリングするフルマネージド型サービスです。また、 AWS セキュリティサービスと統合して、データへの高速かつ安全なアクセスを提供します。オープンソースの PromQL クエリ言語を使用して、メトリクスをクエリし、それらに関するアラートを作成できます。

詳細については、Amazon Managed Service for Prometheus の使用を開始するを参照してください。

コンシューマーラグをモニタリングする

コンシューマーラグをモニタリングすることで、トピックで利用可能な最新のデータに追いついていない、遅いまたはスタックしたコンシューマーを特定できます。必要に応じて、それらのコンシューマーのスケーリングや再起動などの修正アクションを実行できます。コンシューマーラグをモニタリングするには、Amazon CloudWatch を使用するか、Prometheus でオープンモニタリングを使用できます。

コンシューマーラグメトリクスは、トピックに書き込まれた最新のデータとアプリケーションによって読み取られたデータの違いを定量化します。Amazon MSK は、Amazon CloudWatch または Prometheus を使用したオープンモニタリングを通じて取得できる次のコンシューマーラグメトリクスを提供します: `EstimatedMaxTimeLag`、`EstimatedTimeLag`、`MaxOffsetLag`、`OffsetLag`、および `SumOffsetLag`。これらのメトリクスの詳細については、「the section called “標準ブローカーの CloudWatch メトリクス”」を参照してください。

Amazon MSK は、Apache Kafka 2.2.1 以降のバージョンを搭載したクラスターでのコンシューマーラグメトリクスをサポートしています。Kafka および CloudWatch メトリクスを使用する場合は、次の点を考慮してください。

- コンシューマーラグメトリクスは、コンシューマーグループが STABLE または EMPTY 状態にある場合にのみ出力されます。コンシューマーグループは、パーティションがコンシューマー間で均等に分散されるようにするリバランシングが正常に完了した後に安定します。
- コンシューマーラグメトリクスは、次のシナリオでは存在しません。
  - コンシューマーグループが不安定である場合。
  - コンシューマーグループの名前にはコロン (:) が含まれます。
  - コンシューマーグループのコンシューマーオフセットを設定していません。
- コンシューマーグループ名は、CloudWatch のコンシューマーラグメトリクスのディメンションとして使用されます。Kafka はコンシューマーグループ名で UTF-8 文字をサポートしていますが、CloudWatch はディメンション値の ASCII 文字のみをサポートしています。コンシューマーグループ名に ASCII 以外の文字を使用すると、CloudWatch はコンシューマーラグメトリクスを削除します。コンシューマーラグメトリクスが CloudWatch で適切にキャプチャされるようにするには、コンシューマーグループ名に ASCII 文字のみを使用する必要があります。

Amazon MSK ストレージ容量アラートを使用する

Amazon MSK でプロビジョニングされたクラスターでは、クラスターのプライマリストレージ容量を選択します。プロビジョニングしたクラスター内のブローカーのストレージ容量を使い果たすと、そのブローカーのデータ生成と消費能力に影響が及び、コストのかかるダウンタイムにつながる可能性があります。Amazon MSK には、クラスターのストレージ容量をモニタリングするのに役立つ CloudWatch メトリクスが用意されています。ただし、ストレージ容量の問題をより簡単に検出して解決できるように、Amazon MSK は動的クラスターストレージ容量アラートを自動的に送信します。ストレージ容量アラートには、クラスターのストレージ容量を管理するための短期および長期の手順に関する推奨事項が含まれています。Amazon MSK コンソールから、アラート内のクイックリンクを使用して推奨アクションをすぐに実行できます。

MSK ストレージ容量アラートには、事前対応型と修正型の 2 種類があります。

- 事前対応型 (「対応が必要」) ストレージ容量アラートは、クラスターで発生する可能性のあるストレージ問題について警告します。MSK クラスター内のブローカーがディスクストレージ容量の 60% または 80% 以上を使用した場合、影響を受けるブローカーに対して事前対応型のアラートが届きます。
- 修正型 (「重大な対応が必要」) のストレージ容量アラートでは、MSK クラスター内のいずれかのブローカーのディスクストレージ容量が不足した場合に、クラスターの重大な問題を修正するための修正アクションを講じる必要があります。

Amazon MSK は、これらのアラートを Amazon MSK コンソール、AWS ヘルスダッシュボード、Amazon EventBridge、および AWS アカウントの E メール連絡先に自動的に送信します。これらのアラートを Slack や New Relic、Datadog などのツールに配信するように Amazon EventBridge を設定することもできます。

ストレージ容量アラートは MSK でプロビジョニングされたすべてのクラスターでデフォルトで有効になっており、オフにすることはできません。この機能は、MSK がサポートされているすべてのリージョンで利用できます。

ストレージ容量アラートをモニタリングする

ストレージ容量アラートは次のいくつかの方法で確認できます。

- Amazon MSK コンソールに進みます。ストレージ容量アラートは、クラスターアラートペインに 90 日間表示されます。アラートには、ディスクストレージ容量の問題に対処するための推奨事項とワンクリックリンクアクションが含まれています。

- ListClusters、ListClustersV2、DescribeCluster、または DescribeClusterV2 API を使用して、CustomerActionStatus とクラスターのすべてのアラートを表示します。
- AWS Health Dashboard にアクセスして、MSK やその他の AWS サービスからのアラートを確認してください。
- AWS Health API と Amazon EventBridge をセットアップして、アラート通知を Datadog、NewRelic、Slack などのサードパーティープラットフォームにルーティングします。

Amazon MSK クラスターのセキュリティ設定を更新する

UpdateSecurity Amazon MSK オペレーションを使用して、MSK クラスターの認証とクライアントブローカー暗号化の設定を更新します。また、相互 TLS 認証の証明書への署名に使用するプライベートセキュリティ認証を更新することもできます。クラスター内 (ブローカー間) 暗号化の設定は変更できません。

セキュリティ設定を更新するには、クラスターは ACTIVE 状態である必要があります。

IAM、SASL、または TLS を使用して認証をオンにする場合は、クライアントとブローカーの間の暗号化も有効にする必要があります。次の表に、考えられる組み合わせを示します。

認証	クライアントとブローカー間の暗号化オプション	ブローカー間の暗号化
未認証	TLS、PLAINTEXT、TLS_PLAINTEXT	オンでもオフでも可。
mTLS	TLS、TLS_PLAINTEXT	オンのみ可。
SASL/SCRAM	TLS	オンのみ可。
SASL/IAM	TLS	オンのみ可。

クライアント–ブローカー間の暗号化が TLS_PLAINTEXT に設定されており、クライアント認証は mTLS に設定されている場合、Amazon MSK はクライアントが接続可能な 2 種類のリスナーを作成します。一方はクライアントが TLS 暗号化で mTLS 認証を用いて接続できるリスナーで、もう一方は認証または暗号化 (プレーンテキスト) なしで接続可能なリスナーです。

セキュリティの設定についてのより詳細な情報は、「the section called “セキュリティ”」を参照してください。

を使用して Amazon MSK クラスターのセキュリティ設定を更新する AWS Management Console

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. 更新する MSK クラスターを選択します。
3. [Security settings] (セキュリティ設定) セクションで [Edit] (編集) を選択します。
4. クラスターに必要な認証および暗号化の設定を選択し、 [Save changes] (変更の保存) を行います。

を使用した Amazon MSK クラスターのセキュリティ設定の更新 AWS CLI

1. クラスターに適用したい暗号化設定を含む JSON ファイルを作成します。以下に例を示します。

   > **Note**
   >
   > 更新できるのは、クライアント–ブローカー間の暗号化設定のみです。クラスター内部 (ブローカー–ブローカー間) の暗号化設定を更新することはできません。

   ```
   {"EncryptionInTransit":{"ClientBroker": "TLS"}}
   ```

2. クラスターに適用したい認証設定を含む JSON ファイルを作成します。以下に例を示します。

   ```
   {"Sasl":{"Scram":{"Enabled":true}}}
   ```

3. 次の AWS CLI コマンドを実行します。

   ```
   aws kafka update-security --cluster-arn ClusterArn --current-version Current-Cluster-Version --client-authentication file://Path-to-Authentication-Settings-JSON-File --encryption-info file://Path-to-Encryption-Settings-JSON-File
   ```

   この `update-security` オペレーションの出力は、次の JSON のようになります。

   ```
   {
   ```

```
    "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
abcdefab-1234-abcd-5678-cdef0123ab01-2",
    "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef"
}
```

4. update-security オペレーションの結果を取得するには、*[ClusterOperationArn]* を update-security コマンドの出力で取得した ARN に置き換え、次のコマンドを実行します。

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

この describe-cluster-operation コマンドの出力は、次の JSON の例のようになります。

```
{
    "ClusterOperationInfo": {
        "ClientRequestId": "c0b7af47-8591-45b5-9c0c-909a1a2c99ea",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2021-09-17T02:35:47.753000+00:00",
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "PENDING",
        "OperationType": "UPDATE_SECURITY",
        "SourceClusterInfo": {},
        "TargetClusterInfo": {}
    }
}
```

OperationState に値 PENDING または UPDATE_IN_PROGRESS がある場合、しばらく待ってから再度 describe-cluster-operation コマンドを実行します。

> **Note**
>
> クラスターのセキュリティ設定を更新するための AWS CLI および API オペレーションはべき等です。つまり、セキュリティ更新オペレーションを呼び出して、対象のクラスターに現

在設定されているものと同じ認証または暗号化設定を指定しても、その設定は変更されません。

API を使用したクラスターのセキュリティ設定の更新

API を使用して Amazon MSK クラスターのセキュリティ設定を更新するには、「UpdateSecurity」を参照してください。

> **Note**
>
> MSK クラスターのセキュリティ設定を更新するための AWS CLI および API オペレーションはべき等です。つまり、セキュリティ更新オペレーションを呼び出して、対象のクラスターに現在設定されているものと同じ認証または暗号化設定を指定しても、その設定は変更されません。

Amazon MSK クラスター内のブローカーの数を拡張する

この Amazon MSK オペレーションは、お使いの MSK クラスター内のブローカー数を増やしたい場合に使用します。クラスターを拡張するには、クラスターが `ACTIVE` 状態であることを確認します。

> **Important**
>
> MSK クラスターを拡張する際には、必ずこの Amazon MSK オペレーションを使用してください。このオペレーションを使用せずにクラスターにブローカーを追加しようとしないでください。

ブローカーをクラスターに追加した後にパーティションを再調整する方法については、「the section called “パーティションの再割り当て”」を参照してください 。

を使用して Amazon MSK クラスターを拡張する AWS Management Console

このプロセスでは、 AWS Management Consoleを使用して、Amazon MSK クラスター内のブローカーの数を増やす方法について説明します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. ブローカー数を増やしたい MSK クラスターを選択します。
3. Actions ドロップダウンから、ブローカー数の編集を選択します。
4. クラスターに必要なブローカーの数を各アベイラビリティーゾーンごと入力したら、[Save changes] (変更を保存) を選択します。

を使用して Amazon MSK クラスターを拡張する AWS CLI

このプロセスでは、 AWS CLIを使用して、Amazon MSK クラスター内のブローカーの数を増やす方法について説明します。

1. *ClusterArn*をクラスターの作成時に取得したAmazon リソースネーム (ARN) に置き換えて、次のコマンドを実行します。クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。詳細については、「the section called “クラスターを一覧表示する”」を参照してください。

   Current-Cluster-Version を、クラスターの現在のバージョンに置き換えます。

   > **Important**
   >
   > クラスターのバージョンは単純な整数ではありません。クラスターの最新バージョンを検索するには、DescribeCluster オペレーションまたは describe-cluster AWS CLI コマンドを使用します。サンプルのバージョンは KTVPDKIKX0DER です。

   Target-Number-of-Brokers パラメータは、このオペレーションが正常に完了したときにクラスターが持つブローカーノードの総数を表します。*Target-Number-of-Brokers* に指定する値は、クラスター内の現在のブローカー数より大きい整数である必要があります。また、アベイラビリティーゾーンの数の倍数である必要があります。

   ```
   aws kafka update-broker-count --cluster-arn ClusterArn --current-version Current-Cluster-Version --target-number-of-broker-nodes Target-Number-of-Brokers
   ```

   この update-broker-count オペレーションの出力は、次の JSON のようになります。

   ```
   {
   ```

```
    "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
abcdefab-1234-abcd-5678-cdef0123ab01-2",
    "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef"
}
```

2. update-broker-count オペレーションの結果を取得するには、*ClusterOperationArn*を update-broker-count コマンドの出力で取得した ARN に置き換えて、次のコマンドを実行します。

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

この describe-cluster-operation コマンドの出力は、次の JSON の例のようになります。

```
{
    "ClusterOperationInfo": {
        "ClientRequestId": "c0b7af47-8591-45b5-9c0c-909a1a2c99ea",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2019-09-25T23:48:04.794Z",
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "UPDATE_COMPLETE",
        "OperationType": "INCREASE_BROKER_COUNT",
        "SourceClusterInfo": {
            "NumberOfBrokerNodes": 9
        },
        "TargetClusterInfo": {
            "NumberOfBrokerNodes": 12
        }
    }
}
```

この出力では、OperationType は INCREASE_BROKER_COUNT です。OperationState の値が UPDATE_IN_PROGRESS の場合は、しばらく待ってから再度 describe-cluster-operation コマンドを実行します。

API を使用して Amazon MSK クラスターを拡張する

API を使用してクラスター内のブローカー数を増やすには、「UpdateBrokerCount」を参照してください。

Amazon MSK クラスターからブローカーを削除する

Amazon Managed Streaming for Apache Kafka (MSK) のプロビジョニングされたクラスターからブローカーを削除する場合は、この Amazon MSK オペレーションを使用します。一連のブローカーを削除することで、可用性への影響、データ耐久性のリスク、データストリーミングアプリケーションの中断なしに、クラスターのストレージとコンピューティング容量を減らすことができます。

クラスターにブローカーを追加してトラフィックの増加に対処し、トラフィックが沈静化したときにブローカーを削除できます。ブローカーの追加および削除機能を使用すると、クラスター容量を最大限に活用し、MSK インフラストラクチャのコストを最適化できます。ブローカーを削除することで、ワークロードのニーズに合わせて既存のクラスター容量をブローカーレベルで制御し、別のクラスターへの移行を回避できます。

AWS コンソール、コマンドラインインターフェイス (CLI)、 SDK、または を使用して、プロビジョニングされたクラスターのブローカー数 AWS CloudFormation を減らします。MSK では、パーティションを持たないブローカー (canary トピックを除く) を選択し、アプリケーションがそれらのブローカーにデータを生成できないようにしながら、それらのブローカーをクラスターから安全に削除します。

クラスターのストレージとコンピューティングを減らす場合、アベイラビリティーゾーンごとに 1 つのブローカーを削除する必要があります。例えば、2 つのアベイラビリティーゾーンのクラスターから 2 つのブローカーを削除するか、単一のブローカー削除オペレーションで 3 つのアベイラビリティーゾーンのクラスターから 3 つのブローカーを削除できます。

クラスターからブローカーを削除した後にパーティションを再調整する方法については、「the section called “パーティションの再割り当て”」を参照してください。

インスタンスサイズに関係なく、M5 および M7g ベースのプロビジョニングされた MSK クラスターからブローカーを削除できます。

ブローカーの削除は、KRaft モードのクラスターを含む Kafka バージョン 2.8.1 以降でサポートされています。

トピック

- すべてのパーティションを削除してブローカーを削除する準備をする

- AWS マネジメントコンソールを使用してブローカーを削除する
- CLI AWS を使用してブローカーを削除する
- AWS API を使用してブローカーを削除する

すべてのパーティションを削除してブローカーを削除する準備をする

ブローカーを削除するプロセスを開始する前に、まず削除する予定のブローカーからトピック __amazon_msk_canary および __amazon_msk_canary_state のパーティションを除くすべてのパーティションを移動します。これらは Amazon MSK によって作成される、クラスターのヘルスと診断メトリクス用の内部トピックです。

Kafka Admin API または Cruise Control を使用して、クラスターに保持する予定の他のブローカーにパーティションを移動できます。「パーティションの再割り当て」を参照してください。

パーティションを削除するプロセスの例

このセクションでは、削除するブローカーからパーティションを削除する方法の例を示します。各 AZ に 6 つのブローカーと 2 つのブローカーを持つクラスターがあり、次の 4 つのトピックがあるとします。

- __amazon_msk_canary
- __consumer_offsets
- __amazon_msk_connect_offsets_my-mskc-connector_12345678-09e7-c657f7e4ff32-2
- msk-brk-rmv

1. 「クライアントマシンを作成する」の説明に従って、クライアントマシンを作成します。
2. クライアントマシンを設定したら、次のコマンドを実行して、クラスターで使用可能なすべてのトピックを一覧表示します。

   ```
   ./bin/kafka-topics.sh --bootstrap-server “CLUSTER_BOOTSTRAP_STRING” --list
   ```

   この例では、4 つのトピック名 __amazon_msk_canary、__consumer_offsets、__amazon_msk_connect_offsets_my-mskc-connector_12345678-09e7-c657f7e4ff32-2、および msk-brk-rmv が表示されます。

3. クライアントマシンで topics.json という JSON ファイルを作成し、次のコード例のようにすべてのユーザーのトピック名を追加します。これは、必要に応じて自動的に移動されるサービス管理のトピックであるため、トピック名 __amazon_msk_canary を含める必要はありません。

```
{
"topics": [
{"topic": "msk-brk-rmv"},
{"topic": "__consumer_offsets"},
{"topic": "__amazon_msk_connect_offsets_my-mskc-connector_12345678-09e7-
c657f7e4ff32-2"}
],
"version":1
}
```

4. 次のコマンドを実行して、クラスター上の 6 つのブローカーのうち 3 つのブローカーのみにパーティションを移動する提案を生成します。

```
./bin/kafka-reassign-partitions.sh --bootstrap-server “CLUSTER_BOOTSTRAP_STRING” --
topics-to-move-json-file topics.json --broker-list 1,2,3 --generate
```

5. reassignment-file.json というファイルを作成し、上記のコマンドから取得した proposed partition reassignment configuration をコピーします。
6. 次のコマンドを実行して、reassignment-file.json で指定したパーティションを移動します。

```
./bin/kafka-reassign-partitions.sh --bootstrap-server “CLUSTER_BOOTSTRAP_STRING” --
reassignment-json-file reassignment-file.json --execute
```

出力は次の例のようになります:

```
Successfully started partition reassignments for morpheus-test-topic-1-0,test-
topic-1-0
```

7. 次のコマンドを実行して、すべてのパーティションが移動したことを確認します。

```
./bin/kafka-reassign-partitions.sh --bootstrap-server “CLUSTER_BOOTSTRAP_STRING” --
reassignment-json-file reassignment-file.json --verify
```

出力は次の例のようになります。リクエストされたトピック内のすべてのパーティションが正常に再割り当てされるまで、ステータスをモニタリングします。

```
Status of partition reassignment:
Reassignment of partition msk-brk-rmv-0 is completed.
Reassignment of partition msk-brk-rmv-1 is completed.
Reassignment of partition __consumer_offsets-0 is completed.
Reassignment of partition __consumer_offsets-1 is completed.
```

8. 各パーティションでパーティションの再割り当てが完了したことがステータスで示されたら、UserPartitionExists メトリクスを 5 分間モニタリングして、パーティションを移動したブローカーに 0 が表示されることを確認します。これを確認したら、クラスターからブローカーを削除できます。

AWS マネジメントコンソールを使用してブローカーを削除する

AWS マネジメントコンソールでブローカーを削除するには

1. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
2. 削除するブローカーを含む MSK クラスターを選択します。
3. クラスターの詳細ページで、[アクション] ボタンを選択し、[ブローカーの数の編集] オプションを選択します。
4. クラスターでアベイラビリティーゾーンごとに持つブローカーの数を入力します。コンソールでは、削除するアベイラビリティーゾーン全体のブローカーの数がまとめられます。内容が正しいことを確認してください。
5. [Save changes] (変更の保存) をクリックします。

ブローカーの偶発的な削除を防ぐため、コンソールではブローカーの削除について確認するよう求められます。

CLI AWS を使用してブローカーを削除する

ClusterArn をクラスターの作成時に取得した Amazon リソースネーム (ARN) に置き換え、次のコマンドを実行します。クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。詳細については、「Amazon MSK クラスターを一覧表示する」を参照してください。Current-Cluster-Version を現在のクラスターのバージョンに置き換えます。

> **Important**
>
> クラスターのバージョンは単純な整数ではありません。クラスターの最新バージョンを検索するには、DescribeCluster オペレーションまたは describe-cluster AWS CLI コマンドを使用します。サンプルのバージョンは KTVPDKIKX0DER です。

Target-Number-of-Brokers パラメータは、このオペレーションが正常に完了したときにクラスターが持つブローカーノードの総数を表します。*Target-Number-of-Brokers* に指定する値は、クラスター内の現在のブローカーの数より小さい整数である必要があります。また、アベイラビリティーゾーンの数の倍数である必要があります。

```
aws kafka update-broker-count --cluster-arn ClusterArn --current-version Current-
Cluster-Version --target-number-of-broker-nodes Target-Number-of-Brokers
```

この update-broker-count オペレーションの出力は、次の JSON のようになります。

```
{
"ClusterOperationInfo": {
"ClientRequestId": "c0b7af47-8591-45b5-9c0c-909a1a2c99ea",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2019-09-25T23:48:04.794Z",
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "UPDATE_COMPLETE",
        "OperationType": "DECREASE_BROKER_COUNT",
        "SourceClusterInfo": {
"NumberOfBrokerNodes": 12
        },
        "TargetClusterInfo": {
"NumberOfBrokerNodes": 9
        }
    }
}
```

この出力では、OperationType は DECREASE_BROKER_COUNT です。OperationState の値が UPDATE_IN_PROGRESS の場合は、しばらく待ってから再度 describe-cluster-operation コマンドを実行します。

AWS API を使用してブローカーを削除する

API を使用してクラスター内のブローカーを削除するには、「Amazon Managed Streaming for Apache Kafka API リファレンス」の「UpdateBrokerCount」を参照してください。

Amazon MSK クラスターのブローカーサイズを更新する

Apache Kafka パーティションを再割り当てしなくても、ブローカーのサイズを変更することで、MSK クラスターをオンデマンドマンドでスケールできます。ブローカーのサイズを変更すると、クラスター I/O を中断することなく、ワークロードの変化に基づいて MSK クラスターのコンピューティング性能を柔軟に調整できます。Amazon MSK では、特定のクラスター内のすべてのブローカーに対して同じブローカーサイズを使用します。

標準ブローカーの場合、クラスターブローカーのサイズを M5 または T3 から M7g, T3 から M5、または M7g から M5 に更新できます。

> **Note**
>
> より大きなブローカーサイズからより小さなブローカーサイズに移行することはできません。例えば、M7g.large から T3.small などです。

Express ブローカーでは、M7g ブローカーサイズのみを使用できます。

このトピックでは、MSK クラスターのブローカーサイズを更新する方法について説明します。

ブローカーサイズを小さくすると、パフォーマンスが低下し、達成できるブローカーあたりの最大スループットが低下する可能性があることに注意してください。ブローカーサイズを大きくすると、パフォーマンスは向上しますが、コストが高くなる可能性があります。

ブローカーサイズの更新は、クラスターの稼働中にローリング方式で行われます。そのため、Amazon MSK がブローカーサイズの更新を実行する際には、一度に 1 つずつブローカーを停止します。ブローカーサイズを更新する際にクラスターの可用性を高める方法については、「the section called “高可用性クラスターの構築”」を参照してください。生産性に影響が出る可能性をさらに減らすには、トラフィックが少ない期間にブローカーサイズの更新を実行することもできます。

ブローカーサイズの更新中は、引き続きデータの生成と消費を行うことができます。ただし、更新が完了するまでは、ブローカーの再起動やAmazon MSK オペレーションに記載されている更新操作を呼び出すことはできません。

クラスターをより小さいブローカーサイズに更新する場合、まずテストクラスターで更新を試して、シナリオへの影響を確認することをお勧めします。

> **Important**
>
> 各ブローカーごとのパーティション数が the section called " クラスターの適切なサイズ設定: 標準ブローカーあたりのパーティション数" で指定された上限を超える場合、クラスターをより小さなブローカーサイズに更新することはできません。

トピック

- を使用して Amazon MSK クラスターブローカーのサイズを更新する AWS Management Console
- を使用して Amazon MSK クラスターブローカーのサイズを更新する AWS CLI
- API を使用したブローカーサイズの更新

を使用して Amazon MSK クラスターブローカーのサイズを更新する AWS Management Console

このプロセスでは、 を使用して Amazon MSK クラスターブローカーのサイズを更新する方法を示します。 AWS Management Console

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. ブローカーサイズを更新する MSK クラスターを選択します。
3. クラスターの詳細ページで、[ブローカーの概要] セクションにある、[ブローカーサイズの編集] を選択します。
4. リストから必要なブローカーサイズを選択します。
5. 変更の保存。

を使用して Amazon MSK クラスターブローカーのサイズを更新する AWS CLI

*ClusterArn*をクラスターの作成時に取得したAmazon リソースネーム (ARN) に置き換えて、次のコマンドを実行します。クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。詳細については、「the section called "クラスターを一覧表示する"」を参照してください。

1. *[Current-Cluster-Version]* をクラスターの現在のバージョンに、*[TargetType]* を変更先の新しいブローカーサイズに変更します。ブローカーサイズについての詳細は、「the section called “ブローカータイプ”」を参照してください。

```
aws kafka update-broker-type --cluster-arn ClusterArn --current-version Current-Cluster-Version --target-instance-type TargetType
```

次に、このコマンドの使用例を示します。

```
aws kafka update-broker-type --cluster-arn "arn:aws:kafka:us-east-1:0123456789012:cluster/exampleName/abcd1234-0123-abcd-5678-1234abcd-1" --current-version "K1X5R6FKA87" --target-instance-type kafka.m5.large
```

このコマンドの出力は、次の JSON の例のようになります。

```
{
    "ClusterArn": "arn:aws:kafka:us-east-1:0123456789012:cluster/exampleName/abcd1234-0123-abcd-5678-1234abcd-1",
    "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-abcd-4f7f-1234-9876543210ef"
}
```

2. update-broker-type オペレーションの結果を取得するには、*ClusterOperationArn*を update-broker-type コマンドの出力で取得した ARN に置き換えて、次のコマンドを実行します。

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

この describe-cluster-operation コマンドの出力は、次の JSON の例のようになります。

```
{
  "ClusterOperationInfo": {
    "ClientRequestId": "982168a3-939f-11e9-8a62-538df00285db",
    "ClusterArn": "arn:aws:kafka:us-east-1:0123456789012:cluster/exampleName/abcd1234-0123-abcd-5678-1234abcd-1",
    "CreationTime": "2021-01-09T02:24:22.198000+00:00",
```

```
    "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-operation/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
    "OperationState": "UPDATE_COMPLETE",
    "OperationType": "UPDATE_BROKER_TYPE",
    "SourceClusterInfo": {
      "InstanceType": "t3.small"
    },
    "TargetClusterInfo": {
      "InstanceType": "m5.large"
    }
  }
}
```

`OperationState` の値が `UPDATE_IN_PROGRESS` の場合は、しばらく待ってから再度 `describe-cluster-operation` コマンドを実行します。

API を使用したブローカーサイズの更新

API を使用してブローカーサイズを更新するには、「UpdateBrokerType」を参照してください。

`UpdateBrokerType` を使用して、クラスターのブローカーサイズを M5 または T3 から M7g に、または M7g から M5 に更新できます。

LinkedIn の Cruise Control for Apache Kafka を Amazon MSK で使用する

LinkedIn の Cruise Control を使用して、Amazon MSK クラスターのバランスを取り直し、異常を検出して修正し、クラスターの状態とヘルスをモニタリングできます。

Cruise Control をダウンロードして構築するには

1. Amazon MSK クラスターと同じ Amazon VPC に Amazon EC2 インスタンスを作成します。
2. 前のステップで作成した Amazon EC2 インスタンスに Prometheus をインストールします。プライベート IP とポートに注意してください。デフォルトのポート番号は 9090 です。クラスターのメトリクスを集約するように Prometheus を構成する方法については、「the section called “Prometheus でモニタリングする”」を参照してください。
3. Amazon EC2 インスタンスに Cruise Control をダウンロードします。(または、必要に応じて、Cruise Control に別の Amazon EC2 インスタンスを使用することもできます。) Apache Kafka バージョン 2.4.* を搭載したクラスターの場合は、最新の 2.4.* Cruise Control リリースを

使用してください。クラスターに 2.4.* より古いバージョンの Apache Kafka がある場合は、最新の 2.0.* Cruise Control リリースを使用してください。

4. Cruise Control ファイルを解凍してから、解凍したフォルダに移動します。
5. 次のコマンドを実行して git をインストールします。

```
sudo yum -y install git
```

6. 次のコマンドを実行して、ローカルリポジトリを初期化します。*Your-Cruise-Control-Folder*を、現在のフォルダ (Cruise Control のダウンロードを解凍したときに取得したフォルダ) の名前に置き換えます。

```
git init && git add . && git commit -m "Init local repo." && git tag -a Your-Cruise-Control-Folder -m "Init local version."
```

7. 次のコマンドを実行して、ソースコードを構築します。

```
./gradlew jar copyDependantLibs
```

Cruise Control を設定して実行するには

1. config/cruisecontrol.properties ファイルを次のように更新します。サンプルのブートストラップサーバーとブートストラップブローカーの文字列をクラスターの値に置き換えます。クラスターのこれらの文字列を取得するには、コンソールでクラスターの詳細を確認できます。または、GetBootstrapBrokersおよびDescribeCluster API オペレーションまたは同等の CLI オペレーションを使用することもできます。

```
# If using TLS encryption, use 9094; use 9092 if using plaintext
bootstrap.servers=b-1.test-cluster.2skv42.c1.kafka.us-
east-1.amazonaws.com:9094,b-2.test-cluster.2skv42.c1.kafka.us-
east-1.amazonaws.com:9094,b-3.test-cluster.2skv42.c1.kafka.us-
east-1.amazonaws.com:9094

# SSL properties, needed if cluster is using TLS encryption
security.protocol=SSL
ssl.truststore.location=/home/ec2-user/kafka.client.truststore.jks

# Use the Prometheus Metric Sampler
metric.sampler.class=com.linkedin.kafka.cruisecontrol.monitor.sampling.prometheus.Prometheu
```

```
# Prometheus Metric Sampler specific configuration
prometheus.server.endpoint=1.2.3.4:9090 # Replace with your Prometheus IP and port

# Change the capacity config file and specify its path; details below
capacity.config.file=config/capacityCores.json
```

エクスプレスブローカーの場合、アナライザー設定で設定された目標DiskCapacityGoalに を使用しないことをお勧めします。

2. config/capacityCores.json ファイルを編集して、適切なディスクサイズ、CPU コア、およびネットワークの入出力制限を指定します。Express ブローカーの場合、DISK容量エントリは Cruise Control の設定にのみ必要です。MSK は Express ブローカーのすべてのストレージを管理するため、この値を などの非常に高い数値に設定する必要がありますInteger.MAX_VALUE（2147483647）。標準ブローカーの場合、DescribeCluster API オペレーション (または describe-cluster CLI) を使用してディスクサイズを取得できます。CPU コアとネットワークの入出力制限については、Amazon EC2 インスタンスタイプを参照してください。

   Standard broker config/capacityCores.json

   ```
   {
     "brokerCapacities": [
       {
         "brokerId": "-1",
         "capacity": {
           "DISK": "10000",
           "CPU": {
             "num.cores": "2"
           },
           "NW_IN": "5000000",
           "NW_OUT": "5000000"
         },
         "doc": "This is the default capacity. Capacity unit used for disk is in
    MB, cpu is in number of cores, network throughput is in KB."
       }
     ]
   }
   ```

   Express broker config/capacityCores.json

   ```
   {
   ```

```
  "brokerCapacities":[
    {
      "brokerId": "-1",
      "capacity": {
        "DISK": "2147483647",
        "CPU": {"num.cores": "16"},
        "NW_IN": "1073741824",
        "NW_OUT": "1073741824"
      },
      "doc": "This is the default capacity. Capacity unit used for disk is in
 MB, cpu is in number of cores, network throughput is in KB."
    }
  ]
}
```

3. オプションで、Cruise Control UI をインストールできます。ダウンロードするには、Cruise Control Frontend の設定に移動します。
4. 次のコマンドを実行して、Cruise Control をスタートします。screen や tmux などのツールを使用して、長時間実行されるセッションを開いたままにすることを検討してください。

```
<path-to-your-CRUISE-CONTROL-installation>/bin/kafka-cruise-control-start.sh
 config/cruisecontrol.properties 9091
```

5. Cruise Control API または UI を使用して、Cruise Control にクラスター ロードデータがあり、リバランスの提案が行われていることを確認します。メトリクスの有効なウィンドウを取得するには、数分かかる場合があります。

> **Important**
>
> Express ブローカーは Zookeeper エンドポイントを公開しないため、Cruise Control バージョン 2.5.60 以降のみが Express ブローカーと互換性があります。

Cruise Control for Amazon MSK の自動デプロイテンプレートを使用する

この CloudFormation テンプレート を使用して、Cruise Control と Prometheus を簡単にデプロイし、Amazon MSK クラスターのパフォーマンスをより深く把握し、リソース使用率を最適化することもできます。

主な特徴:

- Cruise Control と Prometheus が事前設定された Amazon EC2 インスタンスの自動プロビジョニング。
- Amazon MSK プロビジョニング済みクラスターのサポート。
- PlainText と IAM による柔軟な認証。
- Cruise Control の ZooKeeper 依存関係はありません。
- Amazon S3 バケットに保存されている独自の設定ファイルを指定することで、Prometheus ターゲット、Cruise Control 容量設定、およびその他の設定を簡単にカスタマイズできます。

パーティションの再調整ガイドライン

Kafka パーティションの再割り当てに関するガイドライン

Kafka でのパーティションの再割り当ては、ブローカー間で大量のデータを転送し、ネットワークの輻輳を引き起こし、クライアントオペレーションに影響を与える可能性があるため、リソースを大量に消費する可能性があります。以下のベストプラクティスは、スロットリングレートの調整、同時実行制御の活用、再割り当てタイプの理解により、クラスターオペレーションの中断を最小限に抑えることで、パーティションの再割り当てを効果的に管理するのに役立ちます。

Cruise Control での同時実行の管理

Cruise Control には、パーティションとリーダーシップの動きの同時実行を制御する自動調整パラメータが用意されています。以下のパラメータは、再割り当て中に許容可能な負荷を維持するのに役立ちます。

- 同時パーティション移動の最大数: を定義`num.concurrent.partition.movements.per.broker`して、ブローカー間パーティションの同時移動を制限し、過剰なネットワーク使用率を回避します。

  Example 例

  ```
  num.concurrent.partition.movements.per.broker = 5
  ```

  この設定では、各ブローカーが常に 10 個以下のパーティションを移動するように制限し、ブローカー間で負荷を分散します。

スロットリングを使用して帯域幅を制御する

- スロットルパラメータ: でパーティションの再割り当てを実行する場合kafka-reassign-partitions.sh、 --throttle parameter を使用してブローカー間のデータ移動の最大転送レート (バイト/秒) を設定します。

  Example 例

  ```
  --throttle 5000000
  ```

  これにより、最大帯域幅が 5 MB/秒に設定されます。
- バランススロットリング設定: 適切なスロットリングレートを選択することが不可欠です。

  設定が低すぎると、再割り当てにかなり時間がかかる場合があります。

  設定が高すぎると、クライアントでレイテンシーが増加する可能性があります。
- 控えめなスロットルレートから始めて、クラスターのパフォーマンスモニタリングに基づいて調整します。本番環境に適用する前に選択したスロットルをテストして、最適なバランスを見つけます。

ステージング環境でのテストと検証

本番環境に再割り当てを実装する前に、同様の設定のステージング環境で負荷テストを実行します。これにより、パラメータをファインチューニングし、ライブ本番環境での予期しない影響を最小限に抑えることができます。

Amazon MSK クラスターの設定を更新する

クラスターの設定を更新するには、クラスターが ACTIVE 状態であることを確認します。また、MSK クラスター上のブローカーあたりのパーティション数が the section called “ クラスターの適切なサイズ設定: 標準ブローカーあたりのパーティション数” で説明されている制限を下回っていることを確認する必要があります。これらの制限を超えるクラスターの設定を更新することはできません。

カスタム設定の作成方法、更新可能なプロパティ、既存のクラスター の設定を更新した場合の動作など、MSK 設定について詳細は、「the section called “ブローカー設定”」を参照してください。

トピック

- 設定更新中のブローカーの可用性

- を使用したクラスターの設定の更新 AWS CLI
- API を使用して Amazon MSK クラスターの設定を更新する

設定更新中のブローカーの可用性

Amazon MSK は、ほとんどのクラスター設定の更新中に高可用性を維持します。Amazon MSK はローリング更新を実行し、一度に 1 つのブローカーを更新します。このプロセス中、クラスターは引き続き使用できますが、個々のブローカーは設定が更新されると再起動します。ただし、一部の設定変更では、すべてのブローカーを同時に更新する必要がある場合があります。これにより、クラスター全体の停止が短時間発生する可能性があります。更新中のブローカーの可用性への影響の詳細については、「」を参照してくださいAmazon MSK プロビジョンド設定。

本番クラスターを更新する前に、非本番稼働環境で設定変更をテストし、メンテナンス期間中に更新をスケジュールすることをお勧めします。

MSK クラスターのアップグレード中に問題が発生した場合は、「Amazon MSK クラスターをアップグレードするときに問題をトラブルシューティングするにはどうすればよいですか？」を参照してください。

を使用したクラスターの設定の更新 AWS CLI

1. 次の JSON をコピーして、ファイルに保存します。ファイルを `configuration-info.json` と名付けます。*`ConfigurationArn`* をクラスターの更新に使用する設定の Amazon リソースネーム (ARN) に置き換えます。次の JSON では、ARN 文字列は引用符で囲む必要があります。

   `Configuration-Revision` を、使用する設定のリビジョンに置き換えます。設定のリビジョンは、1 から始まる整数です。次の JSON では、この整数を引用符で囲むことはできません。

   ```
   {
        "Arn": ConfigurationArn,
        "Revision": Configuration-Revision
   }
   ```

2. *`ClusterArn`* をクラスター作成時に取得した ARN に置き換えて、次のコマンドを実行します。クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。詳細については、「the section called “クラスターを一覧表示する”」を参照してください。

Path-to-Config-Info-File を設定情報ファイルのパスに置き換えます。前の手順で作成したファイルを configuration-info.json と命名して現在のディレクトリに保存した場合、*Path-to-Config-Info-File* は configuration-info.json です。

Current-Cluster-Version を、クラスターの現在のバージョンに置き換えます。

> **Important**
>
> クラスターのバージョンは単純な整数ではありません。クラスターの最新バージョンを検索するには、DescribeCluster オペレーションまたは describe-cluster AWS CLI コマンドを使用します。サンプルのバージョンは KTVPDKIKX0DER です。

```
aws kafka update-cluster-configuration --cluster-arn ClusterArn --configuration-
info file://Path-to-Config-Info-File --current-version Current-Cluster-Version
```

次に、このコマンドの使用例を示します。

```
aws kafka update-cluster-configuration --cluster-arn "arn:aws:kafka:us-
east-1:0123456789012:cluster/exampleName/abcd1234-0123-abcd-5678-1234abcd-1" --
configuration-info file://c:\users\tester\msk\configuration-info.json --current-
version "K1X5R6FKA87"
```

この update-cluster-configuration コマンドの出力は、次の JSON の例のようになります。

```
{
    "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
abcdefab-1234-abcd-5678-cdef0123ab01-2",
    "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef"
}
```

3. update-cluster-configuration オペレーションの結果を取得するには、*ClusterOperationArn*を update-cluster-configuration コマンドの出力で取得した ARN に置き換えて、次のコマンドを実行します。

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

この describe-cluster-operation コマンドの出力は、次の JSON の例のようになります。

```
{
    "ClusterOperationInfo": {
        "ClientRequestId": "982168a3-939f-11e9-8a62-538df00285db",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2019-06-20T21:08:57.735Z",
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "UPDATE_COMPLETE",
        "OperationType": "UPDATE_CLUSTER_CONFIGURATION",
        "SourceClusterInfo": {},
        "TargetClusterInfo": {
            "ConfigurationInfo": {
                "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/
ExampleConfigurationName/abcdabcd-abcd-1234-abcd-abcd123e8e8e-1",
                "Revision": 1
            }
        }
    }
}
```

この出力では、OperationType は UPDATE_CLUSTER_CONFIGURATION です。OperationState の値が UPDATE_IN_PROGRESS の場合は、しばらく待ってから再度 describe-cluster-operation コマンドを実行します。

API を使用して Amazon MSK クラスターの設定を更新する

API を使用して Amazon MSK クラスターの設定を更新するには、「UpdateClusterConfiguration」を参照してください。

Amazon MSK クラスターのブローカーを再起動する

この Amazon MSK オペレーションは、お使いの MSK クラスターのブローカーを再起動する際に使用します。クラスターのブローカーを再起動するには、クラスターが `ACTIVE` 状態であることを確認してください。

Amazon MSK サービスは、パッチ適用やバージョンのアップグレードといったシステムメンテナンス中に MSK クラスターのブローカーを再起動することがあります。ブローカーを手動で再起動すると、Kafka クライアントのレジリエンスをテストできるので、システムメンテナンス時にどのような反応が起こるかを判断できます。

を使用して Amazon MSK クラスターのブローカーを再起動する AWS Management Console

このプロセスでは、 を使用して Amazon MSK クラスターのブローカーを再起動する方法について説明します AWS Management Console。

1. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
2. ブローカーを再起動したい MSK クラスターを選択します。
3. [Broker details] (ブローカーの詳細) セクションまでスクロールし、再起動するブローカーを選択します。
4. [Reboot broker] (ブローカーの再起動) を選択します。

を使用して Amazon MSK クラスターのブローカーを再起動する AWS CLI

このプロセスでは、 を使用して Amazon MSK クラスターのブローカーを再起動する方法について説明します AWS CLI。

1. *`ClusterArn`* をクラスターの作成時に取得した Amazon リソースネーム (ARN) に、また *`BrokerId`* を再起動したいブローカーの ID に置き換えて、次のコマンドを実行します。

> **Note**
>
> `reboot-broker` オペレーションは、一度に 1 つのブローカーの再起動のみをサポートします。

クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。詳細については、「the section called “クラスターを一覧表示する”」を参照してください。

クラスターにブローカー ID がない場合は、ブローカーノードを一覧表示することで見つけられます。詳細については、「list-nodes」を参照してください。

```
aws kafka reboot-broker --cluster-arn ClusterArn --broker-ids BrokerId
```

この reboot-broker オペレーションの出力は、次の JSON のようになります。

```
{

    "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
abcdefab-1234-abcd-5678-cdef0123ab01-2",
    "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef"
}
```

2. reboot-broker オペレーションの結果を取得するには、*ClusterOperationArn* を reboot-broker コマンドの出力で取得した ARN に置き換えて、次のコマンドを実行します。

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

この describe-cluster-operation コマンドの出力は、次の JSON の例のようになります。

```
{
    "ClusterOperationInfo": {
        "ClientRequestId": "c0b7af47-8591-45b5-9c0c-909a1a2c99ea",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2019-09-25T23:48:04.794Z",
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "REBOOT_IN_PROGRESS",
        "OperationType": "REBOOT_NODE",
        "SourceClusterInfo": {},
```

```
        "TargetClusterInfo": {}
    }
}
```

再起動オペレーションが完了すると、OperationState が REBOOT_COMPLETE になります。

API を使用して Amazon MSK クラスターのブローカーを再起動する

API を使用してクラスター内のブローカーを再起動するには、「ReBootBroker」を参照してください。

Amazon MSK クラスターをタグ付ける

独自のメタデータは、タグの形で MSK クラスターなどの Amazon MSK リソースに割り当てることができます。タグは、リソースに対して定義するキーと値のペアです。タグの使用は、 AWS リソースを管理し、請求データを含むデータを整理するためのシンプルで強力な方法です。

トピック

- Amazon MSK クラスターのタグの基本
- タグ付けを使用して Amazon MSK クラスターのコストを追跡する
- タグの制限
- Amazon MSK API を使用してリソースをタグ付ける

Amazon MSK クラスターのタグの基本

次のタスクを実行する際に、 Amazon MSK API を使用できます。

- Amazon MSK リソースにタグを追加します。
- Amazon MSK リソースのタグを一覧表示します。
- Amazon MSK リソースからタグを削除します。

タグを使用して Amazon MSK リソースを分類できます。例えば、目的、所有者、環境などに基づいて Amazon MSK クラスターを分類できます。タグごとにキーと値を定義するため、特定の二ーズを満たすためのカテゴリのカスタムセットを作成できます。たとえば、所有者と、関連するアプリケーションに基づいてクラスターを追跡するのに役立つタグのセットを定義できます。

次に示すのは、いくつかのタグの例です。

- `Project: Project name`
- `Owner: Name`
- `Purpose: Load testing`
- `Environment: Production`

タグ付けを使用して Amazon MSK クラスターのコストを追跡する

タグを使用して、 AWS コストを分類および追跡できます。Amazon MSK クラスターを含む リソースに AWS タグを適用すると、 AWS コスト配分レポートにはタグ別に集計された使用量とコストが含まれます。自社のカテゴリ (たとえばコストセンター、アプリケーション名、所有者) を表すタグを適用すると、複数のサービスにわたってコストを分類することができます。詳細については、「AWS Billing ユーザーガイド」の「コスト配分タグを使用したカスタム請求レポート」を参照してください。

タグの制限

Amazon MSK のタグには以下の制限があります。

基本制限

- リソースあたりのタグの最大数は 50 です。
- タグのキーと値は大文字と小文字が区別されます。
- 削除されたリソースのタグを変更または編集することはできません。

タグキーの制限

- 各タグキーは一意である必要があります。既に使用されているキーを含むタグを追加すると、新しいタグで、既存のキーと値のペアが上書きされます。
- `aws:` は AWSが使用するように予約されているため、このプレフィックスを含むタグキーで開始することはできません。 AWS ではユーザーの代わりにこのプレフィックスで始まるタグを作成しますが、ユーザーはこれらのタグを編集または削除することはできません。
- タグキーの長さは 1 ~ 128 文字 (Unicode) にする必要があります。
- タグキーは、次の文字で構成する必要があります。Unicode 文字、数字、空白、特殊文字 (`_ . / = + - @`)。

タグ値の制限

- タグ値の長さは 0～255 文字 (Unicode) にする必要があります。
- タグ値は空白にすることができます。空白にしない場合は、次の文字で構成する必要があります。Unicode 文字、数字、空白、特殊文字 (_ . / = + - @)。

Amazon MSK API を使用してリソースをタグ付ける

次のオペレーションを使用して、Amazon MSK リソースへのタグの追加と削除をしたり、特定の リソースの現在のタグセットを一覧表示したりできます。

- ListTagsForResource
- TagResource
- UntagResource

Amazon MSK クラスターへの移行

MSK クラスターの移行には、Amazon MSK Replicator を使用できます。「Amazon MSK Replicator とは」を参照してください。または、Apache MirrorMaker 2.0 を使用して MSK 以外のクラスターから Amazon MSK クラスターに移行することもできます。この操作を行う方法の例については、「Migrate an on-premises Apache Kafka cluster to Amazon MSK by using MirrorMaker」を参照してください。MirrorMaker の使用方法については、Apache Kafka のドキュメントの「クラスター間のデータのミラーリング」を参照してください。可用性の高い設定で MirrorMaker をセットアップすることをお勧めします。

MirrorMaker を使用して MSK クラスターに移行する際に従う手順の概要

1. 宛先 MSK クラスターを作成します
2. 宛先クラスターと同じ Amazon VPC 内の Amazon EC2 インスタンスから MirrorMaker をスタートします。
3. MirrorMaker の遅延を調べます。
4. MirrorMaker が追いついたら、MSK クラスターブートストラップブローカーを使用して、プロデューサーとコンシューマーを新しいクラスターにリダイレクトします。
5. MirrorMaker をシャットダウンします。

Apache Kafka クラスターを Amazon MSK に移行する

CLUSTER_ONPREM という名前の Apache Kafka クラスターがあるとします。そのクラスターには、トピックとデータが入力されます。そのクラスターを CLUSTER_AWSMSK という名前の新しく作成された Amazon MSK クラスターに移行する場合、この手順では、実行する必要のあるステップの概要を示します。

既存の Apache Kafka クラスターを Amazon MSK に移行するには

1. CLUSTER_AWSMSK で、移行するすべてのトピックを作成します。

   適切なレプリケーションレベルで移行するトピックが自動的に再作成されないため、このステップでは MirrorMaker を使用できません。CLUSTER_ONPREM で持っていたのと同じレプリケーション係数とパーティション数を使用して、Amazon MSK でトピックを作成できます。また、異なるレプリケーション係数とパーティション数を使用してトピックを作成することもできます。
2. CLUSTER_ONPREM への読み取りアクセス権 と CLUSTER_AWSMSK への書き込みアクセス権を持つインスタンスから MirrorMaker をスタートします。
3. 以下のコマンドを実行して、すべてのトピックをミラーリングします。

   ```
   <path-to-your-kafka-installation>/bin/kafka-mirror-maker.sh --consumer.config
    config/mirrormaker-consumer.properties --producer.config config/mirrormaker-
   producer.properties --whitelist '.*'
   ```

   このコマンドでは、config/mirrormaker-consumer.properties は、CLUSTER_ONPREM のブートストラップブローカーを指します。例えば、bootstrap.servers=localhost:9092 です。また、config/mirrormaker-producer.properties は、CLUSTER_AWSMSK のブートストラップブローカーを指します。例えば、bootstrap.servers=10.0.0.237:9092,10.0.2.196:9092,10.0.1.233:9092 です。
4. MirrorMaker をバックグラウンドで実行したままにし、CLUSTER_ONPREM を引き続き使用します。MirrorMaker は、すべての新しいデータをミラーリングします。
5. ミラーリングの進行状況をチェックするには、各トピックの最後のオフセットと、MirrorMaker が消費する現在のオフセットの間の遅延を調べます。

   MirrorMaker は、単に消費者とプロデューサーを使用していることに注意してください。したがって、kafka-consumer-groups.sh ツールを使用して遅延をチェックすること

ができます。コンシューマーグループ名を検索するには、group.id の mirrormaker-consumer.properties ファイル内を検索し、その値を使用します。ファイルにそのようなキーがない場合は、作成することができます。たとえば、group.id=mirrormaker-consumer-group を設定します。

6. MirrorMaker は、すべてのトピックのミラーリングを終了した後、すべてのプロデューサーとコンシューマーを停止し、MirrorMaker を停止します。次に、プロデューサーおよびコンシューマーブートストラップブローカーの値を変更して、プロデューサーとコンシューマーを CLUSTER_AWSMSK クラスターにリダイレクトします。CLUSTER_AWSMSK のすべてのプロデューサーとコンシューマーを再起動します。

1 つの Amazon MSK クラスターから別の Amazon MSK クラスターに移行する

Apache MirrorMaker 2.0 を使用して、MSK 以外のクラスターから MSK クラスターに移行できます。たとえば、Apache Kafka の 1 つのバージョンから別のバージョンに移行できます。この操作を行う方法の例については、「Migrate an on-premises Apache Kafka cluster to Amazon MSK by using MirrorMaker」を参照してください。または、Amazon MSK Replicator を使用して MSK クラスターの移行を行うこともできます。Amazon MSK Replicator の詳細については、「Amazon MSK Replicator とは」を参照してください。

MirrorMaker 1.0 のベストプラクティス

このベストプラクティスの一覧は、MirrorMaker 1.0 に適用されます。

- 送信先クラスターで MirrorMaker を実行します。この方法では、ネットワークの問題が発生しても、メッセージはソースクラスターで引き続き使用できます。ソースクラスターで MirrorMaker を実行し、プロデューサーでイベントがバッファされ、ネットワークの問題が発生すると、イベントが失われる可能性があります。
- 転送中に暗号化が必要な場合は、ソースクラスターで実行します。
- コンシューマーの場合は、auto.commit.enabled=false を設定します
- プロデューサーの場合は、以下を設定します。
  - max.in.flight.requests.per.connection=1
  - retries=Int.Max_Value
  - acks=all
  - max.block.ms = Long.Max_Value
- 生産者のスループットが高い場合:

  - メッセージのバッファリングとメッセージバッチの入力 — buffer.memory、batch.size、linger.ms を調整します
  - ソケットバッファの調整 — receive.buffer.bytes、send.buffer.bytes
- データの損失を回避するには、ソースで自動コミットをオフにして、MirrorMaker がコミットを制御できるようにします。これは、通常、送信先クラスターから ACK を受け取った後に行います。プロデューサーに acks=all があり、送信先クラスターに min.insync.replicas が 1 を超えて設定されている場合、MirrorMaker コンシューマーがソースでオフセットをコミットする前に、メッセージは送信先の複数のブローカーで永続化されます。
- 順序が重要な場合は、再試行を 0 に設定できます。また、本番環境では、最大インフライト接続数を 1 に設定して、バッチが途中で失敗した場合に、送信されるバッチが順不同でコミットされないようにします。このようにして、送信される各バッチは、次のバッチが送信されるまで再試行されます。max.block.ms が最大値に設定されておらず、プロデューサーバッファがいっぱいになると、(他の設定によっては)データが失われる可能性があります。これは、消費者をブロックし、バックプレッシャーすることができます。
- 高スループット
  - buffer.memory を増やします。
  - バッチサイズを大きくします。
  - linger.ms を調整して、バッチがいっぱいになるようにします。これにより、圧縮率が向上し、ネットワーク帯域幅の使用率を削減して、クラスター上のストレージの使用量が少なくなります。これにより、保存期間が向上します。
  - CPU とメモリの使用状況をモニタリングします。
- 高いコンシューマースループット
  - MirrorMaker プロセスごとのスレッド/コンシューマーの数を増やします — num.streams。
  - スレッドを増やす前に、マシン間で MirrorMaker プロセスの数を増やし、高可用性を実現します。
  - 最初に同じマシン上で MirrorMaker プロセスの数を増やし、次に同じグループ ID を持つ異なるマシン上でプロセスを増やします。
  - スループットが非常に高く、個別の MirrorMaker インスタンスを使用するトピックを分離します。
- 管理と構成
  - Chef AWS CloudFormation や Ansible などの および 設定管理ツールを使用します。
  - Amazon EFS マウントを使用して、すべての Amazon EC2 インスタンスからすべての構成ファイルにアクセスできるようにします。

  - コンテナを使用して、MirrorMaker インスタンスのスケーリングおよび管理を容易にします。
- 通常、MirrorMaker でプロデューサーを飽和させるには複数のコンシューマーが必要です。したがって、複数のコンシューマーを設定します。まず、高可用性を実現するために、異なるマシンにそれらを設定します。次に、各パーティションのコンシューマーを持つように個々のマシンをスケールアップし、コンシューマーをマシン間で均等に分散させます。
- スループットの取り込みと配信を高くするには、受信バッファと送信バッファのデフォルトが低すぎる可能性があるため、受信バッファと送信バッファを調整します。パフォーマンスを最大化するには、ストリーミングの総数 (num.streams) が、MirrorMaker が送信先クラスターにコピーしようとしているすべてのトピック パーティションと一致することを確認します。

MirrorMaker 2 の利点。*

- Apache Kafka Connect フレームワークとエコシステムを利用します。
- 新しいトピックとパーティションを検出します。
- クラスター間でトピック構成を自動的に同期します。
- 「アクティブ/アクティブ」クラスターペアと、任意の数のアクティブクラスターをサポートします。
- 複数のデータセンターおよびクラスターにわたるエンドツーエンドのレプリケーションレイテンシーなどの新しいメトリクスを提供します。
- クラスター間でコンシューマーを移行するために必要なオフセットを放出し、オフセット変換用のツールを提供します。
- 各 MirrorMaker 1.* プロセスの低レベルのプロデューサー/コンシューマープロパティと比較して、複数のクラスターとレプリケーションフローを 1 か所で指定するための高レベル設定ファイルをサポートします。

Amazon MSK プロビジョンドクラスターを削除する

> **Note**
>
> プロビジョニングされた Amazon MSK クラスターに自動スケーリングポリシーがある場合は、クラスターを削除する前にそのポリシーを削除することをお勧めします。詳細については、「Amazon MSK クラスターの自動スケーリング」を参照してください。

トピック

- を使用して Amazon MSK プロビジョンドクラスターを削除する AWS Management Console
- を使用して Amazon MSK プロビジョンドクラスターを削除する AWS CLI
- API を使用して Amazon MSK プロビジョンドクラスターを削除する

を使用して Amazon MSK プロビジョンドクラスターを削除する AWS Management Console

このプロセスでは、 を使用して Amazon MSK プロビジョンドクラスターを削除する方法について説明します AWS Management Console。MSK クラスターを削除する前に、クラスターに保存されている重要なデータのバックアップがあり、クラスターに依存するスケジュールされたタスクがないことを確認してください。MSK クラスターの削除を元に戻すことはできません。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. 削除したい MSK クラスターの隣にあるボックスをチェックして選択します。
3. Delete (削除) を選択し、確定します。

を使用して Amazon MSK プロビジョンドクラスターを削除する AWS CLI

このプロセスでは、 を使用して MSK プロビジョンドクラスターを削除する方法について説明します AWS CLI。MSK クラスターを削除する前に、クラスターに保存されている重要なデータのバックアップがあり、クラスターに依存するスケジュールされたタスクがないことを確認してください。MSK クラスターの削除を元に戻すことはできません。

*ClusterArn*をクラスターの作成時に取得したAmazon リソースネーム (ARN) に置き換えて、次のコマンドを実行します。クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。詳細については、「the section called “クラスターを一覧表示する”」を参照してください。

```
aws kafka delete-cluster --cluster-arn ClusterArn
```

API を使用して Amazon MSK プロビジョンドクラスターを削除する

Amazon MSK API を使用すると、自動インフラストラクチャプロビジョニングまたはデプロイスクリプトの一部として MSK プロビジョンドクラスターをプログラムで作成および管理できます。このプロセスでは、Amazon MSK API を使用して Amazon MSK プロビジョンドクラスターを削除する方

法について説明します。Amazon MSK クラスターを削除する前に、クラスターに保存されている重要なデータのバックアップがあり、クラスターに依存するスケジュールされたタスクがないことを確認してください。MSK クラスターの削除を元に戻すことはできません。

API を使用してクラスターを削除するには、「DeleteCluster」を参照してください。

Amazon MSK の主な機能と概念

Amazon MSK プロビジョンドクラスターには、クラスターのパフォーマンスを最適化し、ストリーミングニーズを満たすのに役立つさまざまな機能が用意されています。以下のトピックでは、これらの機能について詳しく説明します。

- AWS Management Console
- Amazon MSK API リファレンス
- Amazon MSK CLI コマンドリファレンス

トピック

- Amazon MSK ブローカータイプ
- Amazon MSK のブローカーサイズ
- 標準ブローカーのストレージ管理
- Amazon MSK のセキュリティ
- Amazon MSK プロビジョンド設定
- パッチ適用
- ブローカーのオフラインおよびクライアントフェイルオーバー
- Amazon MSK ログ記録
- メタデータ管理
- Amazon MSK リソース
- Apache Kafka バージョン
- Amazon MSK クラスターをトラブルシューティングする

Amazon MSK ブローカータイプ

MSK Provisioned には、Standard と Express の 2 つのブローカータイプがあります。標準ブローカーはクラスターを柔軟に設定でき、Express ブローカーは高性能ストリーミングアプリケーショ

ンを実行するための伸縮性、スループット、耐障害性、ease-of-useを提供します。各サービスの詳細については、以下のサブセクションを参照してください。以下の表では、Standard ブローカーと Express ブローカーの主な機能の比較も示しています。

MSK プロビジョンドブローカータイプの比較

機能	標準ブローカー	Express ブローカー
ストレージ管理	カスタマーマネージド (EBS ストレージ、階層型ストレージ、プロビジョンドストレージスループット、自動スケーリング、ストレージ容量アラートなどの機能)	完全な MSK 管理
サポートされているインスタンス	T3, M5, M7g	M7g
サイジングとスケーリングに関する考慮事項	スループット、接続、パーティション、ストレージ	スループット、接続、パーティション
ブローカーのスケーリング	垂直スケーリングと水平スケーリング	垂直スケーリングと水平スケーリング
Kafka バージョン	「Apache Kafka バージョン」を参照してください。	バージョン 3.6 から開始
Apache Kafka の設定	より設定可能	主に MSK 管理による耐障害性の向上
セキュリティ	暗号化、プライベート/パブリックアクセス、認証と認可 - IAM、SASL/SCRAM、mTLS、プレーンテキスト、Kafka ACLs	暗号化、プライベート/パブリックアクセス、認証と認可 - IAM、SASL/SCRAM、mTLS、プレーンテキスト、Kafka ACLs
モニタリング	CloudWatch、Open Monitoring	CloudWatch、Open Monitoring

> **Note**
>
> MSK API を使用してブローカータイプを切り替えることで、MSK プロビジョンドクラスターをスタンダードブローカータイプから Express ブローカータイプに変更することはできません。必要なブローカータイプ (Standard または Express) で新しいクラスターを作成する必要があります。

トピック

- Amazon MSK Standard ブローカー
- Amazon MSK Express ブローカー

Amazon MSK Standard ブローカー

MSK プロビジョンドの標準ブローカーは、クラスターのパフォーマンスを最も柔軟に設定できます。幅広いクラスター設定から選択して、アプリケーションに必要な可用性、耐久性、スループット、レイテンシーの特性を実現できます。ストレージ容量をプロビジョニングし、必要に応じて増やすこともできます。Amazon MSK は、スタンダードブローカーとアタッチされたストレージリソースのハードウェアメンテナンスを処理し、発生する可能性のあるハードウェアの問題を自動的に修復します。このドキュメントでは、ストレージ管理、設定、メンテナンスに関するトピックなど、スタンダードブローカーに関連するさまざまなトピックについて詳しく説明します。

Amazon MSK Express ブローカー

MSK プロビジョンドの Express ブローカーを使用すると、Apache Kafka の管理が簡単になり、大規模に実行するためのコスト効率が向上し、予想される低レイテンシーで伸縮性が向上します。ブローカーには、自動的にスケールするpay-as-you-goのストレージが含まれており、サイジング、プロビジョニング、プロアクティブモニタリングは必要ありません。選択したインスタンスサイズに応じて、各ブローカーノードは、標準の Apache Kafka ブローカーと比較して、ブローカーあたり最大 3 倍のスループットを提供し、最大 20 倍速くスケールアップし、90% 速く復旧できます。Express ブローカーには、Amazon MSK のベストプラクティスのデフォルトが事前設定されており、クライアントスループットクォータを適用して、クライアントと Kafka のバックグラウンドオペレーション間のリソース競合を最小限に抑えます。

Express ブローカーを使用する際に考慮すべき重要な要因と機能をいくつか紹介します。

- ストレージ管理なし: Express ブローカーを使用すると、ストレージリソースをプロビジョニングまたは管理する必要がなくなります。伸縮自在で実質無制限、pay-as-you-go、フルマネージド型

のストレージを利用できます。高スループットのユースケースでは、コンピューティングインスタンスとストレージボリューム間の相互作用、および関連するスループットのボトルネックについて理由を説明する必要はありません。これらの機能により、クラスター管理が簡素化され、ストレージ管理の運用オーバーヘッドがなくなります。

- 高速スケーリング: Express ブローカーを使用すると、クラスターをスケーリングし、標準ブローカーよりも最大 20 倍速くパーティションを移動できます。この機能は、今後の負荷の急増に対応するためにクラスターをスケールアウトしたり、コストを削減するためにクラスターをスケールインしたりする必要がある場合に不可欠です。クラスターのスケーリングの詳細については、クラスターの拡張、ブローカーの削除、パーティションの再割り当て、再調整のための LinkedIn の Cruise Control の設定に関するセクションを参照してください。
- スループットの向上: Express ブローカーは、標準ブローカーよりもブローカーあたり最大 3 倍高いスループットを提供します。例えば、m7g.16xlarge サイズの Express ブローカーごとに最大 500 MBps で安全にデータを書き込むことができます。これに対して、同等のスタンダードブローカーでは 153.8 MBps です (どちらの番号も、レプリケーションや再調整などのバックグラウンドオペレーションに十分な帯域幅を割り当てることを前提としています)。
- 高いレジリエンスのために設定: Express ブローカーは、クラスターのレジリエンスを向上させるためのさまざまなベストプラクティスを自動的に提供します。これには、重要な Apache Kafka 設定のガードレール、スループットクォータ、バックグラウンドオペレーションや予期しない修復のためのキャパシティ予約が含まれます。これらの機能により、大規模な Apache Kafka アプリケーションを安全かつ簡単に実行できます。詳細については、Express ブローカーの設定「」および「」のセクションAmazon MSK Express ブローカークォータを参照してください。
- メンテナンスウィンドウなし: Express ブローカーのメンテナンスウィンドウはありません。Amazon MSK はクラスターハードウェアを継続的に更新します。詳細については、「Express ブローカーのパッチ適用」を参照してください。

Express ブローカーに関する追加情報

- Express ブローカーは Apache Kafka APIs を使用しますが、KStreams API はまだ完全にはサポートしていません。
- Express ブローカーは 3AZs 設定でのみ使用できます。
- Express ブローカーは、一部のインスタンスサイズでのみ使用できます。更新されたリストについては、「Amazon MSK の料金」を参照してください。
- Express ブローカーは、Apache Kafka バージョン 3.6 および 3.8 でサポートされています。

これらのブログを見る

MSK Express ブローカーの詳細と使用中の Express ブローカーの実際の例については、以下のブログを参照してください。

- Amazon MSK 用の Express ブローカーを導入して、Kafka クラスターに高スループットと高速スケーリングを実現
- Amazon MSK の Express ブローカー: 最大 20 倍の高速パフォーマンスで Turbo-charged Kafka スケーリング

このブログでは、Express ブローカーがどのように機能するかを示します。

- スループットの高速化、迅速なスケーリング、障害からの復旧時間の向上を実現
- ストレージ管理の複雑さを排除する

Amazon MSK のブローカーサイズ

Amazon MSK プロビジョンドクラスターを作成するときは、必要なブローカーのサイズを指定します。ブローカータイプに応じて、Amazon MSK は次のブローカーサイズをサポートします。

標準ブローカーサイズ

- kafka.t3.small
- kafka.m5.large、kafka.m5.xlarge、kafka.m5.2xlarge、kafka.m5.4xlarge、kafka.m5.8xlarge、kafka.m5.12
- kafka.m7g.large、kafka.m7g.xlarge、kafka.m7g.2xlarge、kafka.m7g.4xlarge、kafka.m7g.8xlarge、kafka.

Express ブローカーのサイズ

- express.m7g.large、 express.m7g.xlarge、 express.m7g.2xlarge、 express.m7g.4xlarge、 express.m7g.8xlarge、 express.m7g.12xlarge、 express.m7g.16xlarge

Note

一部のブローカーサイズは、certian AWS リージョンでは使用できない場合があります。利用可能なインスタンスのリージョン別の最新のリストについては、Amazon MSK 料金ページの更新されたブローカーインスタンス料金表を参照してください。

ブローカーサイズに関するその他の注意事項

- M7g ブローカーは AWS Graviton プロセッサ (Amazon Web Services によって構築されたカスタム Arm ベースのプロセッサ) を使用します。M7g ブローカーでは、同等の M5 インスタンスと比較して、価格パフォーマンスが向上しています。M7g ブローカーでは、同等の M5 インスタンスよりも消費電力が少なくなります。
- Amazon MSK は、2.8.2 および 3.3.2 以降の Kafka バージョンを実行している MSK プロビジョンドクラスターで M7g ブローカーをサポートします。
- M7g および M5 ブローカーは、T3 ブローカーよりも高いベースラインのスループットパフォーマンスを備えており、本稼働ワークロードに推奨されます。M7g および M5 ブローカーはまた、ブローカーあたり T3 ブローカーよりも多くのパーティションを持つことができます。より大きな本稼働グレードのワークロードを実行している場合、またはより多くのパーティションを必要とする場合は、M7g または M5 ブローカーを使用します。M7g および M5 インスタンスのサイズについての詳細は、「Amazon EC2 汎用インスタンス」を参照してください。
- T3 ブローカーには、CPU クレジットを使用して一時的にパフォーマンスをバーストする機能があります。小規模から中規模のストリーミングワークロードをテストする場合、またはスループットが一時的に急上昇する低スループットのストリーミングワークロードがある場合は、低コストの開発のために T3 ブローカーを使用します。T3 ブローカーが本稼働ワークロードまたは重要なワークロードに対して十分であるかどうかを判断するために、概念実証テストを実行することをお勧めします。T3 ブローカーのサイズについての詳細は、「Amazon EC2 T3 インスタンス」を参照してください。

ブローカーサイズの選択方法についての詳細は、「Standard ブローカーと Express ブローカーのベストプラクティス」を参照してください。

標準ブローカーのストレージ管理

Amazon MSK には、MSK クラスターのストレージ管理に役立つ機能が用意されています。

> **Note**
>
> Express ブローカーを使用すると、データに使用されるストレージリソースをプロビジョニングまたは管理する必要はありません。これにより、クラスター管理が簡素化され、Apache Kafka クラスターの運用上の問題の一般的な原因の 1 つが排除されます。また、アイドル状態のストレージ容量をプロビジョニングする必要はなく、使用した分だけ支払うため、支出も少なくなります。

標準ブローカータイプ

標準ブローカーでは、さまざまなストレージオプションと機能から選択できます。Amazon MSK には、MSK クラスターのストレージ管理に役立つ機能が用意されています。

スループットの管理については、「」を参照してください???。

トピック

- 標準ブローカーの階層型ストレージ
- Amazon MSK Standard ブローカーストレージのスケールアップ
- Amazon MSK クラスター内の標準ブローカーのストレージスループットを管理する

標準ブローカーの階層型ストレージ

階層型ストレージは Amazon MSK 用の低コストのストレージ階層で、実質的に無制限にストレージをスケーリングできるため、ストリーミングデータアプリケーションの構築を費用対効果の高い方法で行うことができます。

パフォーマンスとコストのバランスをとる階層型ストレージを使用して構成された Amazon MSK クラスターを作成できます。Amazon MSK は、Apache Kafka トピックの保持制限に達するまで、パフォーマンスが最適化されたプライマリストレージ階層にストリーミングデータを保存します。その後、Amazon MSK は新しい低コストのストレージ階層に自動的にデータを移動します。

アプリケーションが階層型ストレージからのデータの読み取りを開始すると、最初の数バイトは読み取りレイテンシーが大きくなることが予想されます。残りのデータを低コスト階層から順次読み取り始めると、プライマリストレージ階層と同様のレイテンシーになることが予想されます。低コストの階層型ストレージ用にストレージをプロビジョニングしたり、インフラストラクチャを管理したりする必要はありません。任意の量のデータを保存することができ、使用量に応じた料金のみが発生します。この機能は、KIP-405: Kafka 階層型ストレージで導入された API と互換性があります。

MSK 階層型ストレージクラスターのサイズ設定、モニタリング、最適化の詳細については、「Amazon MSK 階層型ストレージを使用して本稼働ワークロードを実行するためのベストプラクティス」を参照してください。

階層型ストレージには、次のような特徴があります。

- 実質的に無制限にストレージをスケーリングできます。Apache Kafka インフラストラクチャをスケーリングする方法を考える必要はありません。

- ブローカーの数を増やすことなく、Apache Kafka トピックのデータをより長く保持したり、トピックのストレージを増やしたりできます。
- これにより、安全バッファーが長くなり、処理中の予期しない遅延に対処できるようになります。
- 既存のストリーム処理コードと Kafka API を使用して、古いデータを正確な生成順序で再処理できます。
- セカンダリストレージのデータをブローカーディスク間でレプリケートする必要がないため、パーティションの再調整が高速化されます。
- ブローカーと階層型ストレージ間のデータは VPC 内を移動し、インターネットを経由しません。
- クライアントマシンは、階層型ストレージが有効になっていないクラスターに接続するのと同じプロセスを使用して、階層型ストレージが有効になっている新しいクラスターに接続することができます。「クライアントマシンを作成する」を参照してください。

Amazon MSK クラスターの階層型ストレージの要件

- 階層型ストレージを有効にして新しいトピックを作成するには、Apache Kafka クライアントのバージョン 3.0.0 以降を使用する必要があります。既存のトピックを階層型ストレージに移行するには、バージョン 3.0.0 より前の Kafka クライアント (サポートされている Apache Kafka の最小バージョンは 2.8.2.tiered) を使用するクライアントマシンを再構成して、階層型ストレージを有効にすることができます。「ステップ 4: Amazon MSK クラスターにトピックを作成する」を参照してください。
- 階層型ストレージが有効になっている Amazon MSK クラスターでは、バージョン 3.6.0 以降または 2.8.2.tiered を使用する必要があります。

Amazon MSK クラスターの階層型ストレージの制約と制限

階層型ストレージには、次の制約と制限があります。

- アプリケーションがトランザクション機能をアクティブに使用していない限り、Amazon MSK の remote_tier から読み取りを行う際にクライアントが `read_committed` に設定されていないことを確認してください。
- 階層型ストレージは、 AWS GovCloud (米国) リージョンでは利用できません。
- 階層型ストレージは、プロビジョンドモードのクラスターにのみ適用されます。
- 階層型ストレージでは、ブローカーサイズ t3.small はサポートされていません。
- 低コストストレージでの最小保持期間は 3 日間です。プライマリストレージには最小保持期間はありません。

- 階層型ストレージは、ブローカーの複数のログディレクトリをサポートしていません (JBOD 関連機能)。
- 階層型ストレージは、圧縮されたトピックをサポートしていません。階層型ストレージが有効になっているすべてのトピックで cleanup.policy が「DELETE」のみに設定されていることを確認します。
- 階層型ストレージクラスターは、トピックの作成後の log.cleanup.policy ポリシーの変更をサポートしていません。
- 階層型ストレージは、個々のトピックでは無効にできますが、クラスター全体では無効にできません。いったん無効にすると、トピックに対して階層型ストレージを再度有効にすることはできません。
- Amazon MSK バージョン 2.8.2.tiered を使用している場合、別の階層ストレージがサポートされている Apache Kafka バージョンにのみ移行できます。階層ストレージがサポートされているバージョンをこれ以上使用したくない場合は、新しい MSK クラスターを作成し、データをそのクラスターに移行してください。
- kafka-log-dirs ツールは、階層型ストレージのデータサイズを報告できません。このツールは、プライマリストレージ内のログセグメントのサイズのみを報告します。

トピックレベルで階層型ストレージを設定するときに注意する必要があるデフォルト設定と制約については、「」を参照してくださいAmazon MSK 階層型ストレージのトピックレベル設定に関するガイドライン。

Amazon MSK トピックの階層型ストレージにログセグメントをコピーする方法

新規または既存のトピックに対して階層型ストレージを有効にすると、Apache Kafka はクローズ済みのログセグメントをプライマリストレージから階層型ストレージにコピーします。

- Apache Kafka は、クローズ済みのログセグメントのみをコピーします。ログセグメント内のすべてのメッセージを階層型ストレージにコピーします。
- アクティブセグメントは階層化の対象ではありません。ログセグメントサイズ (segment.bytes) またはセグメントロール時間 (segment.ms) によって、セグメントをクローズする速度と、その後 Apache Kafka がそれらを階層型ストレージにコピーする速度が制御されます。

階層型ストレージが有効になっているトピックの保持設定は、階層型ストレージが有効になっていないトピックの設定とは異なります。次のルールは、階層型ストレージが有効になっているトピックでのメッセージの保持を制御します。

- Apache Kafka では、log.retention.ms (時間) と log.retention.bytes (サイズ) という 2 つの設定で保持を定義します。これらの設定によって、Apache Kafka がクラスターに保持するデータの合計期間とサイズが決まります。階層型ストレージモードを有効にするかどうかにかかわらず、これらの設定はクラスターレベルで設定します。トピックレベルの設定はトピック設定でオーバーライドできます。
- 階層型ストレージを有効にすると、プライマリの高性能ストレージ階層にデータを保存する期間を追加で指定できます。例えば、トピックの全体保持期間 (log.retention.ms) が 7 日間、ローカル保持期間 (local.retention.ms) が 12 時間に設定されている場合、クラスターのプライマリストレージは最初の 12 時間のみデータを保持します。低コストのストレージ階層では、7 日間フルにデータを保持します。
- 通常の保持設定はフルログに適用されます。これには階層型の部分とプライマリの部分が含まれます。
- local.retention.ms 設定または local.retention.bytes 設定は、プライマリストレージでのメッセージの保持を制御します。フルログのデータがプライマリストレージ保持設定のしきい値 (local.retention.ms/bytes) に達すると、Apache Kafka はプライマリストレージ内のデータを階層型ストレージにコピーします。その後、データは有効期限切れの対象になります。
- Apache Kafka がログセグメント内のメッセージを階層型ストレージにコピーすると、retention.ms または retention.bytes の設定に基づいてクラスターからメッセージが削除されます。

Amazon MSK における階層型ストレージのシナリオ例

このシナリオは、階層型ストレージが有効になっている場合に、プライマリストレージにメッセージを持つ既存のトピックがどのように動作するかを示しています。remote.storage.enable を `true` に設定すると、このトピックで階層型ストレージが有効になります。この例では、retention.ms は 5 日間に設定され、local.retention.ms は 2 日間に設定されています。セグメントの有効期限が切れたときの一連のイベントは次のとおりです。

Time T0 - 階層型ストレージを有効にする前。

このトピックに対して階層型ストレージを有効にする前に、2 つのログセグメントがあります。セグメントの 1 つは、既存のトピックパーティション 0 に対してアクティブです。

Time T1 (2 日未満) - 階層型ストレージが有効。セグメント 0 が階層型ストレージにコピーされます。

このトピックに対して階層型ストレージを有効にすると、Apache Kafka は、セグメントが初期保持設定を満たした後に、ログセグメント 0 を階層型ストレージにコピーします。Apache Kafka は、

セグメント 0 のプライマリストレージコピーも保持します。アクティブセグメント 1 は、まだ階層型ストレージへのコピー対象ではありません。このタイムラインではまだ、Amazon MSK によりセグメント 0 とセグメント 1 のメッセージに保持設定が適用されていません (local.retention.bytes/ms, retention.ms/bytes)。

Time T2 - ローカル保持が有効。

2 日後、Apache Kafka が階層型ストレージにコピーしたセグメント 0 に対して、プライマリ保持設定が有効になります。local.retention.ms を 2 日間に設定することでこれを決定します。これで、セグメント 0 はプライマリストレージから有効期限切れになります。アクティブセグメント 1 は、まだ有効期限の対象ではなく、階層型ストレージへのコピー対象でもありません。

Time T3 - 全体保持が有効。

5 日後、保持設定が有効になり、Kafka はログセグメント 0 と関連メッセージを階層型ストレージから消去します。セグメント 1 はアクティブであるため、まだ有効期限の対象ではなく、階層型ストレージへのコピー対象でもありません。セグメント 1 はまだクローズされていないため、セグメントロールの対象ではありません。

を使用して階層型ストレージで Amazon MSK クラスターを作成する AWS Management Console

このプロセスでは、 AWS Management Consoleを使用して階層型ストレージの Amazon MSK クラスターを作成する方法について説明します。

1. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
2. Create cluster (クラスターの作成) を選択します。
3. 階層型ストレージに対して [カスタム作成] を選択します。
4. クラスターの名前を指定します。
5. [クラスタータイプ] で [プロビジョンド] を選択します。
6. クラスターを作成するために使用する Amazon MSK の階層型ストレージをサポートする Amazon Kafka のバージョンを選択します。
7. kafka.t3.small 以外のサイズのブローカーを指定します。
8. Amazon MSK で各アベイラビリティーゾーンに作成するブローカーの数を選択します。ブローカーの最小数は各アベイラビリティーゾーンにつき 1 つで、最大数は各クラスターにつき 30 です。

9. ブローカーを分散させるゾーンの数を指定します。
10. ゾーンごとにデプロイされる Apache Kafka ブローカーの数を指定します。
11. [ストレージオプション] を選択します。これには、階層型ストレージモードを有効にするための [階層化ストレージ と EBS ストレージ] が含まれます。
12. クラスター作成ウィザードの残りのステップを実行します。完了すると、[確認と作成] ビューに、クラスターストレージモードとして [階層化ストレージ と EBS ストレージ] が表示されます。
13. [Create cluster (クラスターの作成)] を選択します。

で階層型ストレージを使用して Amazon MSK クラスターを作成する AWS CLI

クラスターで階層型ストレージを有効にするには、階層型ストレージに適切な Apache Kafka バージョンと属性を使用してクラスターを作成します。以下のコード例に従います。また、次のセクションの「で階層型ストレージを有効にした Kafka トピックを作成する AWS CLI」のステップを実行します。

クラスターの作成でサポートされる属性の完全なリストについては、「create-cluster」を参照してください。

```
aws kafka create-cluster \
 –cluster-name "MessagingCluster" \
 –broker-node-group-info file://brokernodegroupinfo.json \
 –number-of-broker-nodes 3 \
--kafka-version "3.6.0" \
--storage-mode "TIERED"
```

で階層型ストレージを有効にした Kafka トピックを作成する AWS CLI

階層型ストレージを有効にしてクラスターを作成したときに開始したプロセスを完了するには、後述のコード例に示された属性を指定して階層型ストレージを有効にしたトピックも作成します。階層型ストレージ特有の属性には、次のものがあります。

- 時間ベースの保持設定に使用する `local.retention.ms` (10 分間など)、または口グセグメントのサイズ制限に使用する `local.retention.bytes`。
- 階層型ストレージが有効にするには、`remote.storage.enable` を `true` に設定します。

以下の設定では local.retention.ms を使用していますが、この属性を local.retention.bytes に置き換えることもできます。この属性は、Apache Kafka がデータをプライマリストレージから階層型スト

レージにコピーするまでに、待機できる時間またはコピーできるバイト数を制御します。サポートされている設定属性の詳細については、「トピックレベルの設定」を参照してください。

> **Note**
>
> Apache Kafka クライアントのバージョン 3.0.0 以降を使用する必要があります。これらのバージョンでは、`remote.storage.enable` という設定をサポートします。これは、当該クライアントバージョンの `kafka-topics.sh` にのみ含まれます。以前のバージョンの Apache Kafka を使用する既存のトピックで階層型ストレージを有効にするには、「既存の Amazon MSK トピックでの階層型ストレージの有効化」セクションを参照してください。

```
bin/kafka-topics.sh --create --bootstrap-server $bs --replication-factor 2
 --partitions 6 --topic MSKTutorialTopic --config remote.storage.enable=true
 --config local.retention.ms=100000 --config retention.ms=604800000 --config
 segment.bytes=134217728
```

既存の Amazon MSK トピックで階層型ストレージを有効または無効にする

これらのセクションでは、作成済みのトピックで階層型ストレージを有効または無効にする方法について説明します。階層型ストレージを有効にした新しいクラスターとトピックを作成するには、「AWS Management Consoleを使用して階層型ストレージを持つ Amazon MSK クラスターを作成する」を参照してください。

既存の Amazon MSK トピックでの階層型ストレージの有効化

既存のトピックで階層型ストレージを有効にするには、次の例に示す `alter` コマンド構文を使用します。既存のトピックで階層型ストレージを有効にする場合は、特定の Apache Kafka クライアントバージョンに制限されません。

```
bin/kafka-configs.sh --bootstrap-server $bsrv --alter --entity-type topics
 --entity-name msk-ts-topic --add-config 'remote.storage.enable=true,
 local.retention.ms=604800000, retention.ms=15550000000'
```

既存の Amazon MSK トピックの階層型ストレージを無効にする

既存のトピックで階層型ストレージを無効にするには、階層型ストレージを有効にするときと同じ順序で `alter` コマンド構文を使用します。

```
bin/kafka-configs.sh --bootstrap-server $bs --alter --entity-type topics --
entity-name MSKTutorialTopic --add-config 'remote.log.msk.disable.policy=Delete,
 remote.storage.enable=false'
```

> **Note**
>
> 階層型ストレージを無効にすると、階層型ストレージのトピックデータは完全に削除されます。Apache Kafka はプライマリストレージデータを保持しますが、これには、`local.retention.ms` に基づくプライマリ保持ルールが引き続き適用されます。トピックで階層化ストレージを無効にすると、再度有効にすることはできません。既存のトピックで階層型ストレージを無効にする場合は、特定の Apache Kafka クライアントバージョンに制限されません。

AWS CLI を使用して既存の Amazon MSK クラスターで階層型ストレージを有効にする

> **Note**
>
> 圧縮トピックは階層型ストレージではサポートされないため、クラスターの log.cleanup.policy が `delete` に設定されている場合にのみ階層型ストレージを有効にできます。後で、個々のトピックの log.cleanup.policy を `compact` に設定できます (その特定のトピックで階層型ストレージが有効になっていない場合)。サポートされている設定属性の詳細については、「トピックレベルの設定」を参照してください。

1. Kafka バージョンの更新 — クラスターバージョンは単純な整数ではありません。クラスターの最新バージョンを検索するには、 `DescribeCluster`オペレーションまたは `describe-cluster` AWS CLI コマンドを使用します。サンプルのバージョンは `KTVPDKIKX0DER` です。

   ```
   aws kafka update-cluster-kafka-version --cluster-arn ClusterArn --current-version
    Current-Cluster-Version --target-kafka-version 3.6.0
   ```

2. クラスターストレージモードを編集します。次のコード例は、`update-storage` API を使用してクラスターストレージモードを `TIERED` に編集する方法を示しています。

   ```
   aws kafka update-storage --current-version Current-Cluster-Version --cluster-arn
    Cluster-arn --storage-mode TIERED
   ```

コンソールを使用して既存の Amazon MSK クラスターの階層型ストレージを更新する

このプロセスでは、 AWS Management Consoleを使用して階層型ストレージの Amazon MSK クラスターを更新する方法について説明します。

MSK クラスターの現在の Apache Kafka バージョンが 2.8.2.tiered であることを確認します。MSK クラスターを 2.8.2.tiered バージョンにアップグレードする必要がある場合は、「Apache Kafka バージョンの更新」を参照してください。

> **Note**
>
> 圧縮トピックは階層型ストレージではサポートされないため、クラスターの log.cleanup.policy が `delete` に設定されている場合にのみ階層型ストレージを有効にできます。後で、個々のトピックの log.cleanup.policy を `compact` に設定できます (その特定のトピックで階層型ストレージが有効になっていない場合)。サポートされている設定属性の詳細については、「トピックレベルの設定」を参照してください。

1. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
2. クラスターの概要ページに移動し、[プロパティ] を選択します。
3. [ストレージ] セクションに移動し、[クラスターストレージモードを編集] を選択します。
4. [階層化ストレージ と EBS ストレージ] を選択し、[変更を保存] を選択します。

Amazon MSK Standard ブローカーストレージのスケールアップ

ブローカーごとに EBS ストレージの量を増やすことができます。ストレージを減らすことはできません。

ストレージボリュームは、このスケールアップオペレーション中も引き続き使用できます。

> **Important**
>
> MSK クラスター用にストレージをスケーリングすると、追加のストレージがすぐに使用できるようになります。ただし、ストレージのスケーリングイベントが発生するたびに、クラスターにはクールダウン期間が必要です。Amazon MSK は、このクールダウン期間を利用してクラスターを最適化します。その後、クラスターは再びスケーリングできるようになります。この期間は、クラスターのストレージサイズ、使用率、トラフィックに応じて、最小 6 時間から 24 時間以上までさまざまです。これは、自動スケーリングイベント

と、UpdateBrokerStorage オペレーションを使用した手動スケーリングの両方に適用されます。ストレージの適切なサイジングについては、「the section called “標準ブローカーのベストプラクティス”」を参照してください。

階層型ストレージを使用すると、ブローカーのストレージ容量を無制限にスケールアップできます。「標準ブローカーの階層型ストレージ」を参照してください。

トピック

- Amazon MSK クラスターの自動スケーリング
- 標準ブローカーの手動スケーリング

Amazon MSK クラスターの自動スケーリング

使用量の増加に応じてクラスターのストレージを自動的に拡張するには、Amazon MSK のアプリケーション自動スケーリングを設定できます。自動スケーリングポリシーでは、ディスク使用率のターゲットと最大スケーリング容量を設定します。

Amazon MSK の自動スケーリングを使用する前に、次の点を考慮する必要があります。

- **Important**
  ストレージスケーリングアクションは、6 時間ごとに 1 回だけ実行できます。

  まず、ストレージ需要に適したサイズのストレージボリュームから始めることが推奨されます。クラスターの適切なサイズ設定のガイダンスについては、「クラスターの適切なサイズ設定: クラスターあたりの標準ブローカーの数」を参照してください。
- Amazon MSK では、使用量の減少に応じたクラスターストレージの削減は行われません。Amazon MSK では、ストレージボリュームのサイズを減らすことはできません。クラスターストレージの容量を減らす必要がある場合は、既存のクラスターを小さいストレージのクラスターに移行してください。クラスターの移行については、「Amazon MSK クラスターへの移行」を参照してください。
- Amazon MSK は、アジアパシフィック (大阪) およびアフリカ (ケープタウン) リージョンでの自動スケーリングをサポートしていません。
- 自動スケーリングポリシーをクラスターに関連付けると、Amazon EC2 Auto Scaling はターゲット追跡用の Amazon CloudWatch アラームを自動的に作成します。自動スケーリングポリシーのあ

るクラスターを削除しても、この CloudWatch アラームは保持されます。CloudWatch アラームを削除するには、クラスターを削除する前に、クラスターから自動スケーリングポリシーを削除する必要があります。ターゲット追跡の詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon EC2 Auto Scaling のターゲットトラッキングスケーリングポリシー」を参照してください。

トピック

- Amazon MSK の自動スケーリングポリシーの詳細
- Amazon MSK クラスターの自動スケーリングを設定する

Amazon MSK の自動スケーリングポリシーの詳細

自動スケーリングポリシーでは、クラスターに関する次のパラメータを定義します。

- ストレージ使用率ターゲット : Amazon MSK がオートスケーリングオペレーションをトリガーするために参照するストレージ使用率のしきい値。使用率のターゲットは、現在のストレージ容量の 10% ~ 80% に設定できます。ストレージ使用率ターゲットの設定の推奨値は 50% から 60% です。
- 最大ストレージ容量: Amazon MSK でブローカーストレージに対して設定できるスケーリングの上限。最大ストレージ容量は、各ブローカーにつき最大 16 TiB に設定できます。詳細については、「Amazon MSK クォータ」を参照してください。

Amazon MSK は、`Maximum Disk Utilization` メトリクスが `Storage Utilization Target` 設定以上であることを検出すると、10 GiB または現在のストレージの 10% の 2 つの数値のうち大きい方と同量だけストレージ容量を増やします。例えば、現在の容量が 1000 GiB の場合、その量は 100 GiB となります。このサービスは、ストレージ使用率を 1 分ごとにチェックします。さらにスケーリングオペレーションが行われるたびに、10 GiB または現在のストレージの 10% の 2 つの数値のうち大きい方と同量だけストレージが増えていきます。

自動スケーリングオペレーションが発生したかを判別するには、ListClusterOperations オペレーションを使用します。

Amazon MSK クラスターの自動スケーリングを設定する

Amazon MSK コンソール、Amazon MSK API、または AWS CloudFormation を使用して、ストレージの自動スケーリングを実装できます。CloudFormation のサポートは、Application Auto Scaling を通じて利用できます。

> **Note**
>
> クラスターの作成時に自動スケーリングを実装することはできません。最初にクラスターを作成してから、そのクラスターに対して自動スケーリングポリシーを作成して有効にする必要があります。ただし、Amazon MSK サービスによるクラスターの作成時に、ポリシーを作成することはできます。

トピック

- Amazon MSK AWS Management Consoleを使用して自動スケーリングを設定する
- CLI を使用して自動スケーリングを設定する
- API を使用して Amazon MSK の自動スケーリングを設定する

Amazon MSK AWS Management Consoleを使用して自動スケーリングを設定する

このプロセスでは、ストレージの自動スケーリングを実装するための、Amazon MSK コンソールの使用方法について説明します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. クラスターリストから、 お客様のクラスターを選択します。これにより、クラスターの詳細がリストされたページに移動します。
3. [Auto scaling for storage] (ストレージのオートスケーリング) セクションで、[Configure] (設定) を選択します。
4. オートスケーリングポリシーを作成して名前を付けます。ストレージ使用率のターゲット、最大ストレージ容量、およびターゲットメトリクスを指定します。
5. Save changes を選択してください。

新しいポリシーを保存して有効にすると、ポリシーがクラスターに対してアクティブになります。Amazon MSK は、ストレージ使用率が設定した値に達すると、クラスターのストレージを拡張します。

CLI を使用して自動スケーリングを設定する

このプロセスでは、ストレージの自動スケーリングを実装するための、Amazon MSK CLI の使用方法について説明します。

1. [RegisterScalableTarget] コマンドを使用して、ストレージ使用率ターゲットを登録します。
2. [PutScalingPolicy] コマンドを使用して、自動拡張ポリシーを作成します。

API を使用して Amazon MSK の自動スケーリングを設定する

このプロセスでは、ストレージの自動スケーリングを実装するために、Amazon MSK API の使用方法について説明します。

1. [RegisterScalableTarget] API を使用して、ストレージ使用率ターゲットを登録する。
2. PutScalingPolicy API を使用して、自動拡張ポリシーを作成する。

標準ブローカーの手動スケーリング

ストレージを増やすには、クラスターが ACTIVE 状態になるのを待ちます。ストレージのスケーリングのクールダウン期間は、各イベント間で最低で 6 時間です。このオペレーションによって追加のストレージがすぐに使用可能になりますが、クラスターで最適化を実行するため、この操作に最大 24 時間以上かかることがあります。これらの最適化の所要時間は、ストレージの容量に比例します。

を使用したブローカーストレージのスケールアップ AWS Management Console

1. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
2. ブローカーストレージをアップデートしたい MSK クラスターを選択します。
3. [Storage] (ストレージ) セクションで、[Edit] (編集) を選択します。
4. 必要なストレージボリュームを指定します。ストレージの量を増やすだけで、減らすことはできません。
5. [Save changes] (変更の保存) をクリックします。

を使用したブローカーストレージのスケールアップ AWS CLI

*ClusterArn*をクラスターの作成時に取得したAmazon リソースネーム (ARN) に置き換えて、次のコマンドを実行します。クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。詳細については、「the section called “クラスターを一覧表示する”」を参照してください。

Current-Cluster-Version を、クラスターの現在のバージョンに置き換えます。

> **Important**
>
> クラスターのバージョンは単純な整数ではありません。クラスターの最新バージョンを検索するには、DescribeCluster オペレーションまたは describe-cluster AWS CLI コマンドを使用します。サンプルのバージョンは KTVPDKIKX0DER です。

Target-Volume-in GiB パラメータは、各ブローカーが持つストレージの量を表します。すべてのブローカーのストレージのみ、更新できます。ストレージを更新するブローカーを個別に指定することはできません。*Target-Volume-in-GiB* に指定する値は、100 GiB を超える整数である必要があります。更新操作後のブローカーごとのストレージは 16384 GiB を超えることはできません。

```
aws kafka update-broker-storage --cluster-arn ClusterArn --current-version Current-
Cluster-Version --target-broker-ebs-volume-info '{"KafkaBrokerNodeId": "All",
 "VolumeSizeGB": Target-Volume-in-GiB}'
```

API を使用したブローカーストレージのスケールアップ

API を使用してブローカーストレージを更新するには、「UpdateBrokerStorage」を参照してください。

Amazon MSK クラスター内の標準ブローカーのストレージスループットを管理する

Amazon MSK コンソール、CLI、API を使用してスループットをプロビジョニングする方法については、「」を参照してください???。

トピック

- Amazon MSK ブローカースループットのボトルネックと最大スループット設定
- Amazon MSK クラスターのストレージスループットを測定する
- Amazon MSK クラスター内のプロビジョンドストレージの設定更新値
- Amazon MSK クラスター内の標準ブローカーのストレージスループットをプロビジョニングする

Amazon MSK ブローカースループットのボトルネックと最大スループット設定

ブローカースループットのボトルネックには、ボリュームスループット、Amazon EC2 から Amazon EBS へのネットワークスループット、Amazon EC2 のエグレススループットなど、複数の原因があります。プロビジョンドストレージのスループットを有効にして、ボリュームスループット

を調整できます。ただし、ブローカースループットの制限は、Amazon EC2 から Amazon EBS へのネットワークスループットと Amazon EC2 のエグレススループットによって発生する可能性があります。

Amazon EC2 のエグレススループットは、コンシューマーグループの数と、コンシューマーグループあたりのコンシューマーの数の影響を受けます。また、Amazon EC2 から Amazon EBS へのネットワークスループットと Amazon EC2 のエグレススループットはどちらも、ブローカーサイズが大きいほど高くなります。

ボリュームサイズが 10 GiB 以上の場合は、250 MiB/秒以上のストレージスループットをプロビジョニングできます。デフォルトは 250 MiB/秒です。ストレージスループットをプロビジョニングするには、ブローカーサイズ kkafka.m5.4xlarge 以上 (または kafka.m7g.2xlarge 以上) を選択する必要があります。また、次の表に示すように最大スループットを指定できます。

ブローカーサイズ	ストレージの最大スループット (MiB/秒)
kafka.m5.4xlarge	593
kafka.m5.8xlarge	850
kafka.m5.12xlarge	1,000
kafka.m5.16xlarge	1,000
kafka.m5.24xlarge	1,000
kafka.m7g.2xlarge	312.5
kafka.m7g.4xlarge	625
kafka.m7g.8xlarge	1,000
kafka.m7g.12xlarge	1,000
kafka.m7g.16xlarge	1,000

Amazon MSK クラスターのストレージスループットを測定する

`VolumeReadBytes` および `VolumeWriteBytes` メトリクスを使用して、クラスターの平均ストレージスループットを測定できます。これら 2 つのメトリクスを合計は、ストレージの平均スルー

プット (バイト単位) を示します。クラスターのストレージの平均スループットを取得するには、これら 2 つのメトリクスを SUM に、期間を 1 分に設定し、次の式を使用します。

```
Average storage throughput in MiB/s = (Sum(VolumeReadBytes) + Sum(VolumeWriteBytes)) /
 (60 * 1024 * 1024)
```

VolumeReadBytes および VolumeWriteBytes メトリクスについては、「the section called "PER_BROKER レベルモニタリング"」を参照してください。

Amazon MSK クラスター内のプロビジョンドストレージの設定更新値

Amazon MSK 設定は、プロビジョンドスループットを有効にする前でも後でも更新できます。ただし、num.replica.fetchers 設定パラメータの更新と、プロビジョンドスループットの有効化の両方のアクションを実行するまでは、目的のスループットは表示されません。

Amazon MSK のデフォルト設定では、num.replica.fetchers の値は 2 です。num.replica.fetchers を更新する際には、次の表の推奨値を使用できます。これらの値は参考用です。これらの値は、ユースケースに基づいて調整することが推奨されます。

ブローカーサイズ	num.replica.fetchers
kafka.m5.4xlarge	4
kafka.m5.8xlarge	8
kafka.m5.12xlarge	14
kafka.m5.16xlarge	16
kafka.m5.24xlarge	16

更新した設定は最大 24 時間有効にならない場合があり、ソースボリュームが十分に利用されていない場合はさらに時間がかかる場合があります。ただし、移行ボリュームのパフォーマンスは、移行期間中は少なくともソースストレージボリュームのパフォーマンスと同等です。フルに利用されている 1 TiB ボリュームの場合、更新された設定に移行するには、通常は約 6 時間かかります。

Amazon MSK クラスター内の標準ブローカーのストレージスループットをプロビジョニングする

Amazon MSK ブローカーは、データをストレージボリュームに保持します。ストレージ I/O は、プロデューサーがクラスターに書き込むとき、ブローカー間でデータがレプリケートされるとき、お

よびコンシューマーがメモリ内にないデータを読み取るときに消費されます。ボリュームストレージのスループットは、ストレージボリュームにデータを書き込んだり、ストレージボリュームからデータを読み取ったりできる速度です。プロビジョンドストレージのスループットは、クラスター内のブローカーに対してその速度を指定する機能です。

ブローカーのサイズが `kafka.m5.4xlarge` 以上で、ストレージボリュームが 10 GiB 以上のクラスターに対して、プロビジョンドスループットの速度 (MiB/秒) を指定できます。プロビジョンドスループットは、クラスター作成時に指定できます。また、`ACTIVE` 状態のクラスターのプロビジョンドスループットを有効または無効にすることもできます。

スループットの管理については、「」を参照してください???。

トピック

- を使用して Amazon MSK クラスターストレージスループットをプロビジョニングする AWS Management Console
- を使用して Amazon MSK クラスターストレージスループットをプロビジョニングする AWS CLI
- API を使用して Amazon MSK クラスターを作成する際にストレージスループットをプロビジョニングする

を使用して Amazon MSK クラスターストレージスループットをプロビジョニングする AWS Management Console

このプロセスでは、 を使用して、プロビジョニングされたスループットを有効にした Amazon MSK クラスター AWS Management Console を作成する方法の例を示します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. Create cluster (クラスターの作成) を選択します。
3. Custom create (カスタム作成) を選択します。
4. クラスターの名前を指定します。
5. [ストレージ] セクションで、[有効化] を選択します。
6. ブローカーあたりのストレージスループットの値を選択します。
7. VPC、ゾーンとサブネット、セキュリティグループを選択します。
8. [次へ] を選択します。
9. [セキュリティ] ステップの一番下で [次へ] を選択します。

10. [モニタリングおよびタグ] ステップの一番下で [次へ] を選択します。
11. クラスター設定を確認し、[クラスターの作成] を選択します。

を使用して Amazon MSK クラスターストレージスループットをプロビジョニングする AWS CLI

このプロセスでは、 を使用して、プロビジョニングされたスループットが有効になっているクラスター AWS CLI を作成する方法の例を示します。

1. 次の JSON をコピーして、ファイルに貼り付けます。サブネット ID とセキュリティグループ ID のプレースホルダーは、自分のアカウントの値に置き換えてください。ファイルに `cluster-creation.json` という名前を付け、保存します。

```
{
    "Provisioned": {
        "BrokerNodeGroupInfo":{
            "InstanceType":"kafka.m5.4xlarge",
            "ClientSubnets":[
                "Subnet-1-ID",
                "Subnet-2-ID"
            ],
            "SecurityGroups":[
                "Security-Group-ID"
            ],
            "StorageInfo": {
                "EbsStorageInfo": {
                    "VolumeSize": 10,
                    "ProvisionedThroughput": {
                        "Enabled": true,
                        "VolumeThroughput": 250
                    }
                }
            }
        },
        "EncryptionInfo": {
            "EncryptionInTransit": {
                "InCluster": false,
                "ClientBroker": "PLAINTEXT"
            }
        },
        "KafkaVersion":"2.8.1",
        "NumberOfBrokerNodes": 2
    },
```

```
    "ClusterName": "provisioned-throughput-example"
}
```

2. 前のステップで JSON ファイルを保存したディレクトリから次の AWS CLI コマンドを実行します。

```
aws kafka create-cluster-v2 --cli-input-json file://cluster-creation.json
```

API を使用して Amazon MSK クラスターを作成する際にストレージスループットをプロビジョニングする

クラスターの作成時にプロビジョンドストレージのスループットを設定するには、CreateClusterV2 を使用します。

Amazon MSK のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS お客様とお客様の間の責任共有です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – AWS クラウドで AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。 AWS また、 では、安全に使用できるサービスも提供しています。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。Amazon Managed Streaming for Apache Kafka に適用されるコンプライアンス プログラムについては、コンプライアンスプログラムによる対象範囲内の Amazon Web Servicesを参照してください。
- クラウドのセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、お客様のデータの機密性、企業の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、Amazon MSK を使用するときに責任共有モデルを適用する方法を理解するのに役立ちます。次のトピックでは、セキュリティとコンプライアンスの目標を達成するために Amazon MSK を設定する方法を示します。また、Amazon MSK リソースのモニタリングと保護に役立つ他の Amazon Web Services の使用方法についても学びます。

トピック

- Amazon Managed Streaming for Apache Kafka におけるデータ保護
- Amazon MSK API の認証と認可
- Apache Kafka API の認証と認可
- Amazon MSK クラスターのセキュリティグループの変更
- Amazon MSK クラスター内の Apache ZooKeeper ノードへのアクセスを制御する
- Amazon Managed Streaming for Apache Kafka のコンプライアンス検証
- Amazon Managed Streaming for Apache Kafka の復元力
- Amazon Managed Streaming for Apache Kafka におけるインフラストラクチャセキュリティ

Amazon Managed Streaming for Apache Kafka におけるデータ保護

責任 AWS 共有モデル、Amazon Managed Streaming for Apache Kafka のデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された AWS 責任共有モデルおよび GDPR のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。

- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して Amazon MSK AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

トピック

- Amazon MSK 暗号化
- Amazon MSK 暗号化の使用を開始する
- インターフェイス VPC エンドポイントで Amazon MSK APIs を使用する

Amazon MSK 暗号化

Amazon MSK には、厳格なデータ管理要件を満たすために使用できるデータ暗号化オプションが用意されています。Amazon MSK が暗号化に使用する証明書は、13 か月ごとに更新する必要があります。Amazon MSK は、すべてのクラスターに対してこれらの証明書を自動的に更新します。Amazon MSK が証明書の更新オペレーションを開始すると、Express ブローカークラスターは `ACTIVE`状態のままになります。標準ブローカークラスターの場合、Amazon MSK は証明書の更新オペレーションを開始する`MAINTENANCE`ときにクラスターの状態を に設定します。更新が完了すると、MSK はこれを `ACTIVE` に戻します。クラスターが証明書の更新オペレーションにある間は、引き続きデータを生成して使用できますが、更新オペレーションを実行することはできません。

保管中の Amazon MSK 暗号化

Amazon MSK はAWS Key Management Service (KMS) と統合して、透過的なサーバー側暗号化を提供します。Amazon MSK は、保管中のデータを常に暗号化します。MSK クラスターを作成するときに、Amazon MSK が保管中のデータの暗号化に使用する AWS KMS key を指定できます。KMS キーを指定しない場合、Amazon MSK がユーザーの代わりに AWS マネージドキーを作成し、それを使用します。KMS キーの詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS KMS keys」を参照してください。

転送中の Amazon MSK 暗号化

Amazon MSKは TLS 1.2 を使用します。デフォルトでは、MSK クラスターのブローカー間で転送中のデータを暗号化します。このデフォルトは、クラスターの作成時に上書きできます。

クライアントとブローカー間の通信では、次の 3 つの設定のいずれかを指定する必要があります。

- TLS 暗号化データのみを許可します。これはデフォルトの設定です。
- プレーンテキストと TLS 暗号化データの両方を許可します。
- プレーンテキストのデータのみを許可します。

Amazon MSK ブローカーはパブリック AWS Certificate Manager 証明書を使用します。したがって、Amazon Trust Services を信頼するトラストストアは、Amazon MSK ブローカーの証明書も信頼します。

転送中の暗号化を有効にすることを強くお勧めしますが、CPU オーバーヘッドが増加し、数ミリ秒のレイテンシーが発生する可能性があります。ただし、ほとんどのユースケースはこれらの違いに敏感ではなく、影響の大きさは、クラスター、クライアント、および使用プロファイルの構成によって異なります。

Amazon MSK 暗号化の使用を開始する

MSK クラスターを作成するときに、JSON 形式で暗号化設定を指定できます。以下に例を示します。

```
{
   "EncryptionAtRest": {
       "DataVolumeKMSKeyId": "arn:aws:kms:us-east-1:123456789012:key/abcdabcd-1234-
abcd-1234-abcd123e8e8e"
    },
   "EncryptionInTransit": {
        "InCluster": true,
        "ClientBroker": "TLS"
    }
}
```

DataVolumeKMSKeyId では、カスタマーマネージドキーか、またはアカウント内の MSK 用の AWS マネージドキー (alias/aws/kafka) を指定できます。を指定しない場合EncryptionAtRestでも、Amazon MSK は で保管中のデータを暗号化します AWS マ

ネージドキー。クラスターが使用しているキーを判別するには、GET リクエストを送信するか DescribeCluster API オペレーションを呼び出します。

EncryptionInTransit の場合、InCluster のデフォルト値は true ですが、Amazon MSK がブローカー間を通過するときにデータを暗号化しないようにする場合は、false に設定できます。

クライアントとブローカー間で転送されるデータの暗号化モードを指定するには、ClientBroker を TLS、TLS_PLAINTEXT、または PLAINTEXT のいずれかに設定します。

トピック

- Amazon MSK クラスターの作成時に暗号化設定を指定する
- Amazon MSK TLS 暗号化をテストする

Amazon MSK クラスターの作成時に暗号化設定を指定する

このプロセスでは、Amazon MSK クラスターの作成時に暗号化設定を指定する方法について説明します。

クラスターの作成時に暗号化設定を指定する

1. 前述の例の内容をファイルに保存し、任意の名前を付けます。たとえば、encryption-settings.json と呼びます。
2. create-cluster コマンドを実行し、encryption-info オプションを使用して、設定 JSON ファイルを保存したファイルを指定します。以下に例を示します。*{YOUR MSK VERSION}* は、Apache Kafka クライアントバージョンと一致するバージョンに置き換えてください。MSK クラスターバージョンの確認方法については、「MSK クラスターのバージョンの確認」を参照してください。MSK クラスターバージョンと異なる Apache Kafka クライアントバージョンを使用すると、Apache Kafka データの破損、損失、ダウンタイムが発生する可能性があることに注意してください。

```
aws kafka create-cluster --cluster-name "ExampleClusterName" --broker-node-group-
info file://brokernodegroupinfo.json --encryption-info file://encryptioninfo.json
 --kafka-version "{YOUR MSK VERSION}" --number-of-broker-nodes 3
```

次に、このコマンドを実行した後の正常な応答の例を示します。

```
{
    "ClusterArn": "arn:aws:kafka:us-east-1:123456789012:cluster/SecondTLSTest/
abcdabcd-1234-abcd-1234-abcd123e8e8e",
```

```
    "ClusterName": "ExampleClusterName",
    "State": "CREATING"
}
```

Amazon MSK TLS 暗号化をテストする

このプロセスでは、Amazon MSK で TLS 暗号化をテストする方法について説明します。

TLS 暗号化をテストするには

1. the section called “クライアントマシンを作成する” のガイダンスに従って、クライアントマシンを作成します。
2. クライアントマシンに Apache Kafka をインストールします。
3. この例では、JVM トラストストアを使用して MSK クラスターと通信します。これを行うには、まずクライアントマシンに `/tmp` という名前のフォルダを作成します。次に、Apache Kafka インストールの `bin` フォルダに移動し、次のコマンドを実行します。(JVM パスは異なる場合があります)。

   ```
   cp /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.201.b09-0.amzn2.x86_64/jre/lib/security/
   cacerts /tmp/kafka.client.truststore.jks
   ```

4. クライアントマシン上の Apache Kafka インストールの `bin` フォルダに、次の内容の `client.properties` というテキストファイルを作成します。

   ```
   security.protocol=SSL
   ssl.truststore.location=/tmp/kafka.client.truststore.jks
   ```

5. AWS CLI がインストールされているマシンで次のコマンドを実行し、*`clusterARN`* をクラスターの ARN に置き換えます。

   ```
   aws kafka get-bootstrap-brokers --cluster-arn clusterARN
   ```

   正常に実行された場合の結果は次のようになります。次のステップで必要になるため、この結果を保存します。

   ```
   {
       "BootstrapBrokerStringTls": "a-1.example.g7oein.c2.kafka.us-
   east-1.amazonaws.com:0123,a-3.example.g7oein.c2.kafka.us-
   east-1.amazonaws.com:0123,a-2.example.g7oein.c2.kafka.us-east-1.amazonaws.com:0123"
   ```

```
}
```

6. 次のコマンドを実行します。その際、*BootstrapBrokerStringTls* は、前のステップで取得したブローカーエンドポイントの 1 つに置き換えてください。

```
<path-to-your-kafka-installation>/bin/kafka-console-producer.sh --broker-
list BootstrapBrokerStringTls --producer.config client.properties --topic
 TLSTestTopic
```

7. 新しいコマンドウィンドウを開き、同じクライアントマシンに接続します。その後、次のコマンドを実行して、コンソールコンシューマーを作成します。

```
<path-to-your-kafka-installation>/bin/kafka-console-consumer.sh --bootstrap-
server BootstrapBrokerStringTls --consumer.config client.properties --topic
 TLSTestTopic
```

8. プロデューサーウィンドウで、テキストメッセージに続けてリターンを入力し、コンシューマーウィンドウで同じメッセージを探します。Amazon MSK は、このメッセージを転送中に暗号化しました。

暗号化されたデータを操作するように Apache Kafka クライアントを設定する方法の詳細については、「Kafka クライアントの設定」を参照してください。

インターフェイス VPC エンドポイントで Amazon MSK APIs を使用する

AWS PrivateLink を搭載したインターフェイス VPC エンドポイントを使用すると、Amazon VPC と Amazon MSK APIs 間のトラフィックが Amazon ネットワークから出るのを防ぐことができます。インターフェイス VPC エンドポイントには、インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続は必要ありません。AWS PrivateLink は、Amazon VPC 内のプライベート IPs を持つ Elastic Network Interface を使用して、 AWS サービス間のプライベート通信を可能にする AWS テクノロジーです。詳細については、Amazon Virtual Private Cloud and Interface VPC Endpoints (AWS PrivateLink)」を参照してください。

アプリケーションは、 AWS PrivateLink を使用して Amazon MSK プロビジョンド API と MSK Connect APIs に接続できます。開始するには、Amazon MSK API のインターフェイス VPC エンドポイントを作成して、インターフェイス VPC エンドポイントを介して Amazon VPC リソースとの間で流れるトラフィックを開始します。FIPS 対応のインターフェイス VPC エンドポイントは、米国リージョンで使用できます。詳細については、「インターフェイスエンドポイントの作成」を参照してください。

この機能を使用すると、Apache Kafka クライアントは接続文字列を動的に取得して、インターネットを経由せずに MSK プロビジョンドリソースまたは MSK Connect リソースに接続できます。

インターフェイス VPC エンドポイントを作成するときは、次のいずれかのサービス名エンドポイントを選択します。

MSK プロビジョニングの場合：

- com.amazonaws.region.kafka
- com.amazonaws.region.kafka-fips (FIPS 対応)

region はリージョン名です。MSK プロビジョンド互換 APIs を使用するには、このサービス名を選択します。詳細については、https://docs.aws.amazon.com/msk/1.0/apireference/ の「オペレーション」を参照してください。

MSK Connect の場合：

- com.amazonaws.region.kafkaconnect

region はリージョン名です。MSK Connect 互換 APIs。詳細については、「Amazon MSK Connect API リファレンス」の「アクション」を参照してください。

インターフェイス VPC エンドポイントを作成するstep-by-stepなどの詳細については、「 AWS PrivateLink ガイド」の「インターフェイスエンドポイントの作成」を参照してください。

Amazon MSK プロビジョンド API または MSK Connect APIs

VPC エンドポイントポリシーは、VPC エンドポイントにポリシーをアタッチするか、IAM ユーザー、グループ、またはロールにアタッチされたポリシーの追加フィールドを使用して、アクセスが指定された VPC エンドポイント経由のみで行われるように制限することで、アクセスを制御することを可能にします。適切なポリシー例を使用して、MSK プロビジョンドサービスまたは MSK Connect サービスのアクセス許可を定義します。

エンドポイントの作成時にポリシーをアタッチしない場合、サービスへのフルアクセスを許可するデフォルトのポリシーが Amazon VPC によって自動的にアタッチされます。エンドポイントポリシーは、IAM アイデンティティベースのポリシーやサービス固有のポリシーを上書きしたり置き換えたりするものではありません。これは、評価項目から指定されたサービスへのアクセスを制御するための別のポリシーです。

詳細については、「 AWS PrivateLink ガイド」の「VPC エンドポイントを使用したサービスへのアクセスの制御」を参照してください。

MSK Provisioned — VPC policy example

読み取り専用アクセス

このサンプルポリシーは、VPC エンドポイントにアタッチできます。詳細については、Amazon VPC のリソースに対するアクセスの制御を参照してください。これにより、アクションは、それがアタッチされている VPC エンドポイントを介したオペレーションの一覧表示と説明のみに制限されます。

```
{
  "Statement": [
    {
      "Sid": "MSKReadOnly",
      "Principal": "*",
      "Action": [
        "kafka:List*",
        "kafka:Describe*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

MSK プロビジョニング済み — VPC エンドポイントポリシーの例

特定の MSK クラスターへのアクセスを制限する

このサンプルポリシーは、VPC エンドポイントにアタッチできます。これにより、アタッチ先の VPC エンドポイントを介して特定の Kafka クラスターへのアクセスが制限されます。

```
{
  "Statement": [
    {
      "Sid": "AccessToSpecificCluster",
      "Principal": "*",
      "Action": "kafka:*",
      "Effect": "Allow",
      "Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/MyCluster"
```

```
    }
  ]
}
```

MSK Connect — VPC endpoint policy example

コネクタを一覧表示して新しいコネクタを作成する

以下は、MSK Connect のエンドポイントポリシーの例です。このポリシーは、指定されたロールがコネクタを一覧表示し、新しいコネクタを作成することを許可します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "MSKConnectPermissions",
            "Effect": "Allow",
            "Action": [
                "kafkaconnect:ListConnectors",
                "kafkaconnect:CreateConnector"
            ],
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:role/<ExampleRole>"
                ]
            }
        }
    ]
}
```

MSK Connect — VPC エンドポイントポリシーの例

指定された VPC 内の特定の IP アドレスからのリクエストのみを許可する

次の例は、指定した VPC 内の指定した IP アドレスから送信されたリクエストのみが成功するように許可するポリシーを示しています。他の IP アドレスからのリクエストは失敗します。

```
{
```

```
    "Statement": [
        {
            "Action": "kafkaconnect:*",
            "Effect": "Allow",
            "Principal": "*",
            "Resource": "*",
            "Condition": {
                "IpAddress": {
                    "aws:VpcSourceIp": "192.0.2.123"
                },
        "StringEquals": {
                    "aws:SourceVpc": "vpc-555555555555"
                }
            }
        }
    ]
}
```

Amazon MSK API の認証と認可

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、Amazon MSK リソースを使用するための認証 (サインイン) および承認 (許可を持つ) できるユーザーを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

- Amazon MSK と IAM の連携の仕組み
- Amazon MSK の ID ベースのポリシーの例
- Amazon MSK のサービスリンクロール
- AWS Amazon MSK の マネージドポリシー
- Amazon MSK の ID およびアクセスのトラブルシューティング

Amazon MSK と IAM の連携の仕組み

IAM を使用して Amazon MSK へのアクセスを管理する前に、Amazon MSK で使用できる IAM の機能を理解する必要があります。Amazon MSK およびその他の AWS のサービスが IAM と連携する方法の概要については、IAM ユーザーガイドのAWS 「IAM と連携する のサービス」を参照してください。

トピック

- Amazon MSK の ID ベースのポリシー
- Amazon MSK のリソースベースのポリシー
- Amazon MSK タグに基づく認可
- Amazon MSK IAM ロール

Amazon MSK の ID ベースのポリシー

IAM アイデンティティベースポリシーでは、許可または拒否するアクションとリソース、またアクションを許可または拒否する条件を指定できます。Amazon MSK は、特定のアクション、リソース、および条件キーをサポートします。JSON ポリシーで使用するすべての要素については、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

Amazon MSK の ID ベースのポリシーのアクション

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは依存アクションと呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための許可を付与するポリシーで使用されます。

Amazon MSK のポリシーアクションは、アクションの前に次のプレフィックスを使用します : `kafka:`。例えば、Amazon MSK `DescribeCluster` API オペレーションを使用して MSK クラスターを記述する許可を誰かに付与するには、ポリシーに `kafka:DescribeCluster` アクションを含めます。ポリシーステートメントには`Action` または `NotAction` 要素を含める必要があります。Amazon MSK は、このサービスで実行できるタスクを記述する独自の一連のアクションを定義します。

1 つのステートメントで複数のアクションを指定するには、次のようにコンマで区切ります。

```
"Action": ["kafka:action1", "kafka:action2"]
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、Describe という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "kafka:Describe*"
```

Amazon MSK アクションのリストを表示するには、「IAM ユーザーガイド」の「Amazon Managed Streaming for Apache Kafka のアクション、リソース、および条件キー」を参照してください。

Amazon MSK ID ベースのポリシーのリソース

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ステートメントにはResource または NotResource 要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの許可をサポートしないアクションの場合は、ワイルドカード (*) を使用して、ステートメントがすべてのリソースに適用されることを示します。

```
"Resource": "*"
```

Amazon MSK インスタンスリソースには次の ARN があります。

```
arn:${Partition}:kafka:${Region}:${Account}:cluster/${ClusterName}/${UUID}
```

ARN の形式の詳細については、「Amazon リソースネーム (ARNs AWS 「サービス名前空間」を参照してください。

例えば、ステートメントで CustomerMessages インスタンスを指定するには、次の ARN を使用します。

```
"Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/CustomerMessages/abcd1234-
abcd-dcba-4321-a1b2abcd9f9f-2"
```

特定のアカウントに属するすべてのインスタンスを指定するには、ワイルドカード (*) を使用します。

```
"Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/*"
```

リソースを作成するためのアクションなど、一部の Amazon MSK アクションは、特定のリソースで実行できません。このような場合は、ワイルドカード (*) を使用する必要があります。

```
"Resource": "*"
```

複数リソースを単一ステートメントで指定するには、ARN をカンマで区切ります。

```
"Resource": ["resource1", "resource2"]
```

Amazon MSK リソースタイプとその ARN のリストを表示するには、「IAM ユーザーガイド」の「Amazon Managed Streaming for Apache Kafka によって定義されたリソース」を参照してください。各リソースの ARN を指定できるアクションについては、「Amazon Managed Streaming for Apache Kafka によって定義されるアクション」を参照してください。

Amazon MSK ID ベースのポリシーの条件キー

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の Condition 要素を指定する場合、または 1 つの Condition 要素に複数のキーを指定する場合、 AWS では AND 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理ORオペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

Amazon MSK は、独自の条件キーのセットを定義し、いくつかのグローバル条件キーの使用もサポートしています。すべての AWS グローバル条件キーを確認するには、IAM ユーザーガイドのAWS 「グローバル条件コンテキストキー」を参照してください。

Amazon MSK 条件キーのリストを表示するには、IAM ユーザーガイドのAmazon Managed Streaming for Apache Kafka の条件キーを参照してください。条件キーを使用できるアクションとリソースについては、Amazon Managed Streaming for Apache Kafka によって定義されたアクションを参照してください。

Amazon MSK ID ベースのポリシーの例

Amazon MSK アイデンティティベースのポリシーの例を表示するには、Amazon MSK の ID ベースのポリシーの例 を参照してください。

Amazon MSK のリソースベースのポリシー

Amazon MSK は、Amazon MSK クラスターで使用するクラスターポリシー (リソースベースのポリシーとも呼ばれます) をサポートしています。クラスターポリシーを使用して、Amazon MSK クラスターへのプライベート接続を設定するためのクロスアカウントのアクセス許可を付与する IAM プリンシパルを定義できます。IAM クライアント認証と併用すると、クラスターポリシーを使用して、接続クライアントの Kafka データプレーンのアクセス許可を細かく定義することもできます。

クラスターポリシーの設定方法の例については、「ステップ 2: クラスターポリシーを MSK クラスターにアタッチする」を参照してください。

Amazon MSK タグに基づく認可

Amazon MSK クラスターにタグをアタッチすることができます。タグに基づいてアクセスを制御するには`kafka:ResourceTag/`*`key-name`*、`aws:RequestTag/`*`key-name`*、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。Amazon MSK リソースのタグ付けについては、「」を参照してくださいthe section called “クラスターにタグを付ける”。

クラスターへのアクセスは、タグを使用してのみ制御できます。トピックとコンシューマーグループにタグを付けるには、タグなしでポリシーに別のステートメントを追加する必要があります。

そのクラスターのタグに基づいてクラスターへのアクセスを制限するためのアイデンティティベースのポリシーの例を表示するには、「」を参照してくださいタグに基づく Amazon MSK クラスターへのアクセス。

アイデンティティベースのポリシーの条件を使用して、タグに基づいて Amazon MSK リソースへのアクセスを制御できます。次の例は、クラスターの説明、ブートストラップブローカーの取得、ブローカーノードのリスト、更新、削除をユーザーに許可するポリシーを示しています。ただし、このポリシーは、クラスタータグにそのユーザーの の値Ownerがある場合にのみアクセス許可を付与しますusername。次のポリシーの 2 番目のステートメントは、クラスター上のトピックへのアクセスを許可します。このポリシーの最初のステートメントでは、トピックへのアクセスは許可されません。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AccessClusterIfOwner",
      "Effect": "Allow",
      "Action": [
        "kafka:Describe*",
        "kafka:Get*",
        "kafka:List*",
        "kafka:Update*",
        "kafka:Delete*"
      ],
      "Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Owner": "${aws:username}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kafka-cluster:*Topic*",
        "kafka-cluster:WriteData",
        "kafka-cluster:ReadData"
      ],
      "Resource": [
        "arn:aws:kafka:us-east-1:123456789012:topic/*"
      ]
    }
  ]
```

```
}
```

Amazon MSK IAM ロール

IAM ロールは、特定のアクセス許可を持つ、Amazon Web Services アカウント内のエンティティです。

Amazon MSK での一時的な認証情報の使用

一時的な認証情報を使用して、フェデレーションでサインインする、IAM 役割を引き受ける、またはクロスアカウント役割を引き受けることができます。一時的なセキュリティ認証情報を取得するには、AssumeRole や GetFederationToken などの AWS STS API オペレーションを呼び出します。

Amazon MSK は、一時的な認証情報の使用をサポートしています。

サービスにリンクされた役割

サービスにリンクされたロールにより、Amazon Web Services は他のサービスのリソースにアクセスして、ユーザーに代わってアクションを完了することができます。サービスリンクロールは IAM アカウント内に表示され、サービスによって所有されます。 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

Amazon MSK は、サービスにリンクされたロールをサポートします。Amazon MSK サービスにリンクされたロールの作成または管理の詳細については、「the section called “サービスにリンクされた役割”」 。

Amazon MSK の ID ベースのポリシーの例

デフォルトでは、IAM ユーザーとロールには Amazon MSK API アクションを実行する権限がありません。IAM 管理者は、指定されたリソースで特定の API 操作を実行するための許可をユーザーとロールに付与する IAM ポリシーを作成する必要があります。続いて、管理者はそれらの権限が必要な IAM ユーザーまたはグループにそのポリシーをアタッチする必要があります。

JSON ポリシードキュメントのこれらの例を使用して、IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「JSON タブでのポリシーの作成」を参照してください。

トピック

- ポリシーに関するベストプラクティス

- ユーザーが自分の権限を表示できるようにする
- 1 つの Amazon MSK クラスターへのアクセス
- タグに基づく Amazon MSK クラスターへのアクセス

ポリシーに関するベストプラクティス

ID ベースのポリシーは、ユーザーのアカウント内で誰かが Amazon MSK リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行 – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」または「ジョブ機能の AWS マネージドポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーとアクセス許可」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます AWS CloudFormation。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素:条件」を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer でポリシーを検証する」を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーション

が呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA を使用した安全な API アクセス」を参照してください。

IAM でのベストプラクティスの詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティベストプラクティス」を参照してください。

ユーザーが自分の権限を表示できるようにする

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

1 つの Amazon MSK クラスターへのアクセス

この例では、Amazon Web Services アカウントの IAM ユーザーに、クラスターの 1 つである purchaseQueriesCluster へのアクセスを許可します。このポリシーにより、ユーザーはクラスターを記述し、ブートストラップブローカーを取得し、ブローカーノードを一覧表示し、更新することができます。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"UpdateCluster",
         "Effect":"Allow",
         "Action":[
            "kafka:Describe*",
            "kafka:Get*",
            "kafka:List*",
            "kafka:Update*"
         ],
         "Resource":"arn:aws:kafka:us-east-1:012345678012:cluster/
purchaseQueriesCluster/abcdefab-1234-abcd-5678-cdef0123ab01-2"
      }
   ]
}
```

タグに基づく Amazon MSK クラスターへのアクセス

アイデンティティベースのポリシーの条件を使用して、タグに基づいて Amazon MSK リソースへのアクセスを制御できます。この例では、クラスターの記述、クラスターのブートストラップブローカーの取得、ブローカーノードの一覧表示、更新、削除をユーザーに許可するポリシーの作成方法を示しています。ただし、クラスター タグ Owner にそのユーザーのユーザー名の値がある場合のみ、アクセス許可は付与されます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AccessClusterIfOwner",
      "Effect": "Allow",
      "Action": [
        "kafka:Describe*",
        "kafka:Get*",
        "kafka:List*",
        "kafka:Update*",
        "kafka:Delete*"
      ],
      "Resource": "arn:aws:kafka:us-east-1:012345678012:cluster/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Owner": "${aws:username}"
        }
      }
    }
  ]
}
```

このポリシーをアカウントの IAM ユーザーにアタッチできます。richard-roe という名前のユーザーが MSK クラスターを更新しようとする場合、クラスターには Owner=richard-roe または owner=richard-roe のタグを付ける必要があります。それ以外の場合、アクセスは拒否されます。条件キー名では大文字と小文字は区別されないため、条件タグキー Owner は Owner と owner に一致します。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素: 条件」を参照してください。

Amazon MSK のサービスリンクロール

Amazon MSK は AWS Identity and Access Management (IAM) サービスにリンクされたロールを使用します。サービスにリンクされたロールは、Amazon MSK に直接リンクされている一意のタイプの IAM ロールです。サービスにリンクされたロールは Amazon MSK によって事前定義されており、サービスがユーザーに代わって他の AWS サービスを呼び出すために必要なすべてのアクセス許可が含まれています。

サービスにリンクされたロールを使用すると、必要な設定を手動で追加する必要がないため、Amazon MSK の設定が簡単になります。Amazon MSK は、サービスにリンクされたロールの権限を定義します。特に明記されていない限り、Amazon MSK のみがそのロールを引き受けることができます。定義された権限には、信頼ポリシーとアクセス許可ポリシーが含まれ、そのアクセス許可ポリシーを他の IAM エンティティに添付することはできません。

サービスにリンクされたロールをサポートする他のサービスについては、IAM と連携する Amazon Web Services を参照し、サービスにリンクされたロール列がYes (はい) になっているサービスを探してください。そのサービスのサービスにリンクされたロールのドキュメントを表示するには、リンク付きのはいを選択します。

トピック

- Amazon MSK のサービスリンクロールのアクセス許可
- Amazon MSK のサービスリンクロールを作成する
- Amazon MSK のサービスリンクロールを編集する
- Amazon MSK のサービスリンクロールがサポートされるリージョン

Amazon MSK のサービスリンクロールのアクセス許可

Amazon MSK では、AWSServiceRoleForKafka という名前のサービスリンクロールを使用します。Amazon MSK は、このロールを使用してリソースにアクセスし、次のようなオペレーションを実行します。

- `*NetworkInterface` — カスタマーアカウントのネットワークインターフェイスを作成および管理します。これにより、カスタマー VPC 内のクライアントがクラスターブローカーにアクセスできるようなります。
- `*VpcEndpoints` – を使用して、カスタマー VPC 内のクライアントがクラスターブローカーにアクセスできるようにする、カスタマーアカウントの VPC エンドポイントを管理します AWS PrivateLink。Amazon MSK は、`DescribeVpcEndpoints`、`ModifyVpcEndpoint`、および `DeleteVpcEndpoints` に対するアクセス許可を使用します。
- `secretsmanager` – を使用してクライアント認証情報を管理します AWS Secrets Manager。
- `GetCertificateAuthorityCertificate` — プライベート認証局の証明書を取得します。

このサービスリンクロールは、マネージドポリシー `KafkaServiceRolePolicy` にアタッチされます。このポリシーの更新については、「KafkaServiceRolePolicy」を参照してください。

AWSServiceRoleForKafka サービスリンクロールは、以下のサービスを信頼してロールを引き受けます。

- kafka.amazonaws.com

ロールのアクセス許可ポリシーは、Amazon MSK がリソースに対して以下のアクションを実行することを許可します。

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterface",
    "ec2:DescribeNetworkInterfaces",
    "ec2:CreateNetworkInterfacePermission",
    "ec2:AttachNetworkInterface",
    "ec2:DeleteNetworkInterface",
    "ec2:DetachNetworkInterface",
    "ec2:DescribeVpcEndpoints",
    "acm-pca:GetCertificateAuthorityCertificate",
    "secretsmanager:ListSecrets"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:ModifyVpcEndpoint"
   ],
   "Resource": "arn:*:ec2:*:*:subnet/*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DeleteVpcEndpoints",
    "ec2:ModifyVpcEndpoint"
   ],
   "Resource": "arn:*:ec2:*:*:vpc-endpoint/*",
```

```
    "Condition": {
     "StringEquals": {
      "ec2:ResourceTag/AWSMSKManaged": "true"
     },
     "StringLike": {
      "ec2:ResourceTag/ClusterArn": "*"
     }
    }
   },
   {
    "Effect": "Allow",
    "Action": [
     "secretsmanager:GetResourcePolicy",
     "secretsmanager:PutResourcePolicy",
     "secretsmanager:DeleteResourcePolicy",
     "secretsmanager:DescribeSecret"
    ],
    "Resource": "*",
    "Condition": {
     "ArnLike": {
      "secretsmanager:SecretId": "arn:*:secretsmanager:*:*:secret:AmazonMSK_*"
     }
    }
   }
  ]
}
```

サービスリンク役割の作成、編集、削除を IAM エンティティ (ユーザー、グループ、役割など) に許可するにはアクセス許可を設定する必要があります。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの許可」を参照してください。

Amazon MSK のサービスリンクロールを作成する

サービスにリンクされたロールを手動で作成する必要はありません。 AWS Management Console、、 AWS CLIまたは AWS API で Amazon MSK クラスターを作成すると、Amazon MSK によってサービスにリンクされたロールが作成されます。

このサービスリンクロールを削除した後で再度作成する必要が生じた場合は同じ方法でアカウントにロールを再作成できます。Amazon MSK クラスターを作成すると、Amazon MSK はサービスにリンクされたロールを再度作成します。

Amazon MSK のサービスリンクロールを編集する

Amazon MSK では、AWSServiceRoleForKafka サービスにリンクされたロールを編集することはできません。サービスにリンクされたロールを作成すると、多くのエンティティによってロールがリファレンスされる可能性があるため、ロール名を変更することはできません。ただし、IAM を使用してロールの説明を編集することはできます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。

Amazon MSK のサービスリンクロールがサポートされるリージョン

Amazon MSK は、サービスが利用可能なすべてのリージョンでサービスにリンクされたロールの使用をサポートします。詳細については、「AWS リージョンとエンドポイント」を参照してください。

AWS Amazon MSK の マネージドポリシー

AWS 管理ポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。 AWS 管理ポリシーは、多くの一般的なユースケースにアクセス許可を付与するように設計されているため、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できます。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることに注意してください。ユースケースに固有のカスタマー管理ポリシーを定義して、アクセス許可を絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS 管理ポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) に影響します。 AWS は、新しい が起動されるか、新しい API オペレーション AWS のサービス が既存のサービスで使用できるようになったときに、 AWS 管理ポリシーを更新する可能性が最も高くなります。

詳細については「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWS マネージドポリシー: AmazonMSKFullAccess

このポリシーは、プリンシパルにすべての Amazon MSK アクションへのフルアクセスを許可する管理者権限を付与します。このポリシーの権限は、次のようにグループ化されています。

- Amazon MSK 権限は、すべての Amazon MSK アクションを許可します。
- **Amazon EC2** アクセス許可 – このポリシーでは、API リクエストで渡されたリソースを検証するために必要です。これは、Amazon MSK がクラスターでリソースを正常に使用できるようにする

ためです。このポリシーの残りの Amazon EC2 アクセス許可により、Amazon MSK はクラスターへの接続を可能にするために必要な AWS リソースを作成できます。

- **AWS KMS** アクセス許可 – リクエストで渡されたリソースを検証するために、API コール中に使用されます。これらは、Amazon MSK が、渡されたキーを Amazon MSK クラスターで使用できるようにするために必要です。
- **CloudWatch Logs, Amazon S3, and Amazon Data Firehose** アクセス許可 – ログ配信先が到達可能で、ブローカーログの使用に有効であることを、Amazon MSK が確認できるようにするために必要です。
- **IAM** アクセス許可 – Amazon MSK がアカウント内でサービスリンクロールを作成できるようにし、ユーザーがサービス実行ロールを Amazon MSK に渡すことができるようにするために必要です。

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [{
   "Effect": "Allow",
   "Action": [
    "kafka:*",
    "ec2:DescribeSubnets",
    "ec2:DescribeVpcs",
    "ec2:DescribeSecurityGroups",
    "ec2:DescribeRouteTables",
    "ec2:DescribeVpcEndpoints",
    "ec2:DescribeVpcAttribute",
    "kms:DescribeKey",
    "kms:CreateGrant",
    "logs:CreateLogDelivery",
    "logs:GetLogDelivery",
    "logs:UpdateLogDelivery",
    "logs:DeleteLogDelivery",
    "logs:ListLogDeliveries",
    "logs:PutResourcePolicy",
    "logs:DescribeResourcePolicies",
    "logs:DescribeLogGroups",
    "S3:GetBucketPolicy",
    "firehose:TagDeliveryStream"
   ],
```

```
        "Resource": "*"
       },
       {
        "Effect": "Allow",
        "Action": [
         "ec2:CreateVpcEndpoint"
        ],
        "Resource": [
         "arn:*:ec2:*:*:vpc/*",
         "arn:*:ec2:*:*:subnet/*",
         "arn:*:ec2:*:*:security-group/*"
        ]
       },
       {
        "Effect": "Allow",
        "Action": [
         "ec2:CreateVpcEndpoint"
        ],
        "Resource": [
         "arn:*:ec2:*:*:vpc-endpoint/*"
        ],
        "Condition": {
         "StringEquals": {
          "aws:RequestTag/AWSMSKManaged": "true"
         },
         "StringLike": {
          "aws:RequestTag/ClusterArn": "*"
         }
        }
       },
       {
        "Effect": "Allow",
        "Action": [
         "ec2:CreateTags"
        ],
        "Resource": "arn:*:ec2:*:*:vpc-endpoint/*",
        "Condition": {
         "StringEquals": {
          "ec2:CreateAction": "CreateVpcEndpoint"
         }
        }
       },
       {
        "Effect": "Allow",
```

```
        "Action": [
         "ec2:DeleteVpcEndpoints"
        ],
        "Resource": "arn:*:ec2:*:*:vpc-endpoint/*",
        "Condition": {
         "StringEquals": {
          "ec2:ResourceTag/AWSMSKManaged": "true"
         },
         "StringLike": {
          "ec2:ResourceTag/ClusterArn": "*"
         }
        }
       },
       {
        "Effect": "Allow",
        "Action": "iam:PassRole",
        "Resource": "*",
        "Condition": {
         "StringEquals": {
          "iam:PassedToService": "kafka.amazonaws.com"
         }
        }
       },
       {
        "Effect": "Allow",
        "Action": "iam:CreateServiceLinkedRole",
        "Resource": "arn:aws:iam::*:role/aws-service-role/kafka.amazonaws.com/
AWSServiceRoleForKafka*",
        "Condition": {
         "StringLike": {
          "iam:AWSServiceName": "kafka.amazonaws.com"
         }
        }
       },
       {
        "Effect": "Allow",
        "Action": [
         "iam:AttachRolePolicy",
         "iam:PutRolePolicy"
        ],
        "Resource": "arn:aws:iam::*:role/aws-service-role/kafka.amazonaws.com/
AWSServiceRoleForKafka*"
       },
       {
```

```
       "Effect": "Allow",
       "Action": "iam:CreateServiceLinkedRole",
       "Resource": "arn:aws:iam::*:role/aws-service-role/
delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery*",
       "Condition": {
        "StringLike": {
         "iam:AWSServiceName": "delivery.logs.amazonaws.com"
        }
       }
      }

     ]
    }
```

AWS マネージドポリシー: AmazonMSKReadOnlyAccess

このポリシーは、ユーザーが Amazon MSK で情報を表示できるようにする読み取り専用の権限を付与します。このポリシーが添付されているプリンシパルは、既存のリソースを更新または削除したり、新しい Amazon MSK リソースを作成したりすることはできません。例えば、これらの権限を持つプリンシパルは、自分のアカウントに関連付けられているクラスターと構成のリストを表示できますが、クラスターの構成や設定を変更することはできません。このポリシーの権限は、次のようにグループ化されています。

- **Amazon MSK** アクセス許可 – Amazon MSK リソースを一覧表示し、記述し、それらに関する情報を取得できるようにします。
- **Amazon EC2** アクセス許可 – クラスターに関連付けられている Amazon VPC、サブネット、セキュリティグループ、および ENI を記述するために使用されます。
- **AWS KMS** アクセス許可 – クラスターに関連付けられているキーを記述するために使用されます。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "kafka:Describe*",
                "kafka:List*",
                "kafka:Get*",
```

```
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "kms:DescribeKey"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

AWS マネージドポリシー: KafkaServiceRolePolicy

KafkaServiceRolePolicy を IAM エンティティにアタッチすることはできません。このポリシーは、Amazon MSK が MSK クラスター上の VPC エンドポイント (コネクタ) の管理、ネットワークインターフェイスの管理、 AWS Secrets Managerを使用したクラスター認証情報の管理などのアクションを実行できるようにする、サービスリンクロールにアタッチされています。詳細については、「the section called “サービスにリンクされた役割”」を参照してください。

AWS マネージドポリシー: AWSMSKReplicatorExecutionRole

この `AWSMSKReplicatorExecutionRole` ポリシーは、MSK クラスター間でデータをレプリケートするためのアクセス許可を Amazon MSK Replicator に付与します。このポリシーの権限は、次のようにグループ化されています。

- **`cluster`** – IAM 認証を使用してクラスターに接続するアクセス許可を Amazon MSK Replicator に付与します。また、クラスターを記述および変更するアクセス許可も付与します。
- **`topic`** – トピックを記述、作成、変更し、トピックの動的設定を変更するアクセス許可を Amazon MSK Replicator に付与します。
- **`consumer group`** – コンシューマーグループを記述および変更し、MSK クラスターからデータを読み書きし、レプリケーターによって作成された内部トピックを削除するアクセス許可を Amazon MSK Replicator に付与します。

JSON

```
{
 "Version": "2012-10-17",
```

```
"Statement": [
 {
  "Sid": "ClusterPermissions",
  "Effect": "Allow",
  "Action": [
   "kafka-cluster:Connect",
   "kafka-cluster:DescribeCluster",
   "kafka-cluster:AlterCluster",
   "kafka-cluster:DescribeTopic",
   "kafka-cluster:CreateTopic",
   "kafka-cluster:AlterTopic",
   "kafka-cluster:WriteData",
   "kafka-cluster:ReadData",
   "kafka-cluster:AlterGroup",
   "kafka-cluster:DescribeGroup",
   "kafka-cluster:DescribeTopicDynamicConfiguration",
   "kafka-cluster:AlterTopicDynamicConfiguration",
   "kafka-cluster:WriteDataIdempotently"
  ],
  "Resource": [
   "arn:aws:kafka:*:*:cluster/*"
  ]
 },
 {
  "Sid": "TopicPermissions",
  "Effect": "Allow",
  "Action": [
   "kafka-cluster:DescribeTopic",
   "kafka-cluster:CreateTopic",
   "kafka-cluster:AlterTopic",
   "kafka-cluster:WriteData",
   "kafka-cluster:ReadData",
   "kafka-cluster:DescribeTopicDynamicConfiguration",
   "kafka-cluster:AlterTopicDynamicConfiguration",
   "kafka-cluster:AlterCluster"
  ],
  "Resource": [
   "arn:aws:kafka:*:*:topic/*/*"
  ]
 },
 {
  "Sid": "GroupPermissions",
  "Effect": "Allow",
  "Action": [
```

```
    "kafka-cluster:AlterGroup",
    "kafka-cluster:DescribeGroup"
   ],
   "Resource": [
    "arn:aws:kafka:*:*:group/*/*"
   ]
  }
 ]
}
```

AWS マネージドポリシーに対する Amazon MSK の更新

このサービスがこれらの変更の追跡を開始してからの Amazon MSK の AWS マネージドポリシーの更新に関する詳細を表示します。

変更	説明	日付
AWSMSKReplicatorExecutionRole に追加された WriteDataIdempotently アクセス許可 – 既存のポリシーの更新	Amazon MSK は、MSK クラスター間のデータレプリケーションをサポートするために、AWSMSKReplicatorExecutionRole ポリシーに WriteDataIdempotently アクセス許可を追加しました。	2024 年 3 月 12 日
AWSMSKReplicatorExecutionRole – 新しいポリシー	Amazon MSK は、 Amazon MSK Replicator をサポートする AWSMSKReplicatorExecutionRole ポリシーを追加しました。	2023 年 12 月 4 日
AmazonMSKFullAccess – 既存のポリシーに更新	Amazon MSK に、Amazon MSK Replicator をサポートするためのアクセス許可が追加されました。	2023 年 9 月 28 日
KafkaServiceRolePolicy — 既存のポリシーに対する更新	Amazon MSK に、マルチ VPC プライベート接続をサ	2023 年 3 月 8 日

変更	説明	日付
	ポートするためのアクセス許可が追加されました。	
AmazonMSKFullAccess – 既存のポリシーに更新	Amazon MSK は、クラスターへの接続を可能にする新しい Amazon EC2 権限を追加しました。	2021 年 11 月 30 日
AmazonMSKFullAccess – 既存のポリシーに更新	Amazon MSK は、Amazon EC2 ルートテーブルを記述できるようにするための新しい権限を追加しました。	2021 年 11 月 19 日
Amazon MSK が変更の追跡を開始しました	Amazon MSK は、 AWS 管理ポリシーの変更の追跡を開始しました。	2021 年 11 月 19 日

Amazon MSK の ID およびアクセスのトラブルシューティング

次の情報を使用して、Amazon MSK および IAM での作業中に発生する可能性のある一般的な問題を診断および修正するのに役立ててください。

トピック

- 私は Amazon MSK でのアクションの実行を認可されていません

私は Amazon MSK でのアクションの実行を認可されていません

でアクションを実行する権限がないと AWS Management Console 通知された場合は、管理者に連絡してサポートを依頼する必要があります。管理者とは、サインイン認証情報を提供した担当者です。

次のエラー例は、`mateojackson` IAM ユーザーがコンソールを使用してクラスターを削除しようとしているが、`kafka:DeleteCluster` 許可を持っていない場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 kafka:DeleteCluster on resource: purchaseQueriesCluster
```

この場合、Mateo は管理者に、`kafka:DeleteCluster` アクションを使用して `purchaseQueriesCluster` リソースにアクセスできるようにポリシーを更新するように依頼します。

Apache Kafka API の認証と認可

IAM を使用して、クライアントを認証し、Apache Kafka アクションを許可または拒否できます。または、TLS または SASL/SCRAM を使用してクライアントを認証し、Apache Kafka ACL を使用してアクションを許可または拒否することもできます。

クラスターで Amazon MSK オペレーションを実行できるユーザーを制御する方法については、「the section called “Amazon MSK API の認証と認可”」を参照してください。

トピック

- IAM アクセスコントロール
- Amazon MSK の相互 TLS クライアント認証
- AWS Secrets Manager を使用したサインイン認証情報認証
- Apache Kafka ACL

IAM アクセスコントロール

Amazon MSK の IAM アクセス制御により、MSK クラスターの認証と認可の両方を処理できます。これにより、認証に 1 つのメカニズムを使用し、認可に別のメカニズムを使用する必要がなくなります。たとえば、クライアントがクラスターへの書き込みを試みると、Amazon MSK は IAM を使用して、そのクライアントが認証済みアイデンティティであるかどうか、およびクラスターへの作成が認可されているかどうかをチェックします。

IAM アクセスコントロールは Java クライアントと Java 以外のクライアント (Python、Go、JavaScript、.NET で記述された Kafka クライアントを含む) で機能します。Java 以外のクライアントの IAM アクセスコントロールは、Kafka バージョン 2.7.1 以降の MSK クラスターで使用できます。

IAM アクセス制御を可能にするために、Amazon MSK は Apache Kafka ソースコードに小さな変更を加えます。これらの変更によって、Apache Kafka のエクスペリエンスに目立った違いが生じることはありません。Amazon MSK はアクセスイベントを口グに記録するため、それらを監査できます。

IAM アクセス制御を使用する MSK クラスターの Apache Kafka ACL API を呼び出すことができます。ただし、Apache Kafka ACLs は IAM ID の認可には影響しません。IAM ID へのアクセスを制御するには、IAM ポリシーを使用する必要があります。

⚠ 重要な考慮事項

MSK クラスターで IAM アクセスコントロールを使用する場合は、次の重要な考慮事項に注意してください。

- IAM アクセス制御は Apache ZooKeeper ノードには適用されません。これらのノードへのアクセスを制御する方法については、「Amazon MSK クラスター内の Apache ZooKeeper ノードへのアクセスを制御する」を参照してください。
- クラスターが IAM アクセス制御を使用している場合、`allow.everyone.if.no.acl.found` Apache Kafka 設定は効果がありません。
- IAM アクセス制御を使用する MSK クラスターの Apache Kafka ACL API を呼び出すことができます。ただし、Apache Kafka ACLs は IAM ID の認可には影響しません。IAM ID へのアクセスを制御するには、IAM ポリシーを使用する必要があります。

Amazon MSK の IAM アクセス制御の仕組み

Amazon MSK の IAM アクセス制御を使用するには、次のステップを実行します。これらの詳細については、以下のトピックで詳しく説明します。

- IAM アクセス制御を使用する Amazon MSK クラスターを作成する
- IAM アクセス制御用にクライアントを設定する
- IAM ロールの認可ポリシーを作成する
- IAM アクセス制御用のブートストラップブローカーを入手する

IAM アクセス制御を使用する Amazon MSK クラスターを作成する

このセクションでは AWS Management Console、、 API、または を使用して、IAM アクセスコントロールを使用する Amazon MSK クラスター AWS CLI を作成する方法について説明します。既存のクラスターに対して IAM アクセス制御を有効にする方法については、「Amazon MSK クラスターのセキュリティ設定を更新する」を参照してください。

AWS Management Console を使用して IAM アクセスコントロールを使用するクラスターを作成する

1. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
2. Create cluster (クラスターの作成) を選択します。
3. Create cluster with custom settings (カスタム設定でクラスターを作成) を選択します。
4. Authentication (認証) セクションで、IAM access control (IAMアクセス制御) を選択します。
5. クラスターを作成するための残りのワークフローを完了します。

API または AWS CLI を使用して、IAM アクセスコントロールを使用するクラスターを作成する

- IAM アクセス制御を有効にしてクラスターを作成するには、CreateCluster API または create-cluster CLI コマンドを使用して、`ClientAuthentication` パラメータに次の JSON を渡します：`"ClientAuthentication": { "Sasl": { "Iam": { "Enabled": true } }`。

IAM アクセス制御用にクライアントを設定する

クライアントが IAM アクセス制御を使用する MSK クラスターと通信できるようにするために、次のメカニズムのいずれかを使用できます。

- SASL_OAUTHBEARER メカニズムを使用する Java 以外のクライアント設定
- SASL_OAUTHBEARER メカニズムまたは AWS_MSK_IAM メカニズムを使用する Java のクライアント設定

SASL_OAUTHBEARER メカニズムを使用して IAM を設定する

1. 次の Python Kafka クライアントの例を使用して client.properties 設定ファイルを編集します。設定の変更は他の言語でも同様です。

```
from kafka import KafkaProducer
from kafka.errors import KafkaError
from kafka.sasl.oauth import AbstractTokenProvider
import socket
import time
from aws_msk_iam_sasl_signer import MSKAuthTokenProvider

class MSKTokenProvider():
    def token(self):
        token, _ = MSKAuthTokenProvider.generate_auth_token('<my AWS #####>')
```

```
        return token

tp = MSKTokenProvider()

producer = KafkaProducer(
    bootstrap_servers='<myBootstrapString>',
    security_protocol='SASL_SSL',
    sasl_mechanism='OAUTHBEARER',
    sasl_oauth_token_provider=tp,
    client_id=socket.gethostname(),
)

topic = "<my-topic>"
while True:
    try:
        inp=input(">")
        producer.send(topic, inp.encode())
        producer.flush()
        print("Produced!")
    except Exception:
        print("Failed to send message:", e)

producer.close()
```

2. 選択した設定言語のヘルパーライブラリをダウンロードし、その言語ライブラリのホームページの開始方法セクションの指示に従います。

   - JavaScript: https://github.com/aws/aws-msk-iam-sasl-signer-js#getting-started
   - Python: https://github.com/aws/aws-msk-iam-sasl-signer-python#get-started
   - Go: https://github.com/aws/aws-msk-iam-sasl-signer-go#getting-started
   - .NET: https://github.com/aws/aws-msk-iam-sasl-signer-net#getting-started
   - JAVA: SASL_OAUTHBEARER の Java サポートは `aws-msk-iam-auth` jar ファイルを介して提供されます

MSK カスタム AWS_MSK_IAM メカニズムを使用して IAM を設定する

1. 以下を `client.properties` ファイルに追加します。*<PATH_TO_TRUST_STORE_FILE>*を、クライアント上のトラストストアファイルへの完全修飾パスに置き換えます。

> **Note**
>
> 特定の証明書を使用しない場合は、client.properties ファイルから ssl.truststore.location=*<PATH_TO_TRUST_STORE_FILE>* を削除できます。ssl.truststore.location に値を指定しない場合、Java プロセスではデフォルトの証明書が使用されます。

```
ssl.truststore.location=<PATH_TO_TRUST_STORE_FILE>
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
```

AWS 認証情報用に作成した名前付きプロファイルを使用するには、クライアント設定ファイルに awsProfileName="*your profile name*"; を含めます。名前付きプロファイルの詳細については、 AWS CLI ドキュメントの「名前付きプロファイル」を参照してください。

2. 最新の安定したaws-msk-iam-auth JAR ファイルをダウンロードし、クラスパスに配置します。Maven を使用する場合は、次の依存関係を追加し、必要に応じてバージョン番号を調整します。

```
<dependency>
    <groupId>software.amazon.msk</groupId>
    <artifactId>aws-msk-iam-auth</artifactId>
    <version>1.0.0</version>
</dependency>
```

Amazon MSK クライアント プラグインは、Apache 2.0 ライセンスの下でオープンソース化されています。

IAM ロールの認可ポリシーを作成する

クライアントに対応する IAM ロールに認可ポリシーを添付します。認可ポリシーでは、ロールに対して許可または拒否するアクションを指定します。クライアントが Amazon EC2 インスタンス上にある場合は、認可ポリシーをその Amazon EC2 インスタンスの IAM ロールに関連付けます。または、名前付きプロファイルを使用するようにクライアントを設定してから、認可ポリシーをその名前付きプロファイルのロールに関連付けることができます。 IAM アクセス制御用にクライアントを

設定する は、名前付きプロファイルを使用するようにクライアントを設定する方法を説明しています。

IAM ポリシーを作成する方法については、IAM ポリシーの作成を参照してください。

以下は、MyTestCluster という名前のクラスターの認可ポリシーの例です。Action 要素と Resource 要素のセマンティクスを理解するには、「IAM 認可ポリシーアクションとリソースのセマンティクス」を参照してください。

> **Important**
>
> IAM ポリシーに加えた変更は、IAM API と AWS CLI にすぐに反映されます。ただし、ポリシーの変更が有効になるまでにかなりの時間がかかる場合があります。ほとんどの場合、ポリシーの変更は1分以内に有効になります。ネットワークの状態により、遅延が増える場合があります。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:Connect",
                "kafka-cluster:AlterCluster",
                "kafka-cluster:DescribeCluster"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:111122223333:cluster/MyTestCluster/
abcd1234-0123-abcd-5678-1234abcd-1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:*Topic*",
                "kafka-cluster:WriteData",
                "kafka-cluster:ReadData"
            ],
```

```
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:topic/MyTestCluster/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:AlterGroup",
                "kafka-cluster:DescribeGroup"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:group/MyTestCluster/*"
            ]
        }
    ]
}
```

データの生成や消費など、一般的な Apache Kafka のユースケースに対応するアクション要素を使用してポリシーを作成する方法については、「クライアント認可ポリシーの一般的なユースケース」を参照してください。

Kafka バージョン 2.8.0 以降では、WriteDataIdempotently アクセス許可は廃止されました (KIP-679)。デフォルトでは、`enable.idempotence = true` が設定されています。したがって、Kafka バージョン 2.8.0 以降では、IAM は Kafka ACLs と同じ機能を提供しません。トピック`WriteDataIdempotently`へのアクセスのみを提供することで、そのトピック`WriteData`にアクセスすることはできません。これは、 `WriteData`がすべてのトピックに提供される場合のケースには影響しません。その場合は、`WriteDataIdempotently` は許可されます。これは、IAM ロジックの実装と Kafka ACLsものです。さらに、トピックに冪等的に書き込むには、 へのアクセスも必要です`transactional-ids`。

これを回避するには、次のポリシーのようなポリシーを使用することをお勧めします。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "kafka-cluster:Connect",
                "kafka-cluster:AlterCluster",
                "kafka-cluster:DescribeCluster",
                "kafka-cluster:WriteDataIdempotently"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:cluster/MyTestCluster/
abcd1234-0123-abcd-5678-1234abcd-1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:*Topic*",
                "kafka-cluster:WriteData",
                "kafka-cluster:ReadData"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:topic/MyTestCluster/
abcd1234-0123-abcd-5678-1234abcd-1/TestTopic",
                "arn:aws:kafka:us-east-1:123456789012:transactional-id/
MyTestCluster/abcd1234-0123-abcd-5678-1234abcd-1/*"
            ]
        }
    ]
}
```

この場合、WriteData は TestTopic への書き込みを許可し、WriteDataIdempotently はクラスターへの冪等性書き込みを許可します。このポリシーは、必要なtransactional-idリソースへのアクセスも追加します。

WriteDataIdempotently はクラスターレベルのアクセス許可であるため、トピックレベルでは使用できません。WriteDataIdempotently がトピックレベルに制限されている場合、このポリシーは機能しません。

IAM アクセス制御用のブートストラップブローカーを入手する

「Amazon MSK クラスターのブートストラップブローカーを取得する」を参照してください。

IAM 認可ポリシーアクションとリソースのセマンティクス

現在、Amazon MSK の IAM アクセスコントロールは Kafka の内部クラスターアクションをサポートしていません。これには、Kafka がトランザクションの終了に使用する WriteTxnMarkers API が含まれます。トランザクションを終了するには、IAM 認証ではなく、適切な ACLs で SCRAM または mTLS 認証を使用することをお勧めします。

このセクションでは、IAM 認可ポリシーで使用できるアクション要素とリソース要素のセマンティクスについて説明します。ポリシーの例については[IAM ロールの認可ポリシーを作成する]を参照してください。

認可ポリシーアクション

次の表に、Amazon MSK の IAM アクセス制御を使用するときに認可ポリシーに含めることができるアクションを示します。表のアクション列のアクションを認可ポリシーに含める場合は、必須アクション列の対応するアクションも含める必要があります。

アクション	説明	必須アクション	必要なリソース	サーバーレスクラスターに適用可能
`kafka-cluster:Connect`	クラスターに接続して認証するためのアクセス許可を付与します。	なし	クラスター	あり
`kafka-cluster:DescribeCluster`	Apache Kafka の DESCRIBE CLUSTER ACL に相当する、クラスターのさまざまな側面を記述するためのアクセス許可を付与します。	`kafka-cluster:Connect`	クラスター	あり

アクション	説明	必須アクション	必要なリソース	サーバーレスクラスターに適用可能
kafka-cluster:AlterCluster	Apache Kafka の ALTER CLUSTER ACL と同等の、クラスターのさまざまな側面を変更するためのアクセス許可を付与します。	kafka-cluster:Connect kafka-cluster:DescribeCluster	クラスター	なし
kafka-cluster:DescribeClusterDynamicConfiguration	Apache Kafka の DESCRIBE_CONFIGSCLUSTER ACL に相当する、クラスターの動的設定を記述するためのアクセス許可を付与します。	kafka-cluster:Connect	クラスター	なし
kafka-cluster:AlterClusterDynamicConfiguration	Apache Kafka の ALTER_CONFIGSCLUSTER ACL に相当する、クラスターの動的設定を変更するためのアクセス許可を付与します。	kafka-cluster:Connect kafka-cluster:DescribeClusterDynamicConfiguration	クラスター	なし

アクション	説明	必須アクション	必要なリソース	サーバーレスクラスターに適用可能
kafka-cluster:WriteDataIdempotently	Apache Kafka の IDEMPOTENT_WRITE ACL CLUSTER に相当する、クラスターにデータをべき等に書き込むためのアクセス許可を付与します。	kafka-cluster:Connect kafka-cluster:WriteData	クラスター	あり
kafka-cluster:CreateTopic	Apache Kafka の CREATE ACL CLUSTER/TOPIC に相当する、クラスター上にトピックを作成するためのアクセス許可を付与します。	kafka-cluster:Connect	トピック	あり
kafka-cluster:DescribeTopic	Apache Kafka の DESCRIBE ACL TOPIC に相当する、クラスター上のトピックを記述するためのアクセス許可を付与します。	kafka-cluster:Connect	トピック	あり

アクション	説明	必須アクション	必要なリソース	サーバーレスクラスターに適用可能
kafka-cluster:AlterTopic	Apache Kafka の ALTER TOPIC ACL に相当する、クラスター上のトピックを変更するためのアクセス許可を付与します。	kafka-cluster:Connect kafka-cluster:DescribeTopic	トピック	あり
kafka-cluster:DeleteTopic	Apache Kafka の DELETE TOPIC ACL に相当する、クラスター上のトピックを削除するためのアクセス許可を付与します。	kafka-cluster:Connect kafka-cluster:DescribeTopic	トピック	あり
kafka-cluster:DescribeTopicDynamicConfiguration	Apache Kafka の DESCRIBE_CONFIGSTOPIC ACL に相当する、クラスター上のトピックの動的設定を記述するためのアクセス許可を付与します。	kafka-cluster:Connect	トピック	あり

アクション	説明	必須アクション	必要なリソース	サーバーレスクラスターに適用可能
kafka-cluster:AlterTopicDynamicConfiguration	Apache Kafka の ALTER_CONFIGSTOPIC ACL に相当する、クラスター上のトピックの動的設定を変更する許可を付与します。	kafka-cluster:Connect kafka-cluster:DescribeTopicDynamicConfiguration	トピック	あり
kafka-cluster:ReadData	Apache Kafka の READ TOPIC ACL に相当する、クラスター上のトピックからデータを読み取りためのアクセス許可を付与します。	kafka-cluster:Connect kafka-cluster:DescribeTopic kafka-cluster:AlterGroup	トピック	あり
kafka-cluster:WriteData	Apache Kafka の WRITE TOPIC ACL に相当する、クラスター上のトピックにデータを書き込みためのアクセス許可を付与します	kafka-cluster:Connect kafka-cluster:DescribeTopic	トピック	あり

アクション	説明	必須アクション	必要なリソース	サーバーレスクラスターに適用可能
kafka-cluster:DescribeGroup	Apache Kafka の DESCRIBE GROUP ACL に相当する、クラスター上のグループを記述するためのアクセス許可を付与します。	kafka-cluster:Connect	グループ	あり
kafka-cluster:AlterGroup	Apache Kafka の READ GROUP ACL に相当する、クラスター上のグループに参加するためのアクセス許可を付与します。	kafka-cluster:Connect kafka-cluster:DescribeGroup	グループ	あり
kafka-cluster:DeleteGroup	Apache Kafka の DELETE GROUP ACL に相当する、クラスター上のグループを削除するためのアクセス許可を付与します。	kafka-cluster:Connect kafka-cluster:DescribeGroup	グループ	あり

アクション	説明	必須アクション	必要なリソース	サーバーレスクラスターに適用可能
kafka-cluster:DescribeTransactionalId	Apache Kafka の DESCRIBE TRANSACTIONAL_ID ACL に相当する、クラスター上のトランザクション ID を記述するためのアクセス許可を付与します。	kafka-cluster:Connect	transactional-id	あり
kafka-cluster:AlterTransactionalId	Apache Kafka の WRITE TRANSACTIONAL_ID ACL に相当する、クラスター上のトランザクション ID を変更するためのアクセス許可を付与します。	kafka-cluster:Connect kafka-cluster:DescribeTransactionalId kafka-cluster:WriteData	transactional-id	あり

コロンの後のアクションでは、アスタリスク (*) ワイルドカードを何度でも使用できます。以下は例です。

- kafka-cluster:*Topic は、kafka-cluster:CreateTopic、kafka-cluster:DescribeTopic、kafka-cluster:AlterTopic、および kafka-cluster:DeleteTopic の略です。kafka-cluster:DescribeTopicDynamicConfiguration や kafka-cluster:AlterTopicDynamicConfiguration は含まれていません。

- `kafka-cluster:*` はすべての権限を表します。

認可ポリシーリソース

次の表は、Amazon MSK の IAM アクセスコントロールを使用するときに認可ポリシーで使用できる 4 種類のリソースを示しています。クラスターの Amazon リソースネーム (ARN) は、 から、 AWS Management Console または DescribeCluster API または describe-cluster AWS CLI コマンドを使用して取得できます。次に、クラスター ARN を使用して、トピック、グループ、およびトランザクション ID の ARN を作成できます。認可ポリシーでリソースを指定するには、そのリソースのARN を使用します。

リソース	ARN 形式
クラスター	arn:aws:kafka:*region*:*account-id* :cluster/*cluster-name* /*cluster-uuid*
トピック	arn:aws: kafka:*region*:*account-id* :topic/*cluster-name* /*cluster-uuid* /*topic-name*
グループ	arn:aws:kafka:*region*:*account-id* :topic/*cluster-name* /*cluster-uuid* /*group-name*
トランザクション ID	arn:aws:kafka:*region*:*account-id* :transactional-id/*cluster-name* /*cluster-uuid* /*transactional-id*

アスタリスク (*) ワイルドカードは、`:cluster/`、`:topic/`、`:group/`、および `:transactional-id/` の後に続く ARN の部分のどこでも何度でも使用できます。以下は、アスタリスク (*) ワイルドカードを使用して複数のリソースを参照する方法の例です。

- `arn:aws:kafka:us-east-1:0123456789012:topic/MyTestCluster/*`: クラスターの UUID に関係なく、MyTestCluster という名前のクラスター内のすべてのトピック。
- `arn:aws:kafka:us-east-1:0123456789012:topic/MyTestCluster/abcd1234-0123-abcd-5678-1234abcd-1/*_test`: 名前が MyTestCluster で、UUID が abcd1234-0123-abcd-5678-1234abcd-1 であるクラスター内で、名前が「_test」で終わるすべてのトピック。
- `arn:aws:kafka:us-east-1:0123456789012:transactional-id/MyTestCluster/*/5555abcd-1111-abcd-1234-abcd1234-1`: アカウント内の MyTestCluster という名前のクラスターのすべてのインカネーションにわたる、トランザクション ID が 5555abcd-1111-abcd-1234-abcd1234-1 であるすべてのトランザクション。つまり、MyTestCluster という名前の

クラスターを作成し、それを削除してから、同じ名前で別のクラスターを作成すると、このリソース ARN を使用して、両方のクラスターで同じトランザクション ID を表すことができます。ただし、削除されたクラスターにはアクセスできません。

クライアント認可ポリシーの一般的なユースケース

次の表の最初の列は、いくつかの一般的なユースケースを示しています。クライアントに特定のユースケースの実行を許可するには、そのユースケースに必要なアクションをクライアントの認可ポリシーに含め、`Effect` を `Allow` に設定します。

Amazon MSK の IAM アクセス制御の一部であるすべてのアクションについては、「IAM 認可ポリシーアクションとリソースのセマンティクス」を参照してください。

> **Note**
>
> アクションはデフォルトで拒否されます。クライアントに実行を認可するすべてのアクションを明示的に許可する必要があります。

ユースケース	必須アクション
管理者	`kafka-cluster:*`
トピックの作成	`kafka-cluster:Connect` `kafka-cluster:CreateTopic`

データを生成する	`kafka-cluster:Connect`
	`kafka-cluster:DescribeTopic`
	`kafka-cluster:WriteData`

データの使用	`kafka-cluster:Connect`
	`kafka-cluster:DescribeTopic`
	`kafka-cluster:DescribeGroup`
	`kafka-cluster:AlterGroup`

ユースケース	必須アクション
	kafka-cluster:ReadData
データを無差別に生成する	kafka-cluster:Connect kafka-cluster:DescribeTopic kafka-cluster:WriteData kafka-cluster:WriteDataIdempotently
トランザクションでデータを生成する	kafka-cluster:Connect kafka-cluster:DescribeTopic kafka-cluster:WriteData kafka-cluster:DescribeTransactionalId kafka-cluster:AlterTransactionalId
クラスターの設定を説明する	kafka-cluster:Connect kafka-cluster:DescribeClusterDynamicConfiguration
クラスターの設定を更新する	kafka-cluster:Connect kafka-cluster:DescribeClusterDynamicConfiguration kafka-cluster:AlterClusterDynamicConfiguration

ユースケース	必須アクション
トピックの設定を説明する	kafka-cluster:Connect kafka-cluster:DescribeTopicDynamicConfiguration
トピックの設定を更新する	kafka-cluster:Connect kafka-cluster:DescribeTopicDynamicConfiguration kafka-cluster:AlterTopicDynamicConfiguration
トピックを変更する	kafka-cluster:Connect kafka-cluster:DescribeTopic kafka-cluster:AlterTopic

Amazon MSK の相互 TLS クライアント認証

アプリケーションから Amazon MSK ブローカーへの接続に対して、TLS を使用したクライアント認証を有効にできます。クライアント認証を使用するには、 AWS Private CAが必要です。は、クラスター AWS アカウント と同じ にあるか、別のアカウントにある AWS Private CA ことができます。の詳細については、 AWS Private CA「 の作成と管理 AWS Private CA」を参照してください。

> **Note**
>
> TLS 認証は現在、北京および寧夏回族自治区では利用できません。

Amazon MSK は、証明書失効リスト (CRL) をサポートしていません。クラスタートピックへのアクセスを制御したり、侵害された証明書をブロックしたりするには、Apache Kafka ACLs と AWS セキュリティグループを使用します。Apache Kafka ACL の使用については、「the section called “Apache Kafka ACL”」を参照してください。

このトピックには、次のセクションが含まれています。

- クライアント認証をサポートする Amazon MSK クラスターを作成します。
- 認証を使用するようにクライアントを設定する
- 認証を使用してメッセージを生成および消費する

クライアント認証をサポートする Amazon MSK クラスターを作成します。

この手順では、 を使用してクライアント認証を有効にする方法を示します AWS Private CA。

> **Note**
>
> 相互 TLS を使用してアクセスを制御する場合は AWS Private CA 、MSK クラスターごとに独立した を使用することを強くお勧めします。そうすることで、PCA によって署名された TLS 証明書が単一の MSK クラスターでのみ認証されるようになります。

1. 次の内容で、clientauthinfo.json という名前のファイルを作成します。*Private-CA-ARN* を PCA の ARN に置き換えます。

```
{
   "Tls": {
       "CertificateAuthorityArnList": ["Private-CA-ARN"]
    }
}
```

2. the section called “を使用してプロビジョニングされた Amazon MSK クラスターを作成する AWS CLI” の説明に従って、brokernodegroupinfo.json という名前のファイルを作成します。
3. クライアント認証では、クライアントとブローカー間の転送中に暗号化を有効にする必要があります。次の内容で、encryptioninfo.json という名前のファイルを作成します。*KMS-Key-ARN* を KMS キーの ARN と置き換えます。ClientBroker を TLS または TLS_PLAINTEXT に設定できます。

```
{
   "EncryptionAtRest": {
       "DataVolumeKMSKeyId": "KMS-Key-ARN"
    },
   "EncryptionInTransit": {
        "InCluster": true,
        "ClientBroker": "TLS"
```

```
    }
}
```

暗号化の詳細については、「the section called “Amazon MSK 暗号化”」を参照してください。

4. AWS CLI がインストールされているマシンで、次のコマンドを実行して、認証と転送中の暗号化が有効になっているクラスターを作成します。レスポンスで提供されるクラスター ARN を保存します。

```
aws kafka create-cluster --cluster-name "AuthenticationTest" --broker-node-group-
info file://brokernodegroupinfo.json --encryption-info file://encryptioninfo.json
 --client-authentication file://clientauthinfo.json --kafka-version "{YOUR KAFKA
 VERSION}" --number-of-broker-nodes 3
```

認証を使用するようにクライアントを設定する

このプロセスでは、認証を使用するクライアントとして使用する Amazon EC2 インスタンスを設定する方法について説明します。

このプロセスでは、クライアントマシンの作成、トピックの作成、および必要なセキュリティ設定により、認証を使用してメッセージを生成および消費する方法について説明します。

1. クライアントマシンとして使用する Amazon EC2 インスタンスを作成します。わかりやすくするために、クラスターで使用したのと同じ VPC にこのインスタンスを作成します。このようなクライアントマシンの作成方法の例については、「the section called “クライアントマシンを作成する”」を参照してください。
2. [Create a topic] (トピックの作成) 例については、「the section called “トピックの作成”」の手順を参照してください。
3. AWS CLI がインストールされているマシンで、次のコマンドを実行してクラスターのブートストラップブローカーを取得します。*Cluster-ARN* をクラスターの ARN に置き換えます。

```
aws kafka get-bootstrap-brokers --cluster-arn Cluster-ARN
```

BootstrapBrokerStringTls に関連付けられた文字列をレスポンスに保存します。

4. クライアントマシンで次のコマンドを実行して、JVM トラストストアを使用してクライアントトラストストアを作成します。JVM パスが異なる場合は、それに応じてコマンドを調整します。

```
cp /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.201.b09-0.amzn2.x86_64/jre/lib/security/
cacerts kafka.client.truststore.jks
```

5. クライアントマシンで次のコマンドを実行して、クライアントのプライベートキーを作成します。*Distinguished-Name*、*Example-Alias*、*Your-Store-Pass*、および*Your-Key-Pass*を任意の文字列に置き換えます。

```
keytool -genkey -keystore kafka.client.keystore.jks -validity 300 -storepass Your-
Store-Pass -keypass Your-Key-Pass -dname "CN=Distinguished-Name" -alias Example-
Alias -storetype pkcs12 -keyalg rsa
```

6. クライアントマシンで次のコマンドを実行して、前のステップで作成したプライベートキーを使用して証明書リクエストを作成します。

```
keytool -keystore kafka.client.keystore.jks -certreq -file client-cert-sign-request
 -alias Example-Alias -storepass Your-Store-Pass -keypass Your-Key-Pass
```

7. client-cert-sign-request ファイルを開き、それが -----BEGIN CERTIFICATE REQUEST----- で始まり、-----END CERTIFICATE REQUEST----- で終わることを確認します。-----BEGIN NEW CERTIFICATE REQUEST----- で始まる場合は、ファイルの先頭と末尾から単語 NEW (およびそれに続く単一のスペース) を削除します。

8. AWS CLI がインストールされているマシンで、次のコマンドを実行して証明書リクエストに署名します。*Private-CA-ARN* を PCA の ARN に置き換えます。必要に応じて、有効性の値を変更できます。ここでは、例として 300 を使用します。

```
aws acm-pca issue-certificate --certificate-authority-arn Private-CA-ARN --csr
 fileb://client-cert-sign-request --signing-algorithm "SHA256WITHRSA" --validity
 Value=300,Type="DAYS"
```

レスポンスで提供された証明書 ARN を保存します。

> **Note**
>
> クライアント証明書を取得するには、acm-pca get-certificate コマンドを使用して証明書 ARN を指定します。詳細については、「AWS CLI コマンドリファレンス」の「get-certificate」を参照してください。

9. 次のコマンドを実行して、 AWS Private CA が署名した証明書を取得します。*Certificate-ARN* を、前のコマンドに対するレスポンスから取得した ARN に置き換えます。

```
aws acm-pca get-certificate --certificate-authority-arn Private-CA-ARN --
certificate-arn Certificate-ARN
```

10. 前のコマンドを実行した JSON 結果から、Certificate および CertificateChain に関連付けられた文字列をコピーします。これらの 2 つの文字列を、signed-certificate-from-acm という名前の新しいファイルに貼り付けます。Certificate に関連付けられた文字列を貼り付けます。次に、CertificateChain に関連付けられた文字列を貼り付けます。\n 文字を新しい行に置き換えます。証明書と証明書チェーンを貼り付けた後のファイルの構造を次に示します。

```
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
```

11. 次のコマンドを実行して、この証明書をキーストアに追加し、MSK ブローカーと対話するときに提示できるようにします。

```
keytool -keystore kafka.client.keystore.jks -import -file signed-certificate-from-
acm -alias Example-Alias -storepass Your-Store-Pass -keypass Your-Key-Pass
```

12. 次の内容で、client.properties という名前のファイルを作成します。トラストストアとキーストアの場所を、kafka.client.truststore.jks を保存したパスに調整します。*{YOUR KAFKA VERSION}* プレースホルダーは、ご使用の Kafka クライアントバージョンに置き換えてください。

```
security.protocol=SSL
ssl.truststore.location=/tmp/kafka_2.12-{YOUR KAFKA VERSION}/
kafka.client.truststore.jks
ssl.keystore.location=/tmp/kafka_2.12-{YOUR KAFKA VERSION}/
kafka.client.keystore.jks
ssl.keystore.password=Your-Store-Pass
ssl.key.password=Your-Key-Pass
```

認証を使用してメッセージを生成および消費する

このプロセスでは、認証を使用してメッセージを生成および消費する方法について説明します。

1. 次のコマンドを実行して、トピックを作成します。client.properties という名前のファイルは、前の手順で作成したファイルです。

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --create --bootstrap-
server BootstrapBroker-String --replication-factor 3 --partitions 1 --topic
 ExampleTopic --command-config client.properties
```

2. 次のコマンドを実行して、コンソールプロデューサーを起動します。client.properties という名前のファイルは、前の手順で作成したファイルです。

```
<path-to-your-kafka-installation>/bin/kafka-console-producer.sh --bootstrap-
server BootstrapBroker-String --topic ExampleTopic --producer.config
 client.properties
```

3. クライアントマシンの新しいコマンドウィンドウで、次のコマンドを実行してコンソールコンシューマーを起動します。

```
<path-to-your-kafka-installation>/bin/kafka-console-consumer.sh --bootstrap-
server BootstrapBroker-String --topic ExampleTopic --consumer.config
 client.properties
```

4. プロデューサーウィンドウにメッセージを入力し、コンシューマーウィンドウに表示されるようにします。

AWS Secrets Manager を使用したサインイン認証情報認証

AWS Secrets Manager を使用して保存および保護されるサインイン認証情報を使用して、Amazon MSK クラスターへのアクセスを制御できます。ユーザーの認証情報を Secrets Manager に保存すると、認証情報の監査、更新、ローテーションなど、クラスター認証のオーバーヘッドが削減されます。Secrets Manager を使用すると、クラスター間でユーザーの認証情報を共有することもできます。

シークレットを MSK クラスターに関連付けると、MSK は認証情報データを定期的に同期します。

このトピックには、次のセクションが含まれています。

- サインイン認証情報認証の仕組み

- Amazon MSK クラスターの SASL/SCRAM 認証を設定する
- ユーザーの使用
- SCRAM シークレットを使用する場合の制限事項

サインイン認証情報認証の仕組み

Amazon MSK のサインイン認証情報認証では、SASL/SCRAM (Simple Authentication and Security Layer/Salted Challenge Response Mechanism) 認証を使用します。クラスターのサインイン認証情報認証を設定するには、「AWS Secrets Manager」でシークレットリソースを作成し、サインイン認証情報をそのシークレットに関連付けます。

SASL/SCRAMは、RFC 5802で定義されています。SCRAM は、セキュリティで保護されたハッシュアルゴリズムを使用し、クライアントとサーバー間でプレーンテキストのサインイン認証情報を送信しません。

> **Note**
>
> クラスターに SASL/SCRAM 認証を設定すると、Amazon MSK はクライアントとブローカー間のすべてのトラフィックに対して TLS 暗号化をオンにします。

Amazon MSK クラスターの SASL/SCRAM 認証を設定する

AWS Secrets Manager でシークレットを設定するには、Secrets Manager ユーザーガイドの「シークレットの作成と取得」チュートリアルに従います。 AWS

Amazon MSK クラスターのシークレットを作成するときは、次の要件に注意してください。

- シークレットタイプには、他のタイプのシークレット (API キーなど) を選択します。
- シークレット名は、プレフィックス AmazonMSK_ から始まる必要があります。
- 既存のカスタム AWS KMS キーを使用するか、シークレットの新しいカスタム AWS KMS キーを作成する必要があります。Secrets Manager は、デフォルトでシークレットのデフォルト AWS KMS キーを使用します。

> **Important**
>
> デフォルト AWS KMS キーで作成されたシークレットは、Amazon MSK クラスターでは使用できません。

- [プレーンテキスト] オプションを使用してキーと値のペアを入力するには、サインイン認証情報データは次の形式である必要があります。

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

- シークレットの ARN (Amazon リソース名) 値をレコードします。
-

> **Important**
>
> 「the section called “ クラスターの適切なサイズ設定: 標準ブローカーあたりのパーティション数”」で説明されている制限を超えるクラスターに Secrets Manager シークレットを関連付けることはできません。

- を使用してシークレット AWS CLI を作成する場合は、 `kms-key-id`パラメータのキー ID または ARN を指定します。エイリアスは指定しないでください。
- シークレットをクラスターに関連付けるには、Amazon MSK コンソールまたは BatchAssociateScramSecret オペレーションのいずれかを使用します。

> **Important**
>
> シークレットをクラスターに関連付けると、Amazon MSK はそのシークレットにリソースポリシーをアタッチします。これにより、定義したシークレット値にクラスターがアクセスして読み取ることができるようになります。このリソースポリシーは変更しないでください。変更すると、クラスターがシークレットにアクセスできなくなる可能性があります。Secrets リソースポリシーやシークレット暗号化に使用される KMS キーに変更を加える場合は、シークレットを MSK クラスターに再関連付けしてください。これにより、クラスターがシークレットに引き続きアクセスできるようになります。

次の `BatchAssociateScramSecret` オペレーションの JSON 入力の例は、シークレットをクラスターに関連付けます。

```
{
  "clusterArn" : "arn:aws:kafka:us-west-2:0123456789019:cluster/SalesCluster/
abcd1234-abcd-cafe-abab-9876543210ab-4",
  "secretArnList": [
```

```
    "arn:aws:secretsmanager:us-west-2:0123456789019:secret:AmazonMSK_MyClusterSecret"
  ]
}
```

サインイン認証情報を使用したクラスターへの接続

シークレットを作成してクラスターに関連付けると、クライアントをクラスターに接続できます。次の手順は、SASL/SCRAM 認証を使用するクラスターにクライアントを接続する方法を示しています。また、サンプルトピックに対して生成および消費する方法も示します。

トピック

- SASL/SCRAM 認証を使用してクライアントをクラスターに接続する
- 接続の問題のトラブルシューティング

SASL/SCRAM 認証を使用してクライアントをクラスターに接続する

1. AWS CLI がインストールされているマシンで次のコマンドを実行します。*clusterARN* をクラスターの ARN に置き換えます。

   ```
   aws kafka get-bootstrap-brokers --cluster-arn clusterARN
   ```

   このコマンドの JSON 結果から、 という名前の文字列に関連付けられた値を保存しますBootstrapBrokerStringSaslScram。この値は後のステップで使用します。

2. クライアントマシンで、シークレットに保存されているユーザー認証情報を含む JAAS 設定ファイルを作成します。例えば、ユーザー alice の場合、次の内容を含む users_jaas.conf という名前のファイルを作成します。

   ```
   KafkaClient {
      org.apache.kafka.common.security.scram.ScramLoginModule required
      username="alice"
      password="alice-secret";
   };
   ```

3. 次のコマンドを使用して、JAAS 設定ファイルを KAFKA_OPTS 環境パラメータとしてエクスポートします。

```
export KAFKA_OPTS=-Djava.security.auth.login.config=<path-to-jaas-file>/
users_jaas.conf
```

4. /tmp ディレクトリに kafka.client.truststore.jks という名前のファイルを作成します。

5. (オプション) 次のコマンドを使用して、JVM cacertsフォルダから JDK キーストアファイルを前のステップで作成したkafka.client.truststore.jksファイルにコピーします。*JDKFolder* は、インスタンス上の JDK フォルダの名前に置き換えてください。例えば、JDK フォルダには java-1.8.0-openjdk-1.8.0.201.b09-0.amzn2.x86_64 という名前が付いている場合があります。

```
cp /usr/lib/jvm/JDKFolder/lib/security/cacerts /tmp/kafka.client.truststore.jks
```

6. Apache Kafka のインストール済み環境の bin ディレクトリに、次の内容を含む client_sasl.properties という名前のクライアントプロパティファイルを作成します。このファイルは、SASL メカニズムとプロトコルを定義します。

```
security.protocol=SASL_SSL
sasl.mechanism=SCRAM-SHA-512
```

7. サンプルトピックを作成するには、次のコマンドを実行します。*BootstrapBrokerStringSaslScram* を、このトピックのステップ 1 で取得したブートストラップブローカー文字列に置き換えます。

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --create --bootstrap-
server BootstrapBrokerStringSaslScram --command-config <path-to-client-
properties>/client_sasl.properties --replication-factor 3 --partitions 1 --topic
 ExampleTopicName
```

8. 作成したサンプルトピックにデータを生成するには、クライアントマシンで次のコマンドを実行します。*BootstrapBrokerStringSaslScram* を、このトピックのステップ 1 で取得したブートストラップブローカー文字列に置き換えます。

```
<path-to-your-kafka-installation>/bin/kafka-console-producer.sh --broker-
list BootstrapBrokerStringSaslScram --topic ExampleTopicName --producer.config
 client_sasl.properties
```

9. 作成したトピックからデータを消費するには、クライアントマシンで次のコマンドを実行します。*BootstrapBrokerStringSaslScram* を、このトピックのステップ 1 で取得したブートストラップブローカー文字列に置き換えます。

```
<path-to-your-kafka-installation>/bin/kafka-console-consumer.sh --bootstrap-
server BootstrapBrokerStringSaslScram --topic ExampleTopicName --from-beginning --
consumer.config client_sasl.properties
```

接続の問題のトラブルシューティング

Kafka クライアントコマンドを実行すると、特に大きなトピックやデータセットを操作するときに、Java ヒープメモリエラーが発生する可能性があります。これらのエラーは、Kafka ツールがデフォルトのメモリ設定で Java アプリケーションとして実行され、ワークロードに不十分な可能性があるために発生します。

`Out of Memory Java Heap` エラーを解決するには、`KAFKA_OPTS`環境変数を変更してメモリ設定を含めることで、Java ヒープサイズを増やすことができます。

次の例では、最大ヒープサイズを 1GB () に設定します`-Xmx1G`。この値は、使用可能なシステムメモリと要件に基づいて調整できます。

```
export KAFKA_OPTS="-Djava.security.auth.login.config=<path-to-jaas-file>/
users_jaas.conf -Xmx1G"
```

大規模なトピックを使用する場合は、メモリ使用量を制限するのではなく`--from-beginning`、時間ベースまたはオフセットベースのパラメータを使用することを検討してください。

```
<path-to-your-kafka-installation>/bin/kafka-console-consumer.sh --bootstrap-
server BootstrapBrokerStringSaslScram --topic ExampleTopicName --max-messages 1000 --
consumer.config client_sasl.properties
```

ユーザーの使用

ユーザーの作成：キーバリューのペアとして秘密のユーザーを作成します。Secrets Manager コンソールで [プレーンテキスト] オプションを使用する場合は、サインイン認証情報データを次の形式で指定する必要があります。

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

```
}
```

ユーザーアクセスの取り消し：クラスターにアクセスするためのユーザーの認証情報を取り消すには、最初にクラスターの ACL を削除または適用してから、シークレットの関連付けを解除することをお勧めします。これは、次の理由によるものです。

- ユーザーを削除しても既存の接続は閉じられません。
- シークレットへの変更が反映されるまでに最大 10 分かかります。

Amazon MSK で ACL を使用する方法については、「Apache Kafka ACL」を参照してください。

ZooKeeper モードを使用するクラスターの場合、ユーザーが ACL を変更できないように、ZooKeeper ノードへのアクセスを制限することをお勧めします。詳細については、「Amazon MSK クラスター内の Apache ZooKeeper ノードへのアクセスを制御する」を参照してください。

SCRAM シークレットを使用する場合の制限事項

SCRAM シークレットを使用する場合は、次の制限に注意してください。

- Amazon MSK は、SCRAM-SHA-512 認証のみをサポートします。
- Amazon MSK クラスターには、最大 1000 人のユーザーを含めることができます。
- シークレット AWS KMS key で を使用する必要があります。デフォルトの Secrets Manager 暗号化キーを使用するシークレットを Amazon MSK で使用することはできません。KMS キーの作成については、「対称暗号化 KMS キーの作成」を参照してください。
- Secrets Manager で非対称 KMS キーを使用することはできません。
- BatchAssociateScramSecret オペレーションを使用して、一度に最大 10 個のシークレットをクラスターに関連付けることができます。
- Amazon MSK クラスターに関連付けられているシークレットの名前には、プレフィックス Amazon MSK_ が必要です。
- Amazon MSK クラスターに関連付けられたシークレットは、クラスターと同じ Amazon Web Services アカウントと AWS リージョンに存在する必要があります。

Apache Kafka ACL

Apache Kafka には、プラグイン可能なオーソライザーがあり、すぐに使用できるオーソライザーの実装とともに出荷されます。Amazon MSK では、ブローカーの `server.properties` ファイルでこのオーソライザーが有効になります。

Apache Kafka ACL の形式は、「プリンシパル P は [許可/拒否] オペレーション O です。ResourcePattern RP に一致する任意のリソース R のホスト H から」です。RP が特定のリソース R と一致しない場合、R には ACL が関連付けられていないため、スーパーユーザー以外は R にアクセスできません。この Apache Kafka の動作を変更するには、プロパティ `allow.everyone.if.no.acl.found` を true に設定します。Amazon MSK は、デフォルトで true に設定します。これは、Amazon MSK クラスターでは、リソースに ACL を明示的に設定しない場合、すべてのプリンシパルがこのリソースにアクセスできることを意味します。リソースで ACL を有効にすると、許可されたプリンシパルのみがリソースにアクセスできます。トピックへのアクセスを制限し、TLS 相互認証を使用してクライアントを認可する場合は、Apache Kafka オーソライザー CLI を使用して ACL を追加します。ACL の追加、削除、および一覧表示の詳細については、Kafka 認可コマンドラインインターフェイスを参照してください。

Amazon MSK はブローカーをスーパーユーザーとして設定するため、すべてのトピックにアクセスできます。これにより、プロパティがクラスターの設定に定義されているかどうかにかかわらず、ブローカー`allow.everyone.if.no.acl.found`はプライマリパーティションからメッセージをレプリケートできます。

トピックに対する読み取りおよび書き込みアクセス権を追加するか削除するには

1. ブローカーを ACL テーブルに追加して、ACL が設定されているすべてのトピックから読み取りを実行できるようにします。ブローカーにトピックへの読み取りアクセスを許可するには、MSK クラスターと通信できるクライアントマシンで次のコマンドを実行します。

   Distinguished-Name をクラスターのブートストラップブローカーの DNS に置き換え、この識別名の最初のピリオドより前の文字列をアスタリスク (*) に置き換えます。たとえば、クラスターのブートストラップブローカーの 1 つに DNS `b-6.mytestcluster.67281x.c4.kafka.us-east-1.amazonaws.com` がある場合は、以下のコマンドの *Distinguished-Name* を `*.mytestcluster.67281x.c4.kafka.us-east-1.amazonaws.com` に置き換えます。ブートストラップブローカーを取得する方法については、「the section called “ブートストラップブローカーを取得する”」を参照してください。

   ```
   <path-to-your-kafka-installation>/bin/kafka-acls.sh --bootstrap-server
    BootstrapServerString --add --allow-principal "User:CN=Distinguished-Name" --
   operation Read --group=* --topic Topic-Name
   ```

2. クライアントアプリケーションにトピックへの読み取りアクセスを許可するには、クライアントマシンで次のコマンドを実行します。相互 TLS 認証を使用する場合は、プライベートキーの作成時に使用したのと同じ *Distinguished-Name* を使用します。

```
<path-to-your-kafka-installation>/bin/kafka-acls.sh --bootstrap-server
 BootstrapServerString --add --allow-principal "User:CN=Distinguished-Name" --
operation Read --group=* --topic Topic-Name
```

読み取りアクセス権を削除するには、同じコマンドを実行し、--add を --remove に置き換えます。

3. トピックへの書き込みアクセス権を付与するには、クライアントマシンで次のコマンドを実行します。相互 TLS 認証を使用する場合は、プライベートキーの作成時に使用したのと同じ *Distinguished-Name* を使用します。

```
<path-to-your-kafka-installation>/bin/kafka-acls.sh --bootstrap-server
 BootstrapServerString --add --allow-principal "User:CN=Distinguished-Name" --
operation Write --topic Topic-Name
```

書き込みアクセス権を削除するには、同じコマンドを実行し、--add を --remove に置き換えます。

Amazon MSK クラスターのセキュリティグループの変更

このページでは、既存の MSK クラスターのセキュリティグループを変更する方法について説明します。特定のユーザーセットにアクセスを提供したり、クラスターへのアクセスを制限したりするために、クラスターのセキュリティグループを変更する必要がある場合があります。セキュリティグループの詳細については、「Amazon VPC ユーザーガイド」のVPCのセキュリティグループを参照してください。

1. クラスター内のブローカーのリストを取得する AWS CLI には、 の ListNodes API または list-nodes コマンドを使用します。このオペレーションの結果には、ブローカーに関連付けられている Elastic Network Interface (ENI) の ID が含まれます。
2. にサインイン AWS Management Console し、https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
3. 画面の右上隅にあるドロップダウンリストを使用して、クラスターが展開されているリージョンを選択します。
4. 左側のペインの Network & Security (ネットワークとセキュリティ) で、Network Interfaces (ネットワークインターフェイス) を選択します。

5. 最初のステップで取得した最初の ENI を選択します。画面上部の Actions (アクション) メニューを選択し、[Change Security Groups] (セキュリティグループの変更) を選択します。この ENI に新しいセキュリティグループを割り当てます。最初のステップで取得した ENI ごとに、このステップを繰り返します。

> **Note**
>
> Amazon EC2 コンソールを使用してクラスターのセキュリティグループに対して行った変更は、MSK コンソールの [ネットワーク設定] には反映されません。

6. クライアントがブローカーにアクセスできるように、新しいセキュリティグループのルールを構成します。セキュリティグループルールの設定については、「Amazon VPC ユーザーガイド」の[ルールの追加、削除、更新]を参照してください。

> **Important**
>
> クラスターのブローカーに関連付けられているセキュリティグループを変更してから、そのクラスターに新しいブローカーを追加すると、Amazon MSK は、クラスターの作成時にクラスターに関連付けられていた元のセキュリティグループに新しいブローカーを関連付けます。ただし、クラスターが正しく機能するには、すべてのブローカーが同じセキュリティグループに関連付けられている必要があります。したがって、セキュリティグループを変更した後に新しいブローカーを追加する場合は、前の手順を再度実行して、新しいブローカーの ENI を更新する必要があります。

Amazon MSK クラスター内の Apache ZooKeeper ノードへのアクセスを制御する

セキュリティ上の理由から、Amazon MSK クラスターの一部である Apache ZooKeeper ノードへのアクセスを制限できます。ノードへのアクセスを制限するには、それらに別のセキュリティグループを割り当てます。その後、そのセキュリティグループにアクセスできるユーザーを決定できます。

> **Important**
>
> このセクションは、KRaft モードで実行されているクラスターには適用されません。「the section called “KRaft モード”」を参照してください。

このトピックには、次のセクションが含まれています。

- Apache ZooKeeper ノードを別のセキュリティグループに配置するには
- Apache ZooKeeper での TLS セキュリティの使用

Apache ZooKeeper ノードを別のセキュリティグループに配置するには

Apache ZooKeeper ノードへのアクセスを制限するには、それらに別のセキュリティグループを割り当てます。セキュリティグループルールを設定することで、この新しいセキュリティグループにアクセスできるユーザーを選択できます。

1. クラスターの Apache ZooKeeper 接続文字列を取得します。この方法の詳細は、「the section called “ZooKeeper モード”」を参照してください。接続文字列には、Apache ZooKeeper ノードの DNS 名が含まれています。
2. `host` や `ping` などのツールを使用して、前の手順で取得した DNS 名を IP アドレスに変換します。この手順の後半で必要になるため、これらの IP アドレスを保存します。
3. にサインイン AWS Management Console し、https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
4. 左側のペインの [Network & Security (ネットワークとセキュリティ)] で、[Network Interfaces (ネットワークインターフェイス)] を選択します。
5. ネットワークインターフェイスのテーブルの上にある検索フィールドに、クラスターの名前を入力し、「return」と入力します。これにより、テーブルに表示されるネットワークインターフェイスの数を、クラスターに関連付けられたインターフェイスの数に制限します。
6. リストの最初のネットワークインターフェイスに対応する行の先頭にあるチェックボックスをオンにします。
7. ページの下部にある詳細ペインで、[Primary private IPv4 IP (プライマリプライベート IPv4 IP)] を探します。この IP アドレスがこの手順の最初に取得した IP アドレスのいずれかに一致する場合は、このネットワークインターフェイスがクラスターの一部である Apache ZooKeeper ノードに割り当てられていることを意味します。それ以外の場合は、このネットワークインターフェイスの隣にあるチェックボックスをオフにして、リスト内の次のネットワークインターフェイスを選択します。ネットワークインターフェイスを選択する順序は関係ありません。以降のステップでは、Apache ZooKeeper ノードに割り当てられているすべてのネットワークインターフェイスで、同じオペレーションを 1 つずつ実行します。
8. Apache ZooKeeper ノードに対応するネットワークインターフェイスを選択する場合は、ページ上部の [Actions (アクション)] メニューを選択し、[Change Security Groups (セキュリティグループを変更)] を選択します。このネットワークインターフェイスに新しいセキュリティグルー

プを割り当てます。セキュリティグループの作成については、Amazon VPC ドキュメントのセキュリティグループの作成を参照してください。

9. 前のステップを繰り返して、クラスターの Apache ZooKeeper ノードに関連付けられているすべてのネットワークインターフェイスに同じ新しいセキュリティグループを割り当てます。
10. これで、この新しいセキュリティグループにアクセスできるユーザーを選択できるようになりました。セキュリティグループルールの設定については、Amazon VPC ドキュメントのルールの追加、削除、更新を参照してください。

Apache ZooKeeper での TLS セキュリティの使用

クライアントと Apache ZooKeeper ノード間の転送中の暗号化に TLS セキュリティを使用できます。Apache ZooKeeper ノードで TLS セキュリティを実装するには、次の手順を実行します。

- Apache ZooKeeper で TLS セキュリティを使用するには、クラスターで Apache Kafka バージョン 2.5.1 以降を使用する必要があります。
- クラスターを作成または構成するときに TLS セキュリティを有効にします。Apache Kafka バージョン 2.5.1 以降で TLS を有効にして作成されたクラスターは、Apache ZooKeeper エンドポイントで TLS セキュリティを自動的に使用します。TLS セキュリティの設定については、「Amazon MSK 暗号化の使用を開始する」を参照してください。
- DescribeCluster オペレーションを使用して TLS Apache ZooKeeper エンドポイントを取得します。
- `kafka-configs.sh` および `kafka-acls.sh` ツールで使用する、または ZooKeeper シェルで使用する Apache ZooKeeper 設定ファイルを作成します。各ツールでは、`--zk-tls-config-file` パラメータを使用して Apache ZooKeeper の設定を指定します。

  次の例は、典型的な Apache ZooKeeper 設定ファイルを示しています。

```
zookeeper.ssl.client.enable=true
zookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
zookeeper.ssl.keystore.location=kafka.jks
zookeeper.ssl.keystore.password=test1234
zookeeper.ssl.truststore.location=truststore.jks
zookeeper.ssl.truststore.password=test1234
```

- その他のコマンド (`kafka-topics` など) の場合は、`KAFKA_OPTS` 環境変数を使用して Apache ZooKeeper パラメーターを設定する必要があります。次の例は、Apache ZooKeeper パラメーターを他のコマンドに渡すように `KAFKA_OPTS` 環境変数を設定する方法を示しています。

```
export KAFKA_OPTS="
-Dzookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
-Dzookeeper.client.secure=true
-Dzookeeper.ssl.trustStore.location=/home/ec2-user/kafka.client.truststore.jks
-Dzookeeper.ssl.trustStore.password=changeit"
```

KAFKA_OPTS 環境変数を設定すると、CLI コマンドを通常どおりに使用できます。次の例では、KAFKA_OPTS 環境変数の Apache ZooKeeper 設定を使用して Apache Kafka トピックを作成します。

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --create --
zookeeper ZooKeeperTLSConnectString --replication-factor 3 --partitions 1 --topic
 AWSKafkaTutorialTopic
```

> **Note**
>
> Apache ZooKeeper 設定ファイルで使用するパラメーターの名前と、KAFKA_OPTS 環境可変で使用するパラメーターの名前は一貫していません。設定ファイルおよび KAFKA_OPTS 環境変数の、どのパラメーターでどの名前を使用するかに注意してください。

TLS を使用して Apache ZooKeeper ノードにアクセスする方法の詳細については、KIP-515: ZK クライアントが新しい TLS でサポートされている認証を使用できるようにするを参照してください。

Amazon Managed Streaming for Apache Kafka のコンプライアンス検証

サードパーティーの監査人は、 AWS コンプライアンスプログラムの一環として、Amazon Managed Streaming for Apache Kafka とコンプライアンスを評価します。これらには、PCI および HIPAA BAA が含まれます。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、「コンプライアンスプログラムによる対象範囲内のアマゾン サービスコンプライアンスプログラム」を参照してください。一般的な情報については、AWS 「 Compliance ProgramsAssurance」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

Amazon MSK を使用する際のお客様のコンプライアンス責任は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 は、コンプライアンスに役立つ以下のリソース AWS を提供します。

- 「セキュリティ＆コンプライアンスクイックリファレンスガイド」 – これらのデプロイガイドには、アーキテクチャ上の考慮事項の説明と、 AWSでセキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイするための手順が記載されています。
- 「Architecting for HIPAA Security and Compliance」ホワイトペーパー – このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- 「 デベロッパーガイド」の「ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が内部プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – この AWS サービスは、 内のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立ちます。

Amazon Managed Streaming for Apache Kafka の復元力

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。 AWS リージョンは、低レイテンシー、高スループット、高度に冗長なネットワークで接続された、物理的に分離された複数のアベイラビリティーゾーンを提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

Amazon Managed Streaming for Apache Kafka におけるインフラストラクチャセキュリティ

マネージドサービスである Amazon Managed Streaming for Apache Kafka は、ホワイトペーパー「Amazon Web Services: セキュリティプロセスの概要」に記載されている AWS グローバルネットワークセキュリティ手順で保護されています。

AWS 公開された API コールを使用して、ネットワーク経由で Amazon MSK にアクセスします。クライアントで Transport Layer Security (TLS) 1.0 以降がサポートされている必要があります。TLS 1.2 以降が推奨されています。また、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポートされている必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。またはAWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

Amazon MSK プロビジョンド設定

Amazon MSK は、ブローカー、トピック、メタデータノードのデフォルト設定を提供します。また、カスタム設定を作成し、それらを使用して新しい MSK クラスターを作成したり、既存のクラスターを更新することもできます。MSK 設定は、一連のプロパティとそれに対応する値で構成されます。クラスターで使用するブローカータイプに応じて、設定のデフォルトと変更できる設定のセットが異なります。Standard ブローカーと Express ブローカーの設定方法の詳細については、以下のセクションを参照してください。

トピック

- 標準ブローカー設定
- Express ブローカーの設定
- ブローカー設定オペレーション

標準ブローカー設定

このセクションでは、標準ブローカーの設定プロパティについて説明します。

トピック

- カスタム Amazon MSK 設定
- デフォルト Amazon MSK 設定
- Amazon MSK 階層型ストレージのトピックレベル設定に関するガイドライン

カスタム Amazon MSK 設定

Amazon MSK を使用して、次のプロパティを設定するカスタム MSK 設定を作成できます。明示的に設定しないプロパティは、the section called “デフォルト Amazon MSK 設定” にある値を取得します。設定プロパティの詳細については、「Apache Kafka の設定」を参照してください。

Apache Kafka の設定プロパティ

名前	説明
allow.everyone.if.no.acl.found	このプロパティを `false` に設定する場合は、最初に、クラスターに対して Apache Kafka ACL を定義する必要があります。最初に Apache Kafka ACL を定義せずにこのプロパティを `false` に設定すると、クラスターにアクセスできなくなります。その場合は、設定を再度更新し、このプロパティを `true` に設定することで、クラスターへのアクセスを回復できます。
auto.create.topics.enable	サーバーでトピックの自動作成を有効にします。
compression.type	特定のトピックの最終的な圧縮タイプ。このプロパティは、スタンダードの圧縮コーデック (`gzip`、`snappy`、`lz4`、および `zstd`) に設定できます。また、`uncompressed` に設定することもできます。この値は、圧縮しないことと同等です。値を `producer` に設定すると、プロデューサーが設定した元の圧縮コーデックが保持されます。
connections.max.idle.ms	アイドル接続のタイムアウト (ミリ秒単位)。サーバーソケットプロセッサスレッドは、接続のアイドル状態がこのプロパティに対して設定されている値を超えると、その接続を閉じます。
default.replication.factor	自動的に作成されるトピックのデフォルトのレプリケーション係数。

名前	説明
delete.topic.enable	トピックの削除オペレーションを有効にします。この設定をオフにすると、管理ツールからトピックを削除することができません。
group.initial.rebalance.delay.ms	グループコーディネーターが、最初の再調整を実行する前に、新しいグループに追加のデータコンシューマーが参加するのを待機する時間。遅延を長くすると再調整を減らせる可能性がありますが、処理が開始されるまでの時間が長くなります。
group.max.session.timeout.ms	登録されたコンシューマーの最大セッションタイムアウト。タイムアウトを長くすると、コンシューマーはハートビート間でより多くの時間をメッセージの処理に使用できるようになりますが、その代償として、障害検出に要する時間が長くなります。
group.min.session.timeout.ms	登録されたコンシューマーの最小セッションタイムアウト。タイムアウトを短くすると、障害検出までの時間が短縮される一方で、より頻繁にコンシューマーのハートビートが発生するようになります。これにより、ブローカーリソースが圧迫される可能性があります。
leader.imbalance.per.broker.percentage	ブローカーごとに許容されるリーダーの不均衡率。ブローカーごとにこの値を超えると、コントローラーはリーダーバランスをトリガーします。この値はパーセントで指定されます。
log.cleaner.delete.retention.ms	Apache Kafka が削除されたレコードを保持する時間。最小値は 0 です。

名前	説明
log.cleaner.min.cleanable.ratio	この設定プロパティには、0 から 1 までの値を指定できます。この値は、ログコンパクターがログのクリーニングを試行する頻度を決定します (ログ圧縮が有効になっている場合)。デフォルトでは、Apache Kafka は、ログの 50% 以上が圧縮されている場合は、そのログのクリーニングを回避します。この比率は、ログで重複によって浪費される最大スペースを制限します (50% の場合、ログの最大 50% が重複である可能性があることを意味します)。比率が高いほど、クリーニングは少なく、効率的ですが、ログ内の無駄なスペースが多くなります。
log.cleanup.policy	保存ウィンドウ外のセグメントのデフォルトのクリーンアップポリシー。カンマ区切りの有効なポリシーのリスト。有効なポリシーは `delete` および `compact` です。階層型ストレージに対応したクラスターの場合、有効なポリシーは `delete` のみです。
log.flush.interval.messages	メッセージがディスクにフラッシュされるまでにログパーティションに蓄積されるメッセージの数。
log.flush.interval.ms	トピック内のメッセージがディスクにフラッシュされるまでにメモリに保持される最大時間 (ミリ秒単位)。この値を設定しない場合、log.flush.Scheduler.interval.ms の値が使用されます。最小値は 0 です。

名前	説明
log.message.timestamp.difference.max.ms	この設定は Kafka 3.6.0 では廃止されました。`log.message.timestamp.before.max.ms` と の 2 つの設定が追加され `log.message.timestamp.after.max.ms` ました。ブローカーがメッセージを受信したときのタイムスタンプと、メッセージで指定されたタイムスタンプとの間の最大時間差。log.message.timestamp.type=CreateTime の場合、タイムスタンプの差がこのしきい値を超えると、メッセージは拒否されます。log.message.timestamp.type = LogAppendTime の場合、この設定は無視されます。
log.message.timestamp.type	メッセージ内のタイムスタンプがメッセージの作成時刻であるか、ログの追加時刻であるかを指定します。指定できる値は、`CreateTime` および `LogAppendTime` です。
log.retention.bytes	削除する前のログの最大サイズ。
log.retention.hours	log.retention.ms プロパティの 3 次である、ログファイルを削除する前に保持する時間数。
log.retention.minutes	log.retention.ms プロパティに対してセカンダリである、ログファイルを削除する前に保持する時間（分）。この値を設定しない場合、log.retention.hours の値が使用されます。
log.retention.ms	ログファイルを削除する前に保持するミリ秒数（ミリ秒単位）。設定されていない場合は、log.retention.minutes の値が使用されます。

名前	説明
log.roll.ms	新しいログセグメントがロールアウトされるまでの最大時間（ミリ秒単位）。このプロパティを設定しない場合は、log.roll.hours の値が使用されます。このプロパティの可能な最小値は 1 です。
log.segment.bytes	1 つのログファイルの最大サイズ。
max.incremental.fetch.session.cache.slots	維持される増分取得セッションの最大数。
message.max.bytes	Kafka が許容する最大レコードバッチサイズ。この値を増やし、0.10.2 より古いコンシューマーが存在する場合、コンシューマーの取得サイズも増やして、この大きさのレコードバッチを取得できるようにする必要があります。 最新のメッセージ形式バージョンでは、効率を上げるために、常にメッセージがバッチにグループ化されます。以前のメッセージ形式バージョンでは、非圧縮レコードはバッチにグループ化されません。その場合、この制限は単一のレコードにのみ適用されます。 この値は、トピックレベルの max.message.bytes 設定を使用して、トピックごとに設定できます。

名前	説明
min.insync.replicas	プロデューサーが acks を "all" (または "-1") に設定した場合、min.insync.replicas の値は、書き込みが成功したと見なされるために書き込みを確認する必要があるレプリカの最小数を指定します。この最小値を満たせない場合、プロデューサーは例外 (NotEnoughReplicas、または NotEnoughReplicasAfterAppend) を発生させます。 min.insync.replicas と acks の値を使用すると、より高い耐久性を保証できます。例えば、レプリケーション係数が 3 のトピックを作成し、min.insync.replicas を 2 に設定して、acks を "all" に設定して生成するとします。これにより、大部分のレプリカが書き込みを受け取らない場合、プロデューサーが例外を発生させることができます。
num.io.threads	サーバーがリクエストを処理するために使用するスレッドの数。これにはディスク I/O が含まれる場合があります。
num.network.threads	サーバーがネットワークからリクエストを受信し、ネットワークにレスポンスを送信するために使用するスレッドの数。
num.partitions	トピックごとのログパーティションのデフォルト数。
num.recovery.threads.per.data.dir	スタートアップ時にログを復元し、シャットダウン時にログをフラッシュするために使用されるデータディレクトリごとのスレッド数。

名前	説明
num.replica.fetchers	ソースブローカーからのメッセージのレプリケートに使用されるフェッチャースレッドの数。この値を大きくすると、フォロワーブローカーでの I/O 並列処理の度合いが高くなります。
offsets.retention.minutes	コンシューマーグループがすべてのコンシューマーを失う (つまり、空になる) と、オフセットはこの保持期間にわたって保持されてから破棄されます。スタンドアロンのコンシューマー (手動割り当てを使用するコンシューマー) の場合、オフセットは、最後のコミットの時刻にこの保持期間を足した時刻に有効期限切れになります。
offsets.topic.replication.factor	オフセットトピックのレプリケーション係数。可用性を確保するには、この値を高く設定します。内部トピックの作成は、クラスターサイズがこのレプリケーション係数の要件を満たすまで失敗します。
replica.fetch.max.bytes	パーティションごとに取得しようとするメッセージのバイト数。これは絶対最大値ではありません。フェッチの空でない最初のパーティションの最初のレコードバッチがこの値より大きい場合、進行を保証するためにそのレコードバッチが返されます。message.max.bytes (ブローカー設定) または max.message.bytes (トピック設定) は、ブローカーが受け入れる最大レコードバッチサイズを定義します。

名前	説明
replica.fetch.response.max.bytes	フェッチレスポンス全体に対して予想される最大バイト数。レコードはバッチでフェッチされます。また、フェッチの空でない最初のパーティションの最初のレコードバッチがこの値より大きい場合、進行を保証するためにそのレコードバッチが引き続き返されます。これは絶対最大値ではありません。message.max.bytes（ブローカー設定）または max.message.bytes（トピック設定）プロパティは、ブローカーが受け付ける最大レコードバッチサイズを指定します。
replica.lag.time.max.ms	フォロワーがフェッチリクエストを送信していないか、またはリーダーのログ終了オフセットまでこのミリ秒数以上消費されていない場合、リーダーはフォロワーを ISR から削除します。 MinValue: 10000 MaxValue = 30000
replica.selector.class	ReplicaSelector を実装する完全修飾クラス名。ブローカーは、この値を使用して優先リードレプリカを見つけます。Apache Kafka バージョン 2.4.1 以降を使用していて、コンシューマーが最も近いレプリカからフェッチできるようにする場合は、このプロパティを `org.apache.kafka.common.replica.RackAwareReplicaSelector` に設定します。詳細については、「the section called “Apache Kafka バージョン 2.4.1 (代わりに 2.4.1.1 を使用) ”」を参照してください。
replica.socket.receive.buffer.bytes	ネットワークリクエスト用のソケット受信バッファ。

名前	説明
socket.receive.buffer.bytes	ソケットサーバーソケットの SO_RCVBUF バッファー。このプロパティに設定できる最小値は -1 です。値が -1 の場合、Amazon MSK は OS のデフォルトを使用します。
socket.request.max.bytes	ソケットリクエストの最大バイト数。
socket.send.buffer.bytes	ソケットサーバーソケットの SO_SNDBUF バッファー。このプロパティに設定できる最小値は -1 です。値が -1 の場合、Amazon MSK は OS のデフォルトを使用します。
transaction.max.timeout.ms	トランザクションの最大タイムアウト。クライアントのリクエストしたトランザクション時間がこの値を超えると、ブローカーは InitProducerIdRequest でエラーを返します。これにより、クライアントが大きすぎるタイムアウトを設定するのを防ぎ、トランザクションに含まれるトピックから読み取りを行うコンシューマーが停止するのを回避することができます。
transaction.state.log.min.isr	トランザクショントピックに対してオーバーライドされる min.insync.replicas 設定。
transaction.state.log.replication.factor	トランザクショントピックのレプリケーション係数。このプロパティに高い値を設定すると、可用性が向上します。内部トピックの作成は、クラスターサイズがこのレプリケーション係数の要件を満たすまで失敗します。

名前	説明
transactional.id.expiration.ms	トランザクションコーディネーターが、トランザクション ID を有効期限切れにする前に、現在のトランザクションのトランザクションステータスの更新の受信を待機する時間 (ミリ秒単位)。この設定は、プロデューサー ID の有効期限切れにも影響します。これは、プロデューサー ID の有効期限は、指定されたプロデューサー ID での最後の書き込みからこの時間が経過すると切れるためです。トピックの保持設定が原因でプロデューサー ID からの最後の書き込みが削除された場合、プロデューサー ID の有効期限切れが早くなる可能性があります。このプロパティの最小値は 1 ミリ秒です。
unclean.leader.election.enable	ISR セットに含まれていないレプリカを、データ損失の可能性がある場合でも、最終手段として、リーダーとして使用するかどうかを指定します。
zookeeper.connection.timeout.ms	ZooKeeper モードクラスター。クライアントが ZooKeeper への接続を確立するのを待機する最大時間。この値を設定しない場合、zookeeper.session.timeout.ms の値が使用されます。 MinValue = 6000 MaxValue = 18000 クラスターのダウンタイムを避けるため、T3.small ではこの値を 10,000 に設定することをお勧めします。

名前	説明
zookeeper.session.timeout.ms	ZooKeeper モードクラスター。Apache ZooKeeper セッションのタイムアウト (ミリ秒単位)。 MinValue = 6000 MaxValue = 18000

カスタム MSK 設定を作成する方法、すべての設定を一覧表示する方法、または説明する方法については、「the section called “ブローカー設定オペレーション”」を参照してください。カスタム MSK 設定を使用して MSK クラスターを作成したり、新しいカスタム設定でクラスターを更新したりするには、「the section called “主な機能と概念”」を参照してください。

既存の MSK クラスターをカスタム MSK 設定で更新すると、Amazon MSK は、必要に応じてローリング再起動を実行し、顧客のダウンタイムを最小限に抑えるためのベストプラクティスを使用します。例えば、Amazon MSK は各ブローカーを再起動した後、次のブローカーに移動する前に、設定の更新中にブローカーが見逃した可能性のあるデータにブローカーがキャッチアップできるようにします。

動的 Amazon MSK 設定

Amazon MSK が提供する設定プロパティに加えて、ブローカーの再起動を必要としないクラスターレベルおよびブローカーレベルの設定プロパティを動的に設定できます。一部の設定プロパティを動的に設定できます。Apache Kafka のドキュメントの「Broker Configs」の表で、読み取り専用としてマークされていないプロパティがこれに該当します。動的設定およびコマンド例については、Apache Kafka のドキュメントの「Updating Broker Configs」を参照してください。

> **Note**
>
> `advertised.listeners` プロパティは設定できますが、`listeners` プロパティは設定できません。

トピックレベルの Amazon MSK 設定

Apache Kafka コマンドを使用して、新規および既存のトピックのトピックレベルの設定プロパティを設定するか変更することができます。トピックレベルの設定プロパティの詳細と設定方法の例については、Apache Kafka のドキュメントの「Topic-Level Configs」を参照してください。

デフォルト Amazon MSK 設定

MSK クラスターを作成し、カスタム MSK 設定を指定しない場合、Amazon MSK は、次の表に示す値を使用してデフォルト設定を作成および使用します。このテーブルにないプロパティの場合、Amazon MSK はご使用のバージョンの Apache Kafka に関連付けられているデフォルトを使用します。これらのデフォルト値のリストについては、Apache Kafka の設定を参照してください。

デフォルトの設定値

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
allow.everyone.if.no.acl.found	特定のリソースに一致するリソースパターンがない場合、リソースには ACL が関連付けられていません。この場合、このプロパティを `true` に設定すると、スーパーユーザーだけでなく、すべてのユーザーがリソースにアクセスできます。	`true`	`true`
auto.create.topics.enable	サーバー上のトピックの自動作成を有効にします。	`false`	`false`
auto.leader.rebalance.enable	自動リーダーバランシングを有効にします。バックグラウンドスレッドは、必要	`true`	`true`

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
	に応じて一定の間隔でリーダーバランスをチェックして開始します。		
default.replication.factor	自動的に作成されるトピックのデフォルトのレプリケーション係数。	3 つのアベイラビリティーゾーン内のクラスターの場合は 3、2 つのアベイラビリティーゾーン内のクラスターの場合は 2。	3 つのアベイラビリティーゾーン内のクラスターの場合は 3、2 つのアベイラビリティーゾーン内のクラスターの場合は 2。

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
local.retention.bytes	古いセグメントを削除する前の、パーティションのローカルログセグメントの最大サイズ。この値を設定しない場合、log.retention.bytes の値が使用されます。有効な値は、常に log.retention.bytes の値以下にする必要があります。デフォルト値 -2 は、ローカル保持に制限がないことを示します。これは、retention.ms/bytes 設定の -1 に相当します。local.retention.ms プロパティと local.retention.bytes プロパティは、ログセグメントをローカルストレージに保持する期間を決定するために使用されるという点で log.retention に似ています。既存の log.retention.* 設定は、トピックパーティションの保持設定です。これには、ローカルストレー	-2 は制限なし	-2 は制限なし

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
	ジとリモートストレージの両方が含まれます。有効な値: [-2; +Inf] の整数		

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
local.retention.ms	削除するまでローカルログセグメントを保持するミリ秒数。この値を設定しない場合、Amazon MSK は log.retention.ms の値を使用します。有効な値は、常に log.retention.bytes の値以下にする必要があります。デフォルト値 -2 は、ローカル保持に制限がないことを示します。これは、retention.ms/bytes 設定の -1 に相当します。local.retention.ms と local.retention.bytes の値は log.retention と似ています。MSK はこの設定を使用して、ログセグメントをローカルストレージに保持する期間を決定します。既存の log.retention.* 設定は、トピックパーティションの保持設定です。これには、ローカルストレージとリモートストレー	-2 は制限なし	-2 は制限なし

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
	ジの両方が含まれます。有効な値は 0 より大きい整数です。		

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
log.message.timestamp.difference.max.ms	この設定は Kafka 3.6.0 では廃止されました。`log.message.timestamp.before.max.ms` と の 2 つの設定が追加されlog.message.timestamp.after.max.ms ました。ブローカーがメッセージを受信したときのタイムスタンプと、メッセージで指定されたタイムスタンプとの間に許容される最大差です。log.message.timestamp.type=CreateTime の場合、タイムスタンプの差がこのしきい値を超えると、メッセージは拒否されます。log.message.timestamp.type = LogAppendTime の場合、この設定は無視されます。不必要な頻度でのログローリングを避けるために、許容されるタイ	9223372036854775807	Kafka 2.8.2.tiered および Kafka 3.7.x tiered の場合は 86400000。

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
	ムスタンプの最大差は log.retention.ms 以下にする必要があります。		
log.segment.bytes	1 つのログファイルの最大サイズ。	1073741824	134217728

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
min.insync.replicas	プロデューサーが acks の値を "all" (または "-1") に設定した場合、min.insync.replicas の値は、書き込みが成功したと見なされるために書き込みを確認する必要があるレプリカの最小数を指定します。この値がこの最小値を満たしていない場合，プロデューサーは例外 (NotEnoughReplicas または NotEnough ReplicasAfterAppend) を発生させます。 min.insync.replicas と acks の値を組み合わせて使用すると、より高い耐久性を保証できます。例えば、レプリケーション係数が 3 のトピックを作成し、min.insync.replicas を 2 に設定して、acks を "all" に設定して生成するとします。これにより、大部分のレプリカが書き込	3 つのアベイラビリティーゾーン内のクラスターの場合は 2、2 つのアベイラビリティーゾーン内のクラスターの場合は 1。	3 つのアベイラビリティーゾーン内のクラスターの場合は 2、2 つのアベイラビリティーゾーン内のクラスターの場合は 1。

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
	みを受け取らない場合、プロデューサーが例外を発生させることができます。		
num.io.threads	サーバーがリクエストを生成するために使用するスレッドの数。これにはディスク I/O が含まれる場合があります。	8	max(8, vCPUs)。vCPUs はブローカーのインスタンスサイズによって異なります
num.network.threads	サーバーがネットワークからリクエストを受信し、ネットワークにレスポンスを送信するために使用するスレッドの数。	5	max(5, vCPUs / 2)。vCPUs はブローカーのインスタンスサイズによって異なります
num.partitions	トピックごとのログパーティションのデフォルト数。	1	1
num.replica.fetchers	ソースブローカーからのメッセージをレプリケートするために使用されるフェッチャースレッドの数。この値を大きくすると、フォロワーブローカーの I/O 並列処理の度合いを上げることができます。	2	max(2, vCPUs / 4)。vCPUs はブローカーのインスタンスサイズによって異なります

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
remote.log.msk.disable.policy	階層型ストレージを無効にする場合は、remote.storage.enable と一緒に使用します。remote.storage.enable を false に設定した場合に階層型ストレージ内のデータが削除されるようにするには、このポリシーを Delete に設定します。	該当なし	なし
remote.log.reader.threads	リモートログリーダーのスレッドプールサイズ。リモートストレージからデータを取得するタスクをスケジュールする際に使用されます。	該当なし	max(10, vCPUs * 0.67)。vCPUs はブローカーのインスタンスサイズによって異なります

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
remote.storage.enable	これを true に設定すると、トピックの階層型 (リモート) ストレージが有効になります。これを false に設定し、remote.log.msk.disable.policy が Delete に設定されている場合、トピックレベルの階層型ストレージが無効になります。階層型ストレージを無効にすると、リモートストレージからデータが削除されます。トピックの階層型ストレージを無効にすると、再度有効にすることはできません。	false	false
replica.lag.time.max.ms	フォロワーがフェッチリクエストを送信していないか、またはリーダーのログ終了オフセットまでこのミリ秒数以上消費されていない場合、リーダーはフォロワーを ISR から削除します。	30000	30000

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
retention.ms	必須フィールド。最短期間は 3 日間です。設定は必須であるため、デフォルトはありません。 Amazon MSK は retention.ms 値と local.retention.ms を使用して、データがローカルから階層型ストレージに移動するタイミングを決定します。local.retention.ms 値は、データをローカルから階層型ストレージに移動するタイミングを指定します。retention.ms 値は、階層型ストレージからデータを削除する (つまり、クラスターから削除する) タイミングを指定します。有効な値: [-1; +Inf] の整数	最小 259,200,000 ミリ秒 (3 日間)。無期限に保持する場合は -1。	最小 259,200,000 ミリ秒 (3 日間)。無期限に保持する場合は -1。
socket.receive.buffer.bytes	ソケットサーバーソケットの SO_RCVBUF バッファー。値が -1 の場合、OS のデフォルトが使用されます。	102400	102400

名前	説明	非階層型ストレージクラスターのデフォルト値	階層型ストレージ対応クラスターのデフォルト値
socket.request.max.bytes	ソケットリクエストの最大バイト数。	104857600	104857600
socket.send.buffer.bytes	ソケットサーバーソケットの SO_SNDBUF バッファー。値が -1 の場合、OS のデフォルトが使用されます。	102400	102400
unclean.leader.election.enable	ISR セットに含まれていないレプリカを、データ損失の可能性がある場合でも、最終手段として、リーダーとして使用するかどうかを指定します。	true	false
zookeeper.session.timeout.ms	Apache ZooKeeper セッションのタイムアウト (ミリ秒単位)。	18000	18000
zookeeper.set.acl	セキュア ACL を使用するようにクライアントを設定します。	false	false

カスタム設定値の指定方法については、「the section called “カスタム Amazon MSK 設定”」を参照してください。

Amazon MSK 階層型ストレージのトピックレベル設定に関するガイドライン

階層型ストレージをトピックレベルで設定する場合のデフォルト設定と制限は次のとおりです。

- Amazon MSK では、階層型ストレージが有効になっているトピックのログセグメントサイズを小さくすることはできません。セグメントを作成する場合、最小ログセグメントサイズは 48 MiB、または最小セグメントロール時間は 10 分です。これらの値は segment.bytes プロパティと segment.ms プロパティにマッピングされます。
- local.retention.ms/bytes の値は、retention.ms/bytes より小さくする必要があります。これは階層型ストレージの保持設定です。
- local.retention.ms/bytes のデフォルト値は -2 です。つまり、retention.ms の値が local.retention.ms/bytes に使用されます。この場合、データはローカルストレージと階層型ストレージの両方に残り (それぞれ 1 つのコピー)、同時に有効期限切れになります。このオプションでは、ローカルデータのコピーがリモートストレージに保持されます。この場合、消費トラフィックから読み取られるデータはローカルストレージから取得されます。
- retention.ms のデフォルト値は 7 日間です。retention.bytes には、デフォルトサイズの制限はありません。
- retention.ms/bytes の最小値は -1 です。これは、無期限に保持することを意味します。
- local.retention.ms/bytes の最小値は -2 です。これは、ローカルストレージでは無期限に保持されることを意味します。これは、retention.ms/bytes を -1 に設定するのと同じです。
- 階層型ストレージが有効になっているトピックには、トピックレベル設定の retention.ms が必須です。retention.ms の最小値は 3 日間です。

階層型ストレージの制約の詳細については、「」を参照してくださいAmazon MSK クラスターの階層型ストレージの制約と制限。

Express ブローカーの設定

Apache Kafka には、MSK プロビジョンドクラスターのパフォーマンスを調整するために使用できる数百のブローカー設定があります。誤った値または最適でない値を設定すると、クラスターの信頼性とパフォーマンスに影響する可能性があります。Express ブローカーは、重要な設定に最適な値を設定し、一般的な設定ミスから保護することで、MSK プロビジョンドクラスターの可用性と耐久性を向上させます。読み取りおよび書き込みアクセスに基づく設定には、読み取り/書き込み (編集可能)、読み取り専用、および読み取り/書き込み以外の設定の 3 つのカテゴリがあります。一部の設定では、クラスターが実行されている Apache Kafka バージョンの Apache Kafka のデフォルト値が引き続き使用されます。これらを Apache Kafka Default としてマークします。

トピック

- カスタム MSK Express ブローカー設定 (読み取り/書き込みアクセス)
- Express ブローカーの読み取り専用設定

カスタム MSK Express ブローカー設定 (読み取り/書き込みアクセス)

Amazon MSK の更新設定機能を使用するか、Apache Kafka の AlterConfig API を使用して、読み取り/書き込みブローカー設定を更新できます。 AlterConfig Apache Kafka ブローカー設定は静的または動的です。静的設定を適用するにはブローカーの再起動が必要ですが、動的設定にはブローカーの再起動は必要ありません。設定プロパティと更新モードの詳細については、「ブローカー設定の更新」を参照してください。

トピック

- MSK Express ブローカーの静的設定
- Express ブローカーの動的設定
- Express ブローカーのトピックレベルの設定

MSK Express ブローカーの静的設定

Amazon MSK を使用してカスタム MSK 設定ファイルを作成し、次の静的プロパティを設定できます。Amazon MSK は、設定していない他のすべてのプロパティを設定および管理します。静的設定ファイルは、MSK コンソールから、または configuration コマンドを使用して作成および更新できます。

プロパティ	説明	デフォルト値
allow.everyone.if.no.acl.found	このプロパティを false に設定する場合は、最初にクラスターに Apache Kafka ACL を定義していることを確認してください。このプロパティを false に設定し、最初に Apache Kafka ACLs を定義しない場合、クラスターにアクセスできなくなります。その場合は、設定を再度更新し、このプロパティを true に設定して、クラスターへのアクセスを回復できます。	true

プロパティ	説明	デフォルト値
auto.create.topics.enable	サーバー上のトピックの自動作成を有効にします。	false
compression.type	特定のトピックの最終圧縮タイプを指定します。この設定では、gzip、snapy、lz4、zstd という標準の圧縮コーデックを使用できます。 この設定では、圧縮なしuncompressed に相当する とproducer、プロデューサーによって設定された元の圧縮コーデックを保持する も受け入れます。	Apache Kafka のデフォルト
connections.max.idle.ms	アイドル接続のタイムアウト (ミリ秒単位)。サーバーソケットプロセッサスレッドは、接続のアイドル状態がこのプロパティに対して設定されている値を超えると、その接続を閉じます。	Apache Kafka のデフォルト
delete.topic.enable	トピックの削除オペレーションを有効にします。この設定をオフにすると、管理ツールからトピックを削除することができません。	Apache Kafka のデフォルト

プロパティ	説明	デフォルト値
group.initial.rebalance.delay.ms	グループコーディネーターが、最初の再調整を実行する前に、新しいグループに追加のデータコンシューマーが参加するのを待機する時間。遅延を長くすると再調整を減らせる可能性がありますが、処理が開始されるまでの時間が長くなります。	Apache Kafka のデフォルト
group.max.session.timeout.ms	登録されたコンシューマーの最大セッションタイムアウト。タイムアウトを長くすると、コンシューマーはハートビート間でより多くの時間をメッセージの処理に使用できるようになりますが、その代償として、障害検出に要する時間が長くなります。	Apache Kafka のデフォルト
leader.imbalance.per.broker.percentage	ブローカーごとに許容されるリーダーの不均衡率。ブローカーごとにこの値を超えると、コントローラーはリーダーバランスをトリガーします。この値はパーセントで指定されます。	Apache Kafka のデフォルト

プロパティ	説明	デフォルト値
log.cleanup.policy	保存ウィンドウ外のセグメントのデフォルトのクリーンアップポリシー。カンマ区切りの有効なポリシーのリスト。有効なポリシーは delete および compact です。階層型ストレージ対応クラスターの場合、有効なポリシーは deleteのみです。	Apache Kafka のデフォルト
log.message.timestamp.after.max.ms	メッセージタイムスタンプとブローカーのタイムスタンプの間の許容されるタイムスタンプの差。メッセージタイムスタンプはブローカーのタイムスタンプ以降で、この設定で設定された値によって最大許容差が決まります。 の場合log.message.timestamp.type=CreateTime 、タイムスタンプの差がこの指定されたしきい値を超えると、メッセージは拒否されます。この設定は、 の場合無視されますlog.message.timestamp.type=LogAppendTime 。	86400000 (24 * 60 * 60 * 1000 ms、つまり 1 日)

プロパティ	説明	デフォルト値
log.message.timestamp.before.max.ms	ブローカーのタイムスタンプとメッセージタイムスタンプの許容されるタイムスタンプの差。メッセージタイムスタンプはブローカーのタイムスタンプ以前とすることができ、最大許容差はこの設定で設定された値によって決まります。 の場合`log.message.timestamp.type=CreateTime` 、タイムスタンプの差がこの指定されたしきい値を超えると、メッセージは拒否されます。この設定は、 の場合無視されます`log.message.timestamp.type=LogAppendTime` 。	86400000 (24 * 60 * 60 * 1000 ms、つまり 1 日)
log.message.timestamp.type	メッセージ内のタイムスタンプがメッセージの作成時刻であるか、ログの追加時刻であるかを指定します。指定できる値は、`CreateTime` および `LogAppendTime` です。	Apache Kafka のデフォルト
log.retention.bytes	削除する前のログの最大サイズ。	Apache Kafka のデフォルト
log.retention.ms	ログファイルを削除する前に保持するミリ秒数。	Apache Kafka のデフォルト

プロパティ	説明	デフォルト値
max.connections.per.ip	各 IP アドレスから許可される接続の最大数。これは、`max.connections.per.ip.overrides` プロパティを使用してオーバーライド`0`が設定されている場合にに設定できます。制限に達すると、IP アドレスからの新しい接続は削除されます。	Apache Kafka のデフォルト
max.incremental.fetch.session.cache.slots	維持される増分取得セッションの最大数。	Apache Kafka のデフォルト
message.max.bytes	Kafka が許容する最大レコードバッチサイズ。この値を増やし、0.10.2 より古いコンシューマーが存在する場合、コンシューマーの取得サイズも増やして、この大きさのレコードバッチを取得できるようにする必要があります。 最新のメッセージ形式バージョンでは、効率を上げるために、常にメッセージがバッチにグループ化されます。以前のメッセージ形式バージョンでは、非圧縮レコードはバッチにグループ化されません。その場合、この制限は単一のレコードにのみ適用されます。トピックレベル`max.message.bytes` 設定を使用して、トピックごとにこの値を設定できます。	Apache Kafka のデフォルト

プロパティ	説明	デフォルト値
num.partitions	トピックあたりのパーティションのデフォルト数。	1
offsets.retention.minutes	コンシューマーグループがすべてのコンシューマーを失う（つまり、空になる）と、オフセットはこの保持期間にわたって保持されてから破棄されます。スタンドアロンコンシューマー (手動割り当てを使用するコンシューマー) の場合、オフセットは最後のコミットの時刻にこの保持期間を加えた後に期限切れになります。	Apache Kafka のデフォルト
replica.fetch.max.bytes	パーティションごとに取得しようとするメッセージのバイト数。これは絶対最大値ではありません。フェッチの空でない最初のパーティションの最初のレコードバッチがこの値より大きい場合、進行を保証するためにそのレコードバッチが返されます。message.max.bytes (ブローカー設定) または max.message.bytes (トピック設定) は、ブローカーが受け入れる最大レコードバッチサイズを定義します。	Apache Kafka のデフォルト

プロパティ	説明	デフォルト値
replica.selector.class	ReplicaSelector を実装する完全修飾クラス名。ブローカーは、この値を使用して優先リードレプリカを見つけます。コンシューマーが最も近いレプリカからフェッチできるようにするには、このプロパティを に設定しますorg.apache.kafka.common.replica.RackAwareReplicaSelector 。	Apache Kafka のデフォルト
socket.receive.buffer.bytes	ソケットサーバーソケットの SO_RCVBUF バッファー。値が -1 の場合、OS のデフォルトが使用されます。	102400
socket.request.max.bytes	ソケットリクエストの最大バイト数。	104857600
socket.send.buffer.bytes	ソケットサーバーソケットの SO_SNDBUF バッファー。値が -1 の場合、OS のデフォルトが使用されます。	102400

プロパティ	説明	デフォルト値
transaction.max.timeout.ms	トランザクションの最大タイムアウト。クライアントのリクエストしたトランザクション時間がこの値を超えると、ブローカーは InitProducerIdRequest でエラーを返します。これにより、クライアントが大きすぎるタイムアウトを設定するのを防ぎ、トランザクションに含まれるトピックから読み取りを行うコンシューマーが停止するのを回避することができます。	Apache Kafka のデフォルト
transactional.id.expiration.ms	トランザクションコーディネーターが、トランザクション ID を有効期限切れにする前に、現在のトランザクションのトランザクションステータスの更新の受信を待機する時間 (ミリ秒単位)。この設定は、指定されたプロデューサー ID での最後の書き込み後にこの時間が経過すると、プロデューサー IDs の有効期限が切れるため、プロデューサー ID の有効期限にも影響します。トピックの保持設定が原因でプロデューサー ID からの最後の書き込みが削除された場合、プロデューサー ID の有効期限切れが早くなる可能性があります。このプロパティの最小値は 1 ミリ秒です。	Apache Kafka のデフォルト

Express ブローカーの動的設定

Apache Kafka AlterConfig API または Kafka-configs.sh ツールを使用して、次の動的設定を編集できます。Amazon MSK は、設定していない他のすべてのプロパティを設定および管理します。ブローカーの再起動を必要としないクラスターレベルおよびブローカーレベルの設定プロパティを動的に設定できます。

プロパティ	説明	デフォルト値
advertised.listeners	`listeners` 設定プロパティと異なる場合、クライアントが使用するために公開するリスナー。IaaS 環境では、これはブローカーがバインドするインターフェイスとは異なる必要がある場合があります。これを設定しない場合、リスナーの値が使用されます。リスナーとは異なり、0.0.0.0 メタアドレスをアドバタイズすることは有効ではありません。 また、 と は異なり`listeners` 、このプロパ	null

プロパティ	説明	デフォルト値
	ティにはポートが重複している可能性があるため、あるリスナーが別のリスナーのアドレスをアドバタイズするように設定できます。これは、外部ロードバランサーを使用する場合に便利です。 このプロパティはブローカーごとに設定されます。	

プロパティ	説明	デフォルト値
compression.type	特定のトピックの最終的な圧縮タイプ。このプロパティは、スタンダードの圧縮コーデック (gzip、snappy、および zstd) に設定できます。また、uncompressed に設定することもできます。この値は、圧縮しないことと同等です。値を producer に設定すると、プロデューサーが設定した元の圧縮コーデックが保持されます。	Apache Kafka のデフォルト

プロパティ	説明	デフォルト値
log.cleaner.delete.retention.ms	ログ圧縮トピックの削除墓石マーカーを保持する時間。この設定は、最終ステージの有効なスナップショットを取得するために、コンシューマーがオフセット 0 から開始した場合に読み取りを完了する必要がある時間にも制限を与えます。そうしないと、スキャンを完了する前に削除墓石が収集される場合があります。	86400000 (24 * 60 * 60 * 1000 ms、つまり 1 日）、Apache Kafka デフォルト
log.cleaner.min.compaction.lag.ms	メッセージがログに圧縮されない最小時間。この設定は、圧縮されているログにのみ適用されます。	0、Apache Kafka デフォルト

プロパティ	説明	デフォルト値
log.cleaner.max.compaction.lag.ms	メッセージがログに圧縮されない最大時間。この設定は、圧縮されているログにのみ適用されます。この設定は、[7 days, Long.Max] の範囲に制限されます。	9223372036854775807、Apache Kafka デフォルト
log.cleanup.policy	保存ウィンドウ外のセグメントのデフォルトのクリーンアッププポリシー。カンマ区切りの有効なポリシーのリスト。有効なポリシーは delete および compact です。階層型ストレージ対応クラスターの場合、有効なポリシーは deleteのみです。	Apache Kafka のデフォルト

プロパティ	説明	デフォルト値
log.message.timestamp.after.max.ms	メッセージタイムスタンプとブローカーのタイムスタンプの間の許容されるタイムスタンプの差。メッセージタイムスタンプはブローカーのタイムスタンプ以降で、この設定で設定された値によって最大許容差が決まります。の場合`log.message.timestamp.type=CreateTime` 、タイムスタンプの差がこの指定されたしきい値を超えると、メッセージは拒否されます。この設定は、 の場合無視されます`log.message.timestamp.type=LogAppendTime` 。	86400000 (24 * 60 * 60 * 1000 ms、つまり 1 日)

プロパティ	説明	デフォルト値
log.message.timestamp.before.max.ms	ブローカーのタイムスタンプとメッセージタイムスタンプの許容されるタイムスタンプの差。メッセージタイムスタンプはブローカーのタイムスタンプ以前とすることができ、最大許容差はこの設定で設定された値によって決まります。の場合`log.message.timestamp.type=CreateTime` 、タイムスタンプの差がこの指定されたしきい値を超えると、メッセージは拒否されます。この設定は、 の場合無視されます`log.message.timestamp.type=LogAppendTime` 。	86400000 (24 * 60 * 60 * 1000 ms、つまり 1 日)

プロパティ	説明	デフォルト値
log.message.timestamp.type	メッセージ内のタイムスタンプがメッセージの作成時刻であるか、ログの追加時刻であるかを指定します。指定できる値は、`CreateTime` および `LogAppendTime` です。	Apache Kafka のデフォルト
log.retention.bytes	削除する前のログの最大サイズ。	Apache Kafka のデフォルト
log.retention.ms	ログファイルを削除する前に保持するミリ秒数。	Apache Kafka のデフォルト
max.connection.creation.rate	ブローカーでいつでも許可される最大接続作成レート。	Apache Kafka のデフォルト

プロパティ	説明	デフォルト値
最大接続数	ブローカーで許可される接続の最大数。この制限は、 を使用して設定された IP ごとの制限に加えて適用されますmax.connections.per.ip 。	Apache Kafka のデフォルト
max.connections.per.ip	各 IP アドレスから許可される接続の最大数。これは、max.connections.per.ip.overrides プロパティを使用して0オーバーライドが設定されている場合にに設定できます。制限に達すると、IP アドレスからの新しい接続は削除されます。	Apache Kafka のデフォルト

プロパティ	説明	デフォルト値
max.connections.per.ip.overrides	ip 単位またはホスト名のカンマ区切りリストは、デフォルトの最大接続数に上書きされます。値の例は です。`hostName:100,127.0.0.1:200`	Apache Kafka のデフォルト

プロパティ	説明	デフォルト値
message.max.bytes	Kafka が許容する最大レコードバッチサイズ。この値を増やし、0.10.2 より古いコンシューマーが存在する場合、コンシューマーの取得サイズも増やして、この大きさのレコードバッチを取得できるようにする必要があります。最新のメッセージ形式バージョンでは、効率を上げるために、常にメッセージがバッチにグループ化されます。以前のメッセージ形式バージョンでは、非圧縮レコードはバッチにグループ化されません。その場合、この制限は単一のレコードにのみ適用されます。トピックレベ	Apache Kafka のデフォルト

プロパティ	説明	デフォルト値
	ルmax.message.bytes 設定を使用して、トピックごとにこの値を設定できます。	

プロパティ	説明	デフォルト値
producer.id.expiration.ms	トピックパーティションリーダーがプロデューサー IDs の有効期限が切れるまでに待機するミリ秒単位の時間。プロデューサー IDs は、それらに関連付けられたトランザクションがまだ進行中の間は期限切れになりません。トピックの保持設定が原因でプロデューサー IDs からの最後の書き込みが削除された場合、プロデューサー ID の有効期限が早まる可能性があることに注意してください。この値を 以上に設定すると、再試行中の期限切れを防ぎ、メッセージの重複を防ぐdelivery.timeout.m	Apache Kafka のデフォルト

プロパティ	説明	デフォルト値
	s ことができますが、ほとんどのユースケースではデフォルトが合理的である必要があります。	

Express ブローカーのトピックレベルの設定

Apache Kafka コマンドを使用して、新規および既存のトピックのトピックレベルの設定プロパティを設定するか変更することができます。トピックレベルの設定を指定できない場合、Amazon MSK はブローカーのデフォルトを使用します。ブローカーレベルの設定と同様に、Amazon MSK はトピックレベルの設定プロパティの一部を変更から保護します。例としては、レプリケーション係数、、 `min.insync.replicas`などがありますunclean.leader.election.enable。以外のレプリケーション係数値を持つトピックを作成しようとすると3、Amazon MSK は3デフォルトで のレプリケーション係数を持つトピックを作成します。トピックレベルの設定プロパティの詳細と設定方法の例については、Apache Kafka のドキュメントの「Topic-Level Configs」を参照してください。

プロパティ	説明
cleanup.policy	この設定は、ログセグメントで使用する保持ポリシーを指定します。保持時間またはサイズ制限に達すると、「削除」ポリシー (デフォルト) は古いセグメントを破棄します。「コンパクト」ポリシーは、各キーの最新の値を保持するログ圧縮を有効にします。また、両方のポリシーをカンマ区切りリスト (「delete, compact」など) で指定することもできます。この場合、古いセグメントは保持時間とサイズ設定ごとに破棄され、保持されたセグメントは圧縮されます。Express ブローカーでの圧縮は、パーティション内のデータが 256 MB に達した後にトリガーされます。

プロパティ	説明
compression.type	特定のトピックの最終圧縮タイプを指定します。この設定は、標準の圧縮コーデック (gzip、snappy、lz4、) を受け入れますzstd。さらに、圧縮なしに相当する uncompressed を受け入れます。producerつまり、プロデューサーによって設定された元の圧縮コーデックを保持します。
delete.retention.ms	ログ圧縮トピックの削除墓石マーカーを保持する時間。この設定は、最終ステージの有効なスナップショットを取得するために、コンシューマーがオフセット 0 から開始した場合に読み取りを完了する必要がある時間にも制限を与えます。そうしないと、スキャンを完了する前に削除墓石が収集される場合があります。 この設定のデフォルト値は 86400000 (24 * 60 * 60 * 1000 ms、つまり 1 日)、Apache Kafka Default
max.message.bytes	Kafka で許可される最大レコードバッチサイズ (圧縮が有効な場合は圧縮後)。これが増加し、 より古いコンシューマーがある場合0.10.2、コンシューマーのフェッチサイズも増加して、この大きなレコードバッチを取得できるようにする必要があります。最新のメッセージ形式バージョンでは、効率を上げるために、レコードは常にバッチにグループ化されます。以前のメッセージ形式バージョンでは、圧縮されていないレコードはバッチにグループ化されず、この制限は単一のレコードにのみ適用されます。これは、トピックレベル でトピックごとに設定できますmax.message.bytes config。

プロパティ	説明
message.timestamp.after.max.ms	この設定では、メッセージタイムスタンプとブローカーのタイムスタンプの許容タイムスタンプ差を設定します。メッセージタイムスタンプはブローカーのタイムスタンプ以降で、この設定で設定された値によって最大許容差が決まります。の場合message.timestamp.type=CreateTime 、タイムスタンプの差がこの指定されたしきい値を超えると、メッセージは拒否されます。この設定は、 の場合無視されますmessage.timestamp.type=LogAppendTime 。
message.timestamp.before.max.ms	この設定では、ブローカーのタイムスタンプとメッセージタイムスタンプの許容されるタイムスタンプの差を設定します。メッセージタイムスタンプはブローカーのタイムスタンプ以前とすることができ、最大許容差はこの設定で設定された値によって決まります。の場合message.timestamp.type=CreateTime 、タイムスタンプの差がこの指定されたしきい値を超えると、メッセージは拒否されます。この設定は、 の場合無視されますmessage.timestamp.type=LogAppendTime 。
message.timestamp.type	メッセージ内のタイムスタンプがメッセージ作成時刻かログ追加時刻かを定義します。値は CreateTime または のいずれかである必要があります。 LogAppendTime

プロパティ	説明
min.compaction.lag.ms	メッセージがログに圧縮されない最小時間。この設定は、圧縮されているログにのみ適用されます。 この設定のデフォルト値は 0、Apache Kafka Default
max.compaction.lag.ms	メッセージがログに圧縮されない最大時間。この設定は、圧縮されているログにのみ適用されます。この設定は、[7 days, Long.Max] の範囲に制限されます。 この設定のデフォルト値は 9223372036854775807、Apache Kafka Default です。
retention.bytes	この設定は、「削除」保持ポリシーを使用している場合に、古いログセグメントを破棄して領域を解放する前に、パーティション (ログセグメントで構成される) を拡張できる最大サイズを制御します。デフォルトでは、サイズ制限はなく、時間制限のみです。この制限はパーティションレベルで適用されるため、パーティションの数を乗算してトピックの保持をバイト単位で計算します。さらに、 は segment.ms および segment.bytes 設定とは独立してretention.bytes configuration 動作します。さらに、 がゼロに設定されている場合、新しいセグメントのローリングretention.bytes がトリガーされます。

プロパティ	説明
retention.ms	この設定は、「削除」保持ポリシーを使用している場合に、古いログセグメントを破棄して領域を解放する前に、ログを保持する最大時間を制御します。これは、コンシューマーがデータを読み取るまでの時間に関する SLA を表します。に設定すると-1、時間制限は適用されません。さらに、retention.ms 設定は segment.ms および segment.bytes 設定とは独立して動作します。さらに、retention.ms 条件が満たされると、新しいセグメントのローリングがトリガーされます。

Express ブローカーの読み取り専用設定

Amazon MSK は、これらの設定の値を設定し、クラスターの可用性に影響を与える可能性のある変更から保護します。これらの値は、クラスターで実行されている Apache Kafka バージョンによって変わる可能性があるため、特定のクラスターの値を必ず確認してください。次に例を示します。

Express ブローカーの読み取り専用設定

プロパティ	説明	Express ブローカー値
broker.id	このサーバーのブローカー ID。	1、2、3...
broker.rack	ブローカーのラック。これは耐障害性のためにラック対応レプリケーションの割り当てに使用されます。例: `RACK1`、`us-east-1d`	AZ ID またはサブネット ID
default.replication.factor	すべてのトピックのデフォルトのレプリケーション係数。	3
fetch.max.bytes	フェッチリクエストに対して返される最大バイト数。	Apache Kafka のデフォルト

プロパティ	説明	Express ブローカー値
group.max.size	1 つのコンシューマーグループが対応できるコンシューマーの最大数。	Apache Kafka のデフォルト
inter.broker.listener.name	ブローカー間の通信に使用されるリスナーの名前。	REPLICATION_SECURE または REPLICATION
inter.broker.protocol.version	使用するブローカー間プロトコルのバージョンを指定します。	Apache Kafka のデフォルト
リスナー	リスナーリスト - リッスンする URIs とリスナー名のカンマ区切りリスト。は設定できますが`advertised.listeners property 、listeners` プロパティは設定できません。	MSK 生成
log.message.format.version	ブローカーがログにメッセージを追加するために使用するメッセージ形式バージョンを指定します。	Apache Kafka のデフォルト

プロパティ	説明	Express ブローカー値
min.insync.replicas	プロデューサーが acks を `all` (または `-1`) に設定すると、の値は、書き込みが成功したと見なされるために書き込みを承認する必要があるレプリカの最小数`min.insync.replicas` を指定します。この最小値を満たせない場合、プロデューサーは例外 (`NotEnoughReplicas` または) を発生させます`NotEnoughReplicasAfterAppend` 。 プロデューサーの ack の値を使用して、耐久性の保証を強化できます。acks を「all」に設定します。これにより、大部分のレプリカが書き込みを受け取らない場合、プロデューサーが例外を発生させることができます。	2
num.io.threads	サーバーがリクエストの生成に使用するスレッドの数。これには、ディスク I/O (m7g.large, 8)、 (m7g.xlarge, 8)、 (m7g.2xlarge, 16)、 (m7g.4xlarge, 32)、 (m7g.8xlarge, 64)、 (m7g.12xlarge, 96)、 (m7g.16xlarge, 128) が含まれます。	インスタンスタイプに基づきます。=Math.max(8、2 * vCPUs)

プロパティ	説明	Express ブローカー値
num.network.threads	サーバーがネットワークからリクエストを受信し、ネットワークにレスポンスを送信するために使用するスレッドの数。 (m7g.large, 8)、 (m7g.xlarge, 8)、 (m7g.2xlarge, 8)、 (m7g.4xlarge, 16)、 (m7g.8xlarge, 32)、 (m7g.12xlarge, 48)、 (m7g.16xlarge, 64)	インスタンスタイプに基づきます。=Math.max(8、vCPUs)
replica.fetch.response.max.bytes	フェッチレスポンス全体に対して予想される最大バイト数。レコードはバッチでフェッチされます。また、フェッチの空でない最初のパーティションの最初のレコードバッチがこの値より大きい場合、進行を保証するためにそのレコードバッチが引き続き返されます。これは絶対最大値ではありません。`message.max.bytes` (ブローカー設定) または `max.message.bytes` (トピック設定) プロパティは、ブローカーが受け入れる最大レコードバッチサイズを指定します。	Apache Kafka のデフォルト

プロパティ	説明	Express ブローカー値
request.timeout.ms	設定は、クライアントがリクエストの応答を待機する最大時間を制御します。タイムアウトが経過する前にレスポンスが受信されない場合、クライアントは必要に応じてリクエストを再送信するか、再試行が枯渇した場合はリクエストに失敗します。	Apache Kafka のデフォルト
transaction.state.log.min.isr	トランザクショントピック`min.insync.replicas` の設定が上書きされました。	2
transaction.state.log.replication.factor	トランザクショントピックのレプリケーション係数。	Apache Kafka のデフォルト
unclean.leader.election.enable	ISR セットにないレプリカが、データ損失につながる可能性がある場合でも、最後の手段としてリーダーとして機能することを許可します。	誤

ブローカー設定オペレーション

Apache Kafka ブローカー設定は静的または動的です。静的設定を適用するには、ブローカーを再起動する必要があります。動的設定を更新するためにブローカーを再起動する必要はありません。設定プロパティと更新モードの詳細については、「Apache Kafka 設定」を参照してください。

このトピックでは、カスタム MSK 設定を作成する方法と、それらの設定に対してオペレーションを実行する方法について説明します。MSK 設定を使用してクラスターを作成または更新する方法については、「the section called “主な機能と概念”」を参照してください。

トピック

- 設定を作成する

- 設定の更新
- 設定の削除
- 設定メタデータを取得する
- 設定リビジョンの詳細を取得する
- 現在のリージョンのアカウントの設定を一覧表示する
- Amazon MSK 設定状態

設定を作成する

このプロセスでは、カスタム Amazon MSK 設定を作成する方法と、その設定に対してオペレーションを実行する方法について説明します。

1. 設定する設定プロパティと、それらに割り当てる値を指定するファイルを作成します。次に、設定ファイルの例を示します。

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

log.roll.ms = 604800000
```

2. 次の AWS CLI コマンドを実行し、*config-file-path* を、前のステップで設定を保存したファイルへのパスに置き換えます。

> **Note**
>
> 設定に選択する名前は、次の正規表現「^[0-9A-Za-z][0-9A-Za-z-]{0,}$」と一致する必要があります。

```
aws kafka create-configuration --name "ExampleConfigurationName" --description
 "Example configuration description." --kafka-versions "1.1.1" --server-properties
 fileb://config-file-path
```

このコマンドを実行した後の正常なレスポンスの例を以下に示します。

```
{
    "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/
abcdabcd-1234-abcd-1234-abcd123e8e8e-1",
    "CreationTime": "2019-05-21T19:37:40.626Z",
```

```
    "LatestRevision": {
        "CreationTime": "2019-05-21T19:37:40.626Z",
        "Description": "Example configuration description.",
        "Revision": 1
    },
    "Name": "ExampleConfigurationName"
}
```

3. 上記のコマンドは、新しい設定の Amazon リソースネーム (ARN) を返します。この ARN は、他のコマンドでこの設定を参照する必要があるため、保存します。設定の ARN を紛失した場合は、アカウント内のすべての設定をリストして、ARN を再確認することができます。

設定の更新

このプロセスでは、カスタム Amazon MSK 設定を更新する方法について説明します。

1. 更新する設定プロパティと、それらに割り当てる値を指定するファイルを作成します。次に、設定ファイルの例を示します。

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

min.insync.replicas = 2
```

2. 次の AWS CLI コマンドを実行します。*config-file-path* は、前のステップで設定を保存したファイルへのパスに置き換えてください。

   configuration-arn は、設定の作成時に取得した ARN に置き換えてください。設定の作成時に ARN を保存しなかった場合は、list-configurations コマンドを使用して、アカウント内のすべての設定をリストすることができます。目的の設定がレスポンス内のリストに表示されます。設定の ARN もそのリストに表示されます。

```
aws kafka update-configuration --arn configuration-arn --description "Example
 configuration revision description." --server-properties fileb://config-file-path
```

3. このコマンドを実行した後の正常なレスポンスの例を以下に示します。

```
{
    "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/
abcdabcd-1234-abcd-1234-abcd123e8e8e-1",
    "LatestRevision": {
        "CreationTime": "2020-08-27T19:37:40.626Z",
```

```
        "Description": "Example configuration revision description.",
        "Revision": 2
    }
}
```

設定の削除

次の手順は、クラスターに接続されていない設定を削除する方法を示しています。クラスターに添付されている設定を削除することはできません。

1. この例を実行する際には、*configuration-arn* を、設定の作成時に取得した ARN に置き換えてください。設定の作成時に ARN を保存しなかった場合は、list-configurations コマンドを使用して、アカウント内のすべての設定をリストすることができます。目的の設定がレスポンス内のリストに表示されます。設定の ARN もそのリストに表示されます。

   ```
   aws kafka delete-configuration --arn configuration-arn
   ```

2. このコマンドを実行した後の正常なレスポンスの例を以下に示します。

   ```
   {
       "arn": " arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/
   abcdabcd-1234-abcd-1234-abcd123e8e8e-1",
       "state": "DELETING"
   }
   ```

設定メタデータを取得する

次の手順は、Amazon MSK 設定を記述して、設定に関するメタデータを取得する方法を示しています。

1. 次のコマンドは、設定に関するメタデータを返します。設定の詳細な説明を取得するには、describe-configuration-revision を実行します。

   この例を実行する際には、*configuration-arn* を、設定の作成時に取得した ARN に置き換えてください。設定の作成時に ARN を保存しなかった場合は、list-configurations コマンドを使用して、アカウント内のすべての設定をリストすることができます。目的の設定がレスポンス内のリストに表示されます。設定の ARN もそのリストに表示されます。

```
aws kafka describe-configuration --arn configuration-arn
```

2. このコマンドを実行した後の正常なレスポンスの例を以下に示します。

```
{
    "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/abcdabcd-
abcd-1234-abcd-abcd123e8e8e-1",
    "CreationTime": "2019-05-21T00:54:23.591Z",
    "Description": "Example configuration description.",
    "KafkaVersions": [
        "1.1.1"
    ],
    "LatestRevision": {
        "CreationTime": "2019-05-21T00:54:23.591Z",
        "Description": "Example configuration description.",
        "Revision": 1
    },
    "Name": "SomeTest"
}
```

設定リビジョンの詳細を取得する

このプロセスでは、Amazon MSK 設定リビジョンの詳細な説明を示します。

describe-configuration コマンドを使用して MSK 設定を記述すると、設定のメタデータが表示されます。設定の詳細な記述を取得するには、describe-configuration-revision コマンドを使用します。

- 次のコマンドを実行します。*configuration-arn* は、設定の作成時に取得した ARN に置き換えてください。設定の作成時に ARN を保存しなかった場合は、list-configurations コマンドを使用して、アカウント内のすべての設定をリストすることができます。目的の設定がレスポンス内のリストに表示されます。設定の ARN もそのリストに表示されます。

  ```
  aws kafka describe-configuration-revision --arn configuration-arn --revision 1
  ```

  このコマンドを実行した後の正常なレスポンスの例を以下に示します。

  ```
  {
  ```

```
    "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/abcdabcd-
abcd-1234-abcd-abcd123e8e8e-1",
    "CreationTime": "2019-05-21T00:54:23.591Z",
    "Description": "Example configuration description.",
    "Revision": 1,
    "ServerProperties":
 "YXV0by5jcmVhdGUudG9waWNzLmVuYWJsZSA9IHRydWUKCgp6b29rZWVwZXIuY29ubmVjdGlvbi50aW1lb3V0Lm1zI
}
```

ServerProperties の値は base64 でエンコードされます。base64 デコーダー (https://www.base64decode.org/ など) を使用して手動でデコードする場合は、カスタム設定の作成に使用した元の設定ファイルの内容を取得します。この場合、次のようになります。

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

log.roll.ms = 604800000
```

現在のリージョンのアカウントの設定を一覧表示する

このプロセスでは、現在の AWS リージョンのアカウント内のすべての Amazon MSK 設定を一覧表示する方法について説明します。

- 以下のコマンドを実行してください。

```
aws kafka list-configurations
```

このコマンドを実行した後の正常なレスポンスの例を以下に示します。

```
{
    "Configurations": [
        {
            "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/
abcdabcd-abcd-1234-abcd-abcd123e8e8e-1",
            "CreationTime": "2019-05-21T00:54:23.591Z",
            "Description": "Example configuration description.",
            "KafkaVersions": [
                "1.1.1"
            ],
            "LatestRevision": {
                "CreationTime": "2019-05-21T00:54:23.591Z",
```

```
                "Description": "Example configuration description.",
                "Revision": 1
            },
            "Name": "SomeTest"
        },
        {
            "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/
abcdabcd-1234-abcd-1234-abcd123e8e8e-1",
            "CreationTime": "2019-05-03T23:08:29.446Z",
            "Description": "Example configuration description.",
            "KafkaVersions": [
                "1.1.1"
            ],
            "LatestRevision": {
                "CreationTime": "2019-05-03T23:08:29.446Z",
                "Description": "Example configuration description.",
                "Revision": 1
            },
            "Name": "ExampleConfigurationName"
        }
    ]
}
```

Amazon MSK 設定状態

Amazon MSK 設定は、次のいずれかの状態になります。設定に対してオペレーションを実行するには、設定が ACTIVE または DELETE_FAILED 状態である必要があります。

- ACTIVE
- DELETING
- DELETE_FAILED

パッチ適用

MSK プロビジョンドクラスターへのパッチ適用

Amazon MSK は、クラスター内のブローカーのソフトウェアを定期的に更新します。メンテナンスには、計画された更新または計画外の修復が含まれます。計画されたメンテナンスには、オペレーティングシステムの更新、セキュリティの更新、およびクラスターのヘルス、セキュリティ、パフォーマンスを維持するために必要なその他のソフトウェアの更新が含まれます。計画外のメン

テナンスを実行して、インフラストラクチャの突然の劣化を解決します。Standard ブローカーと Express ブローカーでメンテナンスを実行しますが、エクスペリエンスは異なります。

標準ブローカーのパッチ適用

Standard ブローカーの更新は、ベストプラクティスに従ってもアプリケーションの書き込みと読み取りには影響しません。

Amazon MSK では、ソフトウェアのローリング更新を使用してクラスターの高可用性を維持します。このプロセス中、ブローカーは一度に 1 つずつ再起動され、Kafka により自動的にリーダーシップが別のオンラインブローカーに移行されます。Kafka クライアントには、パーティションのリーダーシップの変更を自動的に検出し、MSK クラスターへのデータの書き込みと読み取りを継続するメカニズムが組み込まれています。パッチ適用中を含め、クラスターを常にApache Kafka クライアントのベストプラクティススムーズに操作するには、 に従います。

ブローカーがオフラインになった後、クライアントで一時的な切断エラーが表示されるのは正常です。また、短い期間 (最大 2 分、通常これより短い) で p99 の読み取りおよび書き込みレイテンシー (通常高いミリ秒、最大約 2 秒) の急増が見られます。これらの急増は予想され、クライアントが新しいリーダーブローカーに再接続することによって引き起こされます。また、生成や消費には影響せず、再接続後に解決されます。詳細については、「ブローカーのオフラインおよびクライアントフェイルオーバー」を参照してください。

また、メトリクス の増加も見られます。これはUnderReplicatedPartitions、シャットダウンされたブローカーのパーティションがデータをレプリケートしていないことが予想されます。これは、他のブローカーでホストされているこれらのパーティションのレプリカとしてのアプリケーションの書き込みと読み取りには影響しません。これらのブローカーは現在、リクエストを提供しています。

ソフトウェアの更新後、ブローカーがオンラインに戻ったら、オフライン中に生成されたメッセージを「キャッチアップ」する必要があります。キャッチアップ中に、ボリュームスループットと CPU の使用率の増加も見られる場合もあります。ブローカーに十分な CPU、メモリ、ネットワーク、ボリュームリソースがある場合、これらはクラスターへの書き込みと読み取りには影響しません。

Express ブローカーのパッチ適用

Express ブローカーのメンテナンスウィンドウはありません。Amazon MSK は、時間分散方式でクラスターを継続的に更新します。つまり、1 か月間にブローカーの再起動が時折発生することが予想されます。これにより、クラスター全体の 1 回限りのメンテナンス期間に計画や対応を行う必要がなくなります。引き続きリクエストを処理する他のブローカーにリーダーシップが変わるため、ブローカーの再起動中もトラフィックは中断されません。

Express ブローカーには、メンテナンス中に発生する可能性のある負荷の変化に対してクラスターの耐障害性を高めるベストプラクティス設定とガードレールが設定されています。Amazon MSK は Express ブローカーにスループットクォータを設定し、ブローカーの再起動中に問題が発生する可能性のあるクラスターの過負荷の影響を軽減します。これらの改善により、Express ブローカーを使用する際に事前通知、計画、メンテナンスウィンドウが不要になります。

Express ブローカーは、再起動時にクライアントが自動的にフェイルオーバーするように、常に 3 つの方法でデータをレプリケートします。レプリケーション係数が 1 または 2 に設定されているため、トピックが使用できなくなることを心配する必要はありません。また、再起動された Express ブローカーのキャッチアップは、スタンダードブローカーよりも高速です。Express ブローカーのパッチ適用速度が速いほど、クラスターにスケジュールしたコントロールプレーンアクティビティの計画中断を最小限に抑えることができます。

すべての Apache Kafka アプリケーションと同様に、Express ブローカーに接続するクライアントには、まだ共有クライアント/サーバー契約があります。ブローカー間のリーダーシップフェイルオーバーを処理するようにクライアントを設定することは依然として重要です。パッチ適用中を含めApache Kafka クライアントのベストプラクティス、クラスターを常にスムーズに操作するには、に従います。ブローカーの再起動後、クライアントで一時的な切断エラーが表示されるのは正常です。これは、フォロワーブローカーがパーティションのリーダーシップを引き継ぐため、生産と消費には影響しません。Apache Kafka クライアントは自動的にフェイルオーバーし、新しいリーダーブローカーへのリクエストの送信を開始します。

ブローカーのオフラインおよびクライアントフェイルオーバー

Kafka はオフラインブローカーを許可します。ベストプラクティスに従う健全でバランスの取れたクラスター内の単一のオフラインブローカーについては、生成や消費の際に影響を及ぼしたり失敗の原因になることはありません。これは、別のブローカーがパーティションのリーダーシップを引き継ぎ、Kafka クライアントライブラリが自動的にフェイルオーバーして新しいリーダーブローカーへのリクエストの送信が開始されるためです。

クライアントサーバー契約

これにより、クライアントライブラリとサーバー側の動作での共有契約になります。サーバーでは 1 つ以上の新しいリーダーを正常に割り当て、クライアントではブローカーを変更して新しいリーダーへタイムリーにリクエストを送信する必要があります。

Kafka では、例外を使用してこのフローを制御します。

手順の例

1. ブローカー A がオフライン状態になります。
2. Kafka クライアントで例外 (通常はネットワーク切断または not_leader_for_partition) を受け取ります。
3. これらの例外により、Kafka クライアントによるメタデータの更新がトリガーされ、最新のリーダーについて知ることができます。
4. Kafka クライアントにより、他のブローカーの新しいパーティションリーダーへのリクエストの送信が再開されます。

通常、このプロセスにかかるのは Java の拡張クライアントとデフォルト設定では 2 秒未満です。クライアント側のエラーは詳細で反復的ですが、「WARN」のレベルで示されているように、懸念が引き起こされることはありません。

例: 例外 1

```
10:05:25.306 [kafka-producer-network-thread | producer-1] WARN
o.a.k.c.producer.internals.Sender - [Producer clientId=producer-1] Got
error produce response with correlation id 864845 on topic-partition
msk-test-topic-1-0, retrying (2147483646 attempts left). Error:
NETWORK_EXCEPTION. Error Message: Disconnected from node 2
```

例: 例外 2

```
10:05:25.306 [kafka-producer-network-thread | producer-1] WARN
o.a.k.c.producer.internals.Sender - [Producer clientId=producer-1] Received
invalid metadata error in produce request on partition msk-test-topic-1-41
due to org.apache.kafka.common.errors.NotLeaderOrFollowerException: For
requests intended only for the leader, this error indicates that the broker
is not the current leader. For requests intended for any replica, this
error indicates that the broker is not a replica of the topic partition..
Going to request metadata update now"
```

Kafka クライアントでは、通常 1 秒以内、最大 3 秒でこれらのエラーが自動的に解決されます。これは、クライアント側のメトリクスで p99 の生成および消費レイテンシー(通常 100 ミリ秒台の高いミリ秒) として表示されます。これより長い場合は通常、クライアント設定またはサーバー側のコントローラーで負荷の問題が発生しています。トラブルシューティングのセクションを参照してください。

他のブローカーでの BytesInPerSec および LeaderCount メトリクスの増加を確認することで、フェイルオーバーが成功したことを確認できます。これにより、トラフィックとリーダーシップが期待どおりに移動されたことが証明されます。また、UnderReplicatedPartitions メトリクスの増加も確認できます。これは、レプリカがシャットダウンブローカーでオフラインになっているときに予想されます。

トラブルシューティング

上記のフローは、クライアントとサーバーの契約を破棄することで中断される可能性があります。最も一般的な問題の原因は次のとおりです。

- Kafka クライアントライブラリの設定ミスまたは誤った使用。
- サードパーティーのクライアントライブラリによる予期しないデフォルトの動作とバグ。
- コントローラーが過負荷になることによる、パーティションリーダーの割り当ての遅延。
- 新しいコントローラーが選択されることによる、パーティションリーダーの割り当ての遅延。

リーダーシップのフェイルオーバーを処理する際に正しい動作が確実に行われるようにするため、以下をお勧めします。

- コントローラーのブローカーが適切にスケーリングされ、リーダーシップの割り当てが遅くならないように、サーバー側のベストプラクティスに従う必要があります。
- クライアントライブラリでは、クライアントがフェイルオーバーを処理できるように再試行を有効にする必要があります。
- クライアントライブラリでは、接続/リクエストストームを回避するため、retry.backoff.ms (デフォルトは 100) を設定する必要があります。
- クライアントライブラリでは、request.timeout.ms と delivery.timeout.ms をアプリケーションの SLA に沿った値に設定する必要があります。値が大きいほど、特定の障害タイプでフェイルオーバーが遅くなります。
- クライアントライブラリでは、初期検出での可用性への影響を回避するため、bootstrap.servers に少なくとも 3 つのランダムなブローカーが含まれていることを確認する必要があります。
- 一部のクライアントライブラリは他のライブラリよりもレベルが低く、アプリケーションデベロッパーによる再試行ロジックと例外処理の実装が期待されます。使用例については、クライアントライブラリ固有のドキュメントを参照し、正しい再接続/再試行ロジックに従っていることを確認してください。
- 生成、成功したリクエスト数、再試行不可能なエラーのエラー数については、クライアント側のレイテンシーをモニタリングすることをお勧めします。

- ブローカーのオフライン期間中、リクエストの生成や消費に影響がないにもかかわらず、古いサードパーティーの golang ライブラリと ruby ライブラリが冗長モードのままであることがわかりました。リクエストメトリクスに加えて、ビジネスレベルのメトリクスを常にモニタリングし、ログに実際の影響とノイズがあるかどうかを判断することをお勧めします。
- ネットワーク/not_leader の一時的な例外は正常であり、影響はなく kafka プロトコルの一部として想定されるため、警告しないでください。
- UnderReplicatedPartitions は正常であり、影響はなく単一のオフラインブローカー中に発生することが予想されるため、警戒しないでください。

Amazon MSK ログ記録

Apache Kafka ブローカーログは、Amazon CloudWatch Logs、Amazon S3、Amazon Data Firehose の 1 つまたは複数の配信先タイプに配信できます。を使用して Amazon MSK API コールを記録することもできます AWS CloudTrail。

> **Note**
>
> ブローカーログは Express ブローカーでは使用できません。

ブローカーログ

ブローカーログを使用すると、Apache Kafka アプリケーションのトラブルシューティングを行い、MSK クラスターとの通信を分析できます。新規または既存の MSK クラスターを設定して、INFO レベルのブローカーログを CloudWatch ロググループ、S3 バケット、Firehose 配信ストリームの 1 つまたは複数の配信先リソースのタイプに配信できます。その後、Firehose を使用して、配信ストリームから OpenSearch Service にログデータを配信できます。ブローカーログをクラスターに配信するようにクラスターを設定する前に、宛先リソースを作成する必要があります。Amazon MSK は、これらの宛先リソースがまだ存在しない場合、それらを作成しません。これらの 3 種類の宛先リソースとその作成方法については、次のドキュメントを参照してください。

- Amazon CloudWatch Logs
- Amazon S3
- Amazon Data Firehose

必要なアクセス許可

Amazon MSK ブローカーログの送信先を設定するには、Amazon MSK アクションに使用する IAM ID に、AWS マネージドポリシー: AmazonMSKFullAccess ポリシーに記載されているアクセス許可が必要です。

ブローカーログを S3 バケットにストリーミングするには、`s3:PutBucketPolicy` アクセス許可も必要です。S3 バケットポリシーについては、「Amazon S3 ユーザーガイド」の「S3 バケットポリシーを追加する方法」を参照してください。一般的な IAM ポリシーの詳細については、「IAM ユーザーガイド」のアクセス管理を参照してください。

SSE-KMS バケットで使用するために必要な KMS キーポリシー

カスタマーマネージドキーで マネージドキー (SSE-KMS) AWS KMSを使用して S3 バケットのサーバー側の暗号化を有効にした場合は、Amazon MSK がバケットにブローカーファイルを書き込めるように、KMS キーのキーポリシーに以下を追加します。

```
{
  "Sid": "Allow Amazon MSK to use the key.",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "delivery.logs.amazonaws.com"
    ]
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

を使用してブローカーログを設定する AWS Management Console

新しいクラスターを作成する場合は、Monitoring (モニタリング) セクションで ブローカーログ配信の見出しを探します。Amazon MSK がブローカーログを配信する宛先を指定できます。

既存のクラスターの場合は、クラスターのリストから当該クラスターを選択し、[プロパティ] タブを選択します。[ログ配信] セクションまで下にスクロールして、[編集] ボタンを選択します。Amazon MSK がブローカーログを配信する宛先を指定できます。

を使用してブローカーログを設定する AWS CLI

`create-cluster` または `update-monitoring` コマンドを使用する場合は、オプションで `logging-info` パラメータを指定し、次の例のように JSON 構造体を渡すことができます。この JSON では、3 つの送信先タイプはすべてオプションです。

```
{
  "BrokerLogs": {
    "S3": {
      "Bucket": "amzn-s3-demo-bucket",
      "Prefix": "ExamplePrefix",
      "Enabled": true
    },
    "Firehose": {
      "DeliveryStream": "ExampleDeliveryStreamName",
      "Enabled": true
    },
    "CloudWatchLogs": {
      "Enabled": true,
      "LogGroup": "ExampleLogGroupName"
    }
  }
}
```

API を使用してブローカーログを設定する

CreateClusterまたはUpdateMonitoringオペレーションに渡す JSON でオプションの `loggingInfo` 構造を指定できます。

> **Note**
>
> デフォルトでは、ブローカーロギングが有効になっている場合、Amazon MSK は `INFO` レベルのログを指定された宛先に記録します。ただし、Apache Kafka 2.4.X 以降のユーザーは、ブローカーのログレベルを任意のlog4j ログレベルに動的に設定できます。ブローカーのログレベルを動的に設定する方法については、KIP-412: 動的なアプリケーションログレベルをサポートするように管理者 API を拡張するを参照してください。ログレベルを `DEBUG` または `TRACE` に動的に設定する場合は、ログの送信先として Amazon S3 または Firehose を使

用することをお勧めします。CloudWatch Logs をログの宛先として使用し、DEBUG または TRACE レベルのロギングを動的に有効にした場合、Amazon MSK はログのサンプルを継続的に配信する場合があります。これはブローカーのパフォーマンスに大きな影響を与える可能性があるため、INFO ログレベルが問題の根本原因を特定するのに十分なほど詳細でない場合にのみ使用する必要があります。

を使用した API コールのログ記録 AWS CloudTrail

> **Note**
>
> AWS CloudTrail ログは、 を使用する場合にのみ Amazon MSK で使用できますIAM アクセスコントロール。

Amazon MSK は AWS CloudTrail、Amazon MSK のユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています。CloudTrail は、API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、Amazon MSK コンソールからの呼び出しと Amazon MSK API オペレーションへのコード呼び出しが含まれます。また、トピックやグループの作成や変更などの Apache Kafka アクションもキャプチャします。

追跡を作成すると、Amazon MSK のイベントを含め、CloudTrail イベントを Amazon S3 バケットに継続的に配信できるようになります。追跡を設定しない場合でも、CloudTrail コンソールの Event history (イベント履歴) で最新のイベントを表示できます。CloudTrail によって収集された情報を使用して、Amazon MSK または Apache Kafka アクションに対して行われた要求、要求が行われた IP アドレス、要求を行った人、要求が行われた日時、および追加の詳細を判別できます。

CloudTrail の設定や有効化の方法など、CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

CloudTrail の Amazon MSK 情報

アカウントを作成すると、Amazon Web Services アカウントで CloudTrail が有効になります。MSK クラスターでサポートされているイベントアクティビティが発生すると、そのアクティビティはイベント履歴の他の AWS サービスイベントとともに CloudTrail イベントに記録されます。AWS アカウントでの最近のイベントを表示、検索、ダウンロードできます。詳細については、CloudTrail イベント履歴を使用したイベントの表示を参照してください。

Amazon MSK のイベントを含む、Amazon Web Services アカウントのイベントの継続的なレコードについては、追跡を作成してください。追跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成すると、すべての リージョンに証跡が適用されます。追跡は AWS パーティション内のすべてのリージョンからのイベントを記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログに収集されたイベントデータをさらに分析して処理するように他の Amazon サービスを設定できます。詳細については、次を参照してください:

- 証跡の作成のための概要
- CloudTrail がサポートするサービスと統合
- CloudTrail 用 Amazon SNS 通知の構成
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

Amazon MSK は、すべての Amazon MSK オペレーションをイベントとして CloudTrail ログファイルに記録します。さらに、次の Apache Kafka アクションを ログに記録します。

- kafka-cluster:DescribeClusterDynamicConfiguration
- kafka-cluster:AlterClusterDynamicConfiguration
- kafka-cluster:CreateTopic
- kafka-cluster:DescribeTopicDynamicConfiguration
- kafka-cluster:AlterTopic
- kafka-cluster:AlterTopicDynamicConfiguration
- kafka-cluster:DeleteTopic

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストがルートユーザーまたは AWS Identity and Access Management (IAM) ユーザー認証情報を使用して行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、「CloudTrail userIdentity 要素」を参照してください。

例 : Amazon MSK ログ ファイルエントリ

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントは、任意のソースからの単一の要求を表し、要求されたアクション、アクションの日時、要求パラメーターなどに関する情報が含まれます。CloudTrail ログファイルは、パブリック API コールと Apache Kafka アクションの順序付けられたスタックトレースではないため、特定の順序で表示されません。

次の例は、`DescribeCluster` および `DeleteCluster` Amazon MSK アクションを示す CloudTrail ログエントリを示しています。

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "ABCDEF0123456789ABCDE",
        "arn": "arn:aws:iam::012345678901:user/Joe",
        "accountId": "012345678901",
        "accessKeyId": "AIDACKCEVSQ6C2EXAMPLE",
        "userName": "Joe"
      },
      "eventTime": "2018-12-12T02:29:24Z",
      "eventSource": "kafka.amazonaws.com",
      "eventName": "DescribeCluster",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "aws-cli/1.14.67 Python/3.6.0 Windows/10 botocore/1.9.20",
      "requestParameters": {
        "clusterArn": "arn%3Aaws%3Akafka%3Aus-east-1%3A012345678901%3Acluster
%2Fexamplecluster%2F01234567-abcd-0123-abcd-abcd0123efa-2"
      },
      "responseElements": null,
      "requestID": "bd83f636-fdb5-abcd-0123-157e2fbf2bde",
      "eventID": "60052aba-0123-4511-bcde-3e18dbd42aa4",
      "readOnly": true,
      "eventType": "AwsApiCall",
      "recipientAccountId": "012345678901"
    },
    {
```

```
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "ABCDEF0123456789ABCDE",
        "arn": "arn:aws:iam::012345678901:user/Joe",
        "accountId": "012345678901",
        "accessKeyId": "AIDACKCEVSQ6C2EXAMPLE",
        "userName": "Joe"
      },
      "eventTime": "2018-12-12T02:29:40Z",
      "eventSource": "kafka.amazonaws.com",
      "eventName": "DeleteCluster",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "aws-cli/1.14.67 Python/3.6.0 Windows/10 botocore/1.9.20",
      "requestParameters": {
        "clusterArn": "arn%3Aaws%3Akafka%3Aus-east-1%3A012345678901%3Acluster
%2Fexamplecluster%2F01234567-abcd-0123-abcd-abcd0123efa-2"
      },
      "responseElements": {
        "clusterArn": "arn:aws:kafka:us-east-1:012345678901:cluster/
examplecluster/01234567-abcd-0123-abcd-abcd0123efa-2",
        "state": "DELETING"
      },
      "requestID": "c6bfb3f7-abcd-0123-afa5-293519897703",
      "eventID": "8a7f1fcf-0123-abcd-9bdb-1ebf0663a75c",
      "readOnly": false,
      "eventType": "AwsApiCall",
      "recipientAccountId": "012345678901"
    }
  ]
}
```

以下の例は、kafka-cluster:CreateTopic アクションを示す CloudTrail ログエントリです。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "ABCDEFGH1IJKLMN2P34Q5",
    "arn": "arn:aws:iam::111122223333:user/Admin",
    "accountId": "111122223333",
    "accessKeyId": "CDEFAB1C2UUUUU3AB4TT",
```

```
    "userName": "Admin"
  },
  "eventTime": "2021-03-01T12:51:19Z",
  "eventSource": "kafka-cluster.amazonaws.com",
  "eventName": "CreateTopic",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "198.51.100.0/24",
  "userAgent": "aws-msk-iam-auth/unknown-version/aws-internal/3 aws-sdk-java/1.11.970
 Linux/4.14.214-160.339.amzn2.x86_64 OpenJDK_64-Bit_Server_VM/25.272-b10 java/1.8.0_272
 scala/2.12.8 vendor/Red_Hat,_Inc.",
  "requestParameters": {
    "kafkaAPI": "CreateTopics",
    "resourceARN": "arn:aws:kafka:us-east-1:111122223333:topic/IamAuthCluster/3ebafd8e-
dae9-440d-85db-4ef52679674d-1/Topic9"
  },
  "responseElements": null,
  "requestID": "e7c5e49f-6aac-4c9a-a1d1-c2c46599f5e4",
  "eventID": "be1f93fd-4f14-4634-ab02-b5a79cb833d2",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "111122223333"
}
```

メタデータ管理

Amazon MSK では、Apache ZooKeeper または KRaft メタデータ管理モードがサポートされています。

Amazon MSK の Apache Kafka バージョン 3.7.x から、ZooKeeper モードの代わりに KRaft モードを使用するクラスターを作成できます。KRaft ベースのクラスターは、メタデータを管理する際に Kafka 内のコントローラーに依存します。

トピック

- ZooKeeper モード
- KRaft モード

ZooKeeper モード

Apache ZooKeeper は、設定情報の維持、名前付け、分散化された同期の提供、およびグループサービスを提供する一元化されたサービスです。これらの種類のサービスはすべて、Apache Kafka を含む分散アプリケーションによって何らかの形で使用されます。

クラスターが ZooKeeper モードを使用している場合は、以下の手順を使用して Apache ZooKeeper 接続文字列を取得できます。ただし、Kafka 2.5 で `--zookeeper` フラグが非推奨になり、Kafka 3.0 から削除されたため、`BootstrapServerString` を使用してクラスターに接続し、管理オペレーションを実行することをお勧めします。

を使用して Apache ZooKeeper 接続文字列を取得する AWS Management Console

1. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
2. このテーブルには、このアカウントにある現在のリージョンのすべてのクラスターが表示されます。説明を表示するには、クラスターの名前を選択します。
3. [Cluster summary] (クラスターの概要) ページで、[View client information] (クライアント情報の表示) を選択します。これにより、ブートストラップブローカーと Apache ZooKeeper 接続文字列が表示されます。

AWS CLIを使用した Apache ZooKeeper 接続文字列の取得

1. クラスターの Amazon リソースネーム (ARN) がわからない場合は、アカウント内のすべてのクラスターを一覧表示することで見つけることができます。詳細については、「the section called “クラスターを一覧表示する”」を参照してください。
2. Apache ZooKeeper 接続文字列とクラスターに関するその他の情報を取得するには、*`ClusterArn`* をクラスターの ARN に置き換えて次のコマンドを実行します。

```
aws kafka describe-cluster --cluster-arn ClusterArn
```

この `describe-cluster` コマンドの出力は、次の JSON の例のようになります。

```
{
    "ClusterInfo": {
        "BrokerNodeGroupInfo": {
            "BrokerAZDistribution": "DEFAULT",
            "ClientSubnets": [
                "subnet-0123456789abcdef0",
                "subnet-2468013579abcdef1",
```

```
                "subnet-1357902468abcdef2"
            ],
            "InstanceType": "kafka.m5.large",
            "StorageInfo": {
                "EbsStorageInfo": {
                    "VolumeSize": 1000
                }
            }
        },
        "ClusterArn": "arn:aws:kafka:us-east-1:111122223333:cluster/
testcluster/12345678-abcd-4567-2345-abcdef123456-2",
        "ClusterName": "testcluster",
        "CreationTime": "2018-12-02T17:38:36.75Z",
        "CurrentBrokerSoftwareInfo": {
            "KafkaVersion": "2.2.1"
        },
        "CurrentVersion": "K13V1IB3VIYZZH",
        "EncryptionInfo": {
            "EncryptionAtRest": {
                "DataVolumeKMSKeyId": "arn:aws:kms:us-
east-1:555555555555:key/12345678-abcd-2345-ef01-abcdef123456"
            }
        },
        "EnhancedMonitoring": "DEFAULT",
        "NumberOfBrokerNodes": 3,
        "State": "ACTIVE",
        "ZookeeperConnectString": "10.0.1.101:2018,10.0.2.101:2018,10.0.3.101:2018"
    }
}
```

前述の JSON の例では、describe-cluster コマンドの出力の ZookeeperConnectString キーを示しています。クラスターにトピックを作成する必要がある場合に備え、このキーに対応する値をコピーして保存します。

> **Important**
>
> Apache ZooKeeper 接続文字列を取得できるようにするには、Amazon MSK クラスターが ACTIVE の状態である必要があります。クラスターがまだ CREATING 状態である場合、describe-cluster コマンドの出力に ZookeeperConnectString は含まれません。このような場合、数分待ってから、クラスターが ACTIVE 状態になった後に describe-cluster を再度実行します。

API を使用した Apache ZooKeeper 接続文字列の取得

API を使用して Apache ZooKeeper 接続文字列を取得するには、「DescribeCluster」を参照してください。

KRaft モード

Amazon MSK では、Kafka バージョン 3.7.x で KRaft (Apache Kafka Raft) のサポートが導入されました。Apache Kafka コミュニティでは、Apache ZooKeeper に代わって Apache Kafka クラスターのメタデータ管理に使用できる KRaft が開発されました。KRaft モードでは、クラスターメタデータは ZooKeeper ノードではなく Kafka クラスターの一部である Kafka コントローラーのグループ内で伝播されます。KRaft コントローラーは追加料金なしで含まれており、追加のセットアップや管理は必要ありません。KRaft の詳細については、「KIP-500」を参照してください。

MSK の KRaft モードに関する注意点をいくつか示します。

- KRaft モードは、新しいクラスターでのみ使用できます。クラスターが作成されると、メタデータモードを切り替えることはできません。
- MSK コンソールでは、クラスター作成ウィンドウで Kafka バージョン 3.7.x を選択し、KRaft チェックボックスをオンにすると、Kraft ベースのクラスターを作成できます。
- MSK API の `CreateCluster` または `CreateClusterV2` オペレーションを使用して KRaft モードでクラスターを作成するには、`3.7.x.kraft` をバージョンとして使用する必要があります。`3.7.x` をバージョンとして使用して、ZooKeeper モードでクラスターを作成します。
- ブローカーあたりのパーティション数は、KRaft ベースのクラスターと ZooKeeper ベースのクラスターで同じです。ただし、KRaft では、クラスターでより多くのブローカーをプロビジョニングすることで、各クラスターでより多くのパーティションをホストできます。
- Amazon MSK で KRaft モードを使用するために必要な API の変更はありません。ただし、クライアントが現在も `--zookeeper` 接続文字列を使用している場合は、`--bootstrap-server` 接続文字列を使用してクラスターに接続するようにクライアントを更新する必要があります。`--zookeeper` フラグは Apache Kafka 2.5 で非推奨になり、Kafka 3.0 以降では削除されています。そのため、クラスターへのすべての接続には、最新の Apache Kafka クライアントバージョンと `--bootstrap-server` 接続文字列を使用することをお勧めします。
- ZooKeeper モードは、ZooKeeper もサポートされているすべての Apache Kafka リリースバージョンで引き続き使用できます。Apache Kafka バージョンのサポート終了と今後の更新の詳細については、「サポート対象の Apache Kafka バージョン」を参照してください。
- 使用するすべてのツールが、ZooKeeper 接続なしで Kafka Admin API を使用できることを確認する必要があります。クラスターを Cruise Control に接続する更新された手順については、

「LinkedIn の Cruise Control for Apache Kafka を Amazon MSK で使用する」を参照してください。Cruise Control には、ZooKeeper なしで Cruise Control を実行する手順もあります。

- 管理アクションのためにクラスターの KRaft コントローラーに直接アクセスする必要はありません。ただし、オープンモニタリングを使用してメトリクスを収集する場合は、クラスターに関するコントローラー関連以外のメトリクスを収集するため、コントローラーの DNS エンドポイントも必要です。これらの DNS エンドポイントは、MSK コンソールから、または ListNodes API オペレーションを使用して取得できます。KRaft ベースのクラスター用にオープンモニタリングを設定する更新された手順については、「Prometheus で MSK プロビジョンドクラスターをモニタリングする」を参照してください。
- ZooKeeper モードのクラスターを介して KRaft モードのクラスターをモニタリングする必要がある追加の CloudWatch メトリクスはありません。MSK では、クラスターで使用される KRaft コントローラーが管理されます。
- `--bootstrap-server` 接続文字列を使用して、KRaft モードのクラスターで ACL の管理を継続できます。ACL の管理に `--zookeeper` 接続文字列を使用しないでください。「Apache Kafka ACL」を参照してください。
- KRaft モードでは、クラスターのメタデータは Kafka 内の KRaft コントローラーに保存され、外部の ZooKeeper ノードには保存されません。したがって、ZooKeeper ノードと同様に、コントローラーノードへのアクセスを個別に制御する必要はありません。

Amazon MSK リソース

リソースとは、Amazon MSK においては、文脈に応じて 2 つの意味があります。API の文脈では、リソースはオペレーションを呼び出すことができる構造を指します。これらのリソースとそれらに対して呼び出すことができるオペレーションのリストについては、Amazon MSK API リファレンスのリソースを参照してください。the section called “IAM アクセスコントロール” の文脈における意味では、the section called “認可ポリシーリソース” セクションで定義されているように、ユーザーはリソースへのアクセスを許可または拒否することができ、このようなエンティティのことをリソースと呼びます。

Apache Kafka バージョン

Amazon MSK クラスターを作成するときは、クラスターに含める Apache Kafka バージョンを指定します。既存のクラスターの Apache Kafka バージョンを更新することもできます。この章のトピックは、Kafka バージョンサポートのタイムラインとベストプラクティスの提案を理解するのに役立ちます。

トピック

- サポート対象の Apache Kafka バージョン
- Amazon MSK バージョンのサポート

サポート対象の Apache Kafka バージョン

Amazon Managed Streaming for Apache Kafka (Amazon MSK) は、次の Apache Kafka および Amazon MSK バージョンをサポートします。Apache Kafka コミュニティによるサポート期間は、バージョンのリリース日から約 12 か月間です。詳細については、「Apache Kafka EOL (end of life) policy」(Apache Kafka EOL (サポート終了) ポリシー) を参照してください。

サポート対象の Apache Kafka バージョン

Apache Kafka バージョン	MSK リリース日	サポート終了日
1.1.1	--	2024-06-05
2.1.0	--	2024-06-05
2.2.1	2019-07-31	2024-06-08
2.3.1	2019-12-19	2024-06-08
2.4.1	2020-04-02	2024-06-08
2.4.1.1	2020-09-09	2024-06-08
2.5.1	2020-09-30	2024-06-08
2.6.0	2020-10-21	2024-09-11
2.6.1	2021-01-19	2024-09-11
2.6.2	2021-04-29	2024-09-11
2.6.3	2021-12-21	2024-09-11
2.7.0	2020-12-29	2024-09-11
2.7.1	2021-05-25	2024-09-11

Apache Kafka バージョン	MSK リリース日	サポート終了日
2.7.2	2021-12-21	2024-09-11
2.8.0	2021-05-19	2024-09-11
2.8.1	2022-10-28	2024-09-11
2.8.2 階層型	2022-10-28	2025-01-14
3.1.1	2022-06-22	2024-09-11
3.2.0	2022-06-22	2024-09-11
3.3.1	2022-10-26	2024-09-11
3.3.2	2023-03-02	2024-09-11
3.4.0	2023-05-04	2025-08-04
3.5.1	2023-09-26	2025-10-23
3.6.0	2023-11-16	--
3.7.x	2024-05-29	--
3.8.x	2025-02-20	--
3.9.x	2025-04-21	--
4.0.x	2025-05-16	--

Amazon MSK バージョンサポートポリシーの詳細については、「」を参照してくださいAmazon MSK バージョンのサポートポリシー。

Amazon MSK バージョン 4.0.x

Amazon Managed Streaming for Apache Kafka (Amazon MSK) が Apache Kafka バージョン 4.0 をサポートするようになりました。このバージョンでは、クラスターの管理とパフォーマンスにおける最新の進歩が MSK プロビジョンドにもたらされます。Kafka 4.0 では、新しいコンシューマー再調整プロトコルが導入されました。これにより、グループの再調整をスムーズかつ迅速に行うことがで

きます。さらに、Kafka 4.0 では、Java 17 を使用するブローカーとツールが必要であり、セキュリティとパフォーマンスが向上し、さまざまなバグ修正と改善が含まれ、Apache ZooKeeper によるメタデータ管理が廃止されます。

詳細と改善点とバグ修正の完全なリストについては、バージョン 4.0 の Apache Kafka リリースノートを参照してください。

Amazon MSK バージョン 3.9.x

Amazon Managed Streaming for Apache Kafka (Amazon MSK) が Apache Kafka バージョン 3.9 をサポートするようになりました。このバージョンでは、トピックレベルで階層型ストレージを無効にするときに階層型データを保持できます。コンシューマーアプリケーションは、ローカルストレージとリモートストレージの両方で継続的なログオフセットを維持しながら、リモートログ開始オフセット (Rx) から履歴データを引き続き読み取ることができます。

詳細と改善点とバグ修正の完全なリストについては、バージョン 3.9.x の Apache Kafka リリースノートを参照してください。

Amazon MSK バージョン 3.8.x

Amazon Managed Streaming for Apache Kafka (Amazon MSK) で Apache Kafka バージョン 3.8 がサポートされるようになりました。メタデータ管理に KRAFT モードまたは ZooKeeper モードを使用してバージョン 3.8 を使用して新しいクラスターを作成したり、既存の ZooKeeper ベースのクラスターをバージョン 3.8 を使用するようにアップグレードしたりできます。Apache Kafka バージョン 3.8 には、パフォーマンスを向上させるいくつかのバグ修正と新機能が含まれています。主な新機能には、圧縮レベル設定のサポートが含まれます。これにより、デフォルトの圧縮レベルを変更できるため、lz4、zstd、gzip などの圧縮タイプを使用する際のパフォーマンスをさらに最適化できます。

詳細と改善とバグ修正の完全なリストについては、バージョン 3.8.x の Apache Kafka リリースノートを参照してください。

Apache Kafka バージョン 3.7.x (本番環境対応の階層型ストレージ搭載)

MSK での Apache Kafka バージョン 3.7.x には、Apache Kafka バージョン 3.7.0 のサポートが含まれています。新しい 3.7.x バージョンを使用する場合は、クラスターを作成するか、既存のクラスターをアップグレードします。このバージョン命名の変更により、Apache Kafka コミュニティで 3.7.1 などの新しいパッチ修正バージョンがリリースされたときに、それを採用する必要がなくなりました。Amazon MSK は、将来のパッチバージョンが使用可能になると、自動的に 3.7.x を更新してサポートします。これにより、バージョンアップグレードをトリガーすることなく、パッチ修正バージョンを通じて使用可能なセキュリティおよびバグ修正のメリットを得ることができ

ます。Apache Kafka によってリリースされたこれらのパッチ修正バージョンでは、バージョンの互換性が損なわれることはありません。そのため、クライアントアプリケーションの読み込みまたは書き込みエラーを心配することなく、新しいパッチ修正バージョンのメリットを享受できます。CloudFormation などのインフラストラクチャ自動化ツールが、このバージョン命名の変更を考慮して更新されていることを確認してください。

Amazon MSK は、Apache Kafka バージョン 3.7.x で KRaft モード (Apache Kafka Raft) をサポートするようになりました。Amazon MSK では、ZooKeeper ノードと同様に、KRaft コントローラーが追加料金なしで含まれており、追加のセットアップや管理は必要ありません。Apache Kafka バージョン 3.7.x で、KRaft モードまたは ZooKeeper モードでクラスターを作成できるようになりました。Kraft モードでは、Zookeeper ベースのクラスターのブローカークォータ 30 と比較して、制限の引き上げをリクエストしなくても最大 60 ブローカーの追加が可能なため、クラスターごとにより多くのパーティションをホストできます。MSK での KRaft の詳細については、「」を参照してくださいKRaft モード。

Apache Kafka バージョン 3.7.x には、パフォーマンスを向上させるいくつかのバグ修正と新機能も含まれています。主な改善点としては、クライアントのリーダー検出の最適化、ログセグメントフラッシュの最適化オプションなどがあります。改善点とバグ修正の詳細な一覧については、Apache Kafka 3.7.0 のリリースノートを参照してください。

Apache Kafka バージョン 3.6.0 (本番移行可能階層型ストレージ搭載)

Apache Kafka バージョン 3.6.0 (本番移行可能階層ストレージ) の詳細については、Apache Kafka ダウンロードサイトのリリースノートを参照してください。

Amazon MSK は、安定性を保つため、このリリースでもクォーラム管理に ZooKeeper を引き続き使用および管理します。

Amazon MSK バージョン 3.5.1

Amazon Managed Streaming for Apache Kafka (Amazon MSK) が、新規および既存のクラスターで Apache Kafka バージョン 3.5.1 をサポートするようになりました。Apache Kafka 3.5.1 には、パフォーマンスを向上させるいくつかのバグ修正と新機能が含まれています。主な機能には、コンシューマーに対する新しいラック認識のパーティション割り当ての導入などがあります。Amazon MSK は、このリリースでもクォーラム管理に ZooKeeper を引き続き使用および管理します。改善点とバグ修正の詳細な一覧については、Apache Kafka 3.5.1 のリリースノートを参照してください。

Apache Kafka バージョン 3.5.1 の詳細については、Apache Kafka ダウンロードサイトのリリースノートを参照してください。

Amazon MSK バージョン 3.4.0

Amazon Managed Streaming for Apache Kafka (Amazon MSK) が、新規および既存のクラスターで Apache Kafka バージョン 3.4.0 をサポートするようになりました。Apache Kafka 3.4.0 には、パフォーマンスを向上させるいくつかのバグ修正と新機能が含まれています。主な機能には、最も近いレプリカからフェッチする際の安定性を向上させる修正が含まれます。Amazon MSK は、このリリースでもクォーラム管理に ZooKeeper を引き続き使用および管理します。改善点とバグ修正の詳細な一覧については、Apache Kafka 3.4.0 のリリースノートを参照してください。

Apache Kafka バージョン 3.4.0 の詳細については、Apache Kafka ダウンロードサイトのリリースノートを参照してください。

Amazon MSK バージョン 3.3.2

Amazon Managed Streaming for Apache Kafka (Amazon MSK) が、新規および既存のクラスターで Apache Kafka バージョン 3.3.2 をサポートするようになりました。Apache Kafka 3.3.2 には、パフォーマンスを向上させるいくつかのバグ修正と新機能が含まれています。主な機能には、最も近いレプリカからフェッチする際の安定性を向上させる修正が含まれます。Amazon MSK は、このリリースでもクォーラム管理に ZooKeeper を引き続き使用および管理します。改善点とバグ修正の詳細な一覧については、Apache Kafka 3.3.2 のリリースノートを参照してください。

Apache Kafka バージョン 3.3.2 の詳細については、Apache Kafka ダウンロードサイトのリリースノートを参照してください。

Amazon MSK バージョン 3.3.1

Amazon Managed Streaming for Apache Kafka (Amazon MSK) が、新規および既存のクラスターで Apache Kafka バージョン 3.3.1 をサポートするようになりました。Apache Kafka 3.3.1 には、パフォーマンスを向上させるいくつかのバグ修正と新機能が含まれています。主な機能には、メトリクスとパーティショナーの強化が含まれます。Amazon MSK は、安定性を保つため、このリリースでもクォーラム管理に ZooKeeper を引き続き使用および管理します。改善点とバグ修正の詳細な一覧については、Apache Kafka 3.3.1 のリリースノートを参照してください。

Apache Kafka バージョン 3.3.1 の詳細については、Apache Kafka ダウンロードサイトのリリースノートを参照してください。

Amazon MSK バージョン 3.1.1

Amazon Managed Streaming for Apache Kafka (Amazon MSK) が、新規および既存のクラスターで Apache Kafka バージョン 3.1.1 および 3.2.0 をサポートするようになりました。Apache Kafka 3.1.1

と Apache Kafka 3.2.0 には、パフォーマンスを向上させるいくつかのバグ修正と新機能が含まれています。主な機能には、メトリクスの強化やトピック ID の使用などがあります。MSK は、安定性を確保するために、このリリースでもクォーラム管理に ZooKeeper を引き続き使用および管理します。改善点とバグ修正の詳細な一覧については、Apache Kafka 3.1.1 および 3.2.0 のリリースノートを参照してください。

Apache Kafka バージョン 3.1.1 および 3.2.0 の詳細については、Apache Kafka ダウンロードサイトの 3.2.0 のリリースノートと 3.1.1 のリリースノートを参照してください。

Amazon MSK 階層型ストレージバージョン 2.8.2.tiered

このリリースは、Apache Kafka バージョン 2.8.2 の Amazon MSK のみのバージョンであり、オープンソースの Apache Kafka クライアントと互換性があります。

2.8.2.tiered リリースには、Apache Kafka の KIP-405 で導入された API と互換性のある階層型ストレージ機能が含まれています。Amazon MSK 階層型ストレージ機能の詳細については、「標準ブローカーの階層型ストレージ」を参照してください。

Apache Kafka バージョン 2.5.1

Apache Kafka バージョン 2.5.1 には、Apache ZooKeeper および管理クライアントの転送中の暗号化など、いくつかのバグ修正と新機能が含まれています。Amazon MSK は、DescribeClusterオペレーションでクエリできる TLS ZooKeeper エンドポイントを提供します。

DescribeClusterオペレーションの出力には、TLS zookeeper エンドポイントを一覧表示する `ZookeeperConnectStringTls` ノードが含まれます。

次の例は、`DescribeCluster` オペレーションの応答の `ZookeeperConnectStringTls` ノードを示しています。

```
"ZookeeperConnectStringTls": "z-3.awskafkatutorialc.abcd123.c3.kafka.us-
east-1.amazonaws.com:2182,z-2.awskafkatutorialc.abcd123.c3.kafka.us-
east-1.amazonaws.com:2182,z-1.awskafkatutorialc.abcd123.c3.kafka.us-
east-1.amazonaws.com:2182"
```

zookeeper で TLS 暗号化を使用する方法については、「Apache ZooKeeper での TLS セキュリティの使用」を参照してください。

Apache Kafka バージョン 2.5.1 の詳細については、Apache Kafka ダウンロードサイトのリリースノートを参照してください。

Amazon MSK のバグ修正バージョン 2.4.1.1

このリリースは、Apache Kafka バージョン 2.4.1 の Amazon MSK のみのバグ修正バージョンです。このバグ修正リリースには、KAFKA-9752の修正が含まれています。これは、コンシューマーグループが継続的にリバランスして `PreparingRebalance` 状態のままになるまれな問題です。この問題は、Apache Kafka バージョン 2.3.1 および 2.4.1 を実行しているクラスターに影響します。このリリースには、Apache Kafka バージョン 2.5.0 で利用可能なコミュニティで作成された修正が含まれています。

> **Note**
>
> バージョン 2.4.1.1 を実行している Amazon MSK クラスターは、Apache Kafka バージョン 2.4.1 と互換性のあるすべての Apache Kafka クライアントと互換性があります。

Apache Kafka 2.4.1 を使用する場合は、新しい Amazon MSK クラスターに MSK バグ修正バージョン 2.4.1.1 を使用することをお勧めします。Apache Kafka バージョン 2.4.1 を実行している既存のクラスターをこのバージョンに更新して、この修正を組み込むことができます。既存のクラスターのアップグレードについては、「Apache Kafka バージョンのアップグレード」を参照してください。

クラスターをバージョン 2.4.1.1 にアップグレードせずにこの問題を回避するには、Amazon MSK クラスターをトラブルシューティングする ガイドの コンシューマーグループが `PreparingRebalance` 状態から変化しない セクションを参照してください。

Apache Kafka バージョン 2.4.1 (代わりに 2.4.1.1 を使用)

> **Note**
>
> Apache Kafka バージョン 2.4.1 で MSK クラスターを作成することはできなくなりました。代わりに、Apache Kafka バージョン 2.4.1 と互換性のあるクライアントで Amazon MSK のバグ修正バージョン 2.4.1.1 を使用できます。また、Apache Kafka バージョン 2.4.1 を使用する MSK クラスターが既にある場合は、代わりに Apache Kafka バージョン 2.4.1.1 を使用するように更新することをお勧めします。

KIP-392 は、Apache Kafka の 2.4.1 リリースに含まれている主要な Kafka 改善提案の1つです。この改善により、コンシューマーは最も近いレプリカからフェッチできます。この機能を使用するには、コンシューマープロパティの `client.rack` をコンシューマーのアベイラビリティーゾーンの ID に

設定します。AZ ID の例は `use1-az1` です。Amazon MSK は、`broker.rack` をブローカーのアベイラビリティーゾーンの ID に設定します。また、`replica.selector.class` 設定プロパティを `org.apache.kafka.common.replica.RackAwareReplicaSelector` に設定する必要があります。これは、Apache Kafka が提供するラック認識の実装です。

Apache Kafka のこのバージョンを使用すると、`PER_TOPIC_PER_BROKER` モニタリングレベルのメトリクスは、その値が初めてゼロ以外になった後にのみ表示されます。詳細については、「the section called “`PER_TOPIC_PER_BROKER` レベルモニタリング”」を参照してください。

アベイラビリティーゾーン IDs「リソースの AZ IDs」を参照してください。 AWS Resource Access Manager

設定プロパティの設定については、「the section called “ブローカー設定”」を参照してください。

KIP-392 の詳細については、Confluence ページの Allow Consumers to Fetch from Closest Replicaを参照してください。

Apache Kafka バージョン 2.4.1 の詳細については、Apache Kafka ダウンロードサイトのリリースノートを参照してください。

Amazon MSK バージョンのサポート

このトピックでは、Amazon MSK バージョンのサポートポリシー と Apache Kafka バージョンのアップグレード の手順について説明します。Kafka バージョンをアップグレードする場合は、「バージョンアップグレードのベストプラクティス」で説明されているベストプラクティスに従ってください。

トピック

- Amazon MSK バージョンのサポートポリシー
- Apache Kafka バージョンのアップグレード
- バージョンアップグレードのベストプラクティス

Amazon MSK バージョンのサポートポリシー

このセクションでは、Amazon MSK でサポートされている Kafka バージョンのサポートポリシーについて説明します。

- すべての Kafka バージョンは、サポート終了日までサポートされます。サポート終了日の詳細については、「サポート対象の Apache Kafka バージョン」を参照してください。サポート終了日

までに、MSK クラスターを推奨される Kafka バージョン以降のバージョンにアップグレードしてください。Apache Kafka バージョンのアップグレードの詳細については、「」を参照してくださいApache Kafka バージョンのアップグレード。サポート終了日を過ぎた Kafka バージョンを使用するクラスターは、推奨される Kafka バージョンに自動的にアップグレードされます。自動アップグレードは、サポート終了日以降にいつでも実行される可能性があります。アップグレード前に通知を受け取ることはありません。

- MSK では、新しく作成された、サポート終了日が公開されている Kafka バージョンを使用するクラスターのサポートを段階的に廃止します。

Apache Kafka バージョンのアップグレード

既存の MSK クラスターを新しいバージョンの Apache Kafka にアップグレードできます。

> **Note**
>
> - 既存の MSK クラスターを ZooKeeper ベースの Apache Kafka バージョンから、KRaft モードを使用する、または必要とする新しいバージョンにアップグレードすることはできません。代わりに、クラスターをアップグレードするには、KRaft がサポートする Kafka バージョンで新しい MSK クラスターを作成し、古いクラスターからデータとワークロードを移行します。
> - Amazon MSK はサーバーソフトウェアのみをアップグレードします。クライアントはアップグレードされません。
> - 既存の MSK クラスターを古いバージョンの Apache Kafka にダウングレードすることはできません。

MSK クラスターの Apache Kafka バージョンをアップグレードするときは、クライアント側のソフトウェアもチェックして、そのバージョンでクラスターの新しい Apache Kafka バージョンの機能を使用できることを確認します。

アップグレード中にクラスターを高可用性にする方法については、「」を参照してくださいthe section called “高可用性クラスターの構築”。

を使用して Apache Kafka バージョンをアップグレードする AWS Management Console

1. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
2. ナビゲーションバーで、MSK クラスターを作成したリージョンを選択します。

3. アップグレードする MSK クラスターを選択します。
4. Properties タブで、Apache Kafka バージョンセクションで Upgrade を選択します。
5. Apache Kafka バージョンセクションで、次の操作を行います。

   a. Apache Kafka バージョンの選択ドロップダウンリストで、アップグレードするバージョンを選択します。例えば、[**3.9.x**] を選択します。

   b. （オプション) バージョンの互換性を選択すると、クラスターの現在のバージョンとアップグレードするバージョンとの互換性が表示されます。次に、「選択して続行する」または「キャンセル」を選択します。

   c. クラスター設定の更新チェックボックスをオンにすると、アップグレードされたバージョンと互換性のある新しい Kafka 設定リビジョンが自動的に適用されます。これにより、互換性が確保され、アップグレードされたバージョンの新機能や改善が可能になります。ただし、既存のカスタム設定を維持する場合はスキップします。

   d. [アップグレード] を選択します。

を使用して Apache Kafka バージョンをアップグレードする AWS CLI

1. *ClusterArn*をクラスターの作成時に取得したAmazon リソースネーム (ARN) に置き換えて、次のコマンドを実行します。クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。詳細については、「the section called “クラスターを一覧表示する”」を参照してください。

   ```
   aws kafka get-compatible-kafka-versions --cluster-arn ClusterArn
   ```

   このコマンドの出力には、クラスターをアップグレードできる Apache Kafka バージョンのリストが含まれます。次の例のようになります。

   ```
   {
       "CompatibleKafkaVersions": [
           {
               "SourceVersion": "2.2.1",
               "TargetVersions": [
                   "2.3.1",
                   "2.4.1",
                   "2.4.1.1",
                   "2.5.1"
               ]
           }
   ```

```
    ]
}
```

2. *ClusterArn*をクラスターの作成時に取得したAmazon リソースネーム (ARN) に置き換えて、次のコマンドを実行します。クラスターの ARN がない場合は、すべてのクラスターを一覧表示することで見つけられます。詳細については、「the section called “クラスターを一覧表示する”」を参照してください。

   Current-Cluster-Version を、クラスターの現在のバージョンに置き換えます。*TargetVersion* では、前のコマンド出力から任意のターゲットバージョンを指定できます。

   > **Important**
   >
   > クラスターのバージョンは単純な整数ではありません。クラスターの最新バージョンを検索するには、DescribeCluster オペレーションまたは describe-cluster AWS CLI コマンドを使用します。サンプルのバージョンは KTVPDKIKX0DER です。

```
aws kafka update-cluster-kafka-version --cluster-arn ClusterArn --current-
version Current-Cluster-Version --target-kafka-version TargetVersion
```

   前のコマンドの出力は以下の JSON のようになります。

```
{

    "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
abcdefab-1234-abcd-5678-cdef0123ab01-2",
    "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef"
}
```

3. update-cluster-kafka-version オペレーションの結果を取得するには、*ClusterOperationArn*を update-cluster-kafka-version コマンドの出力で取得した ARN に置き換えて、次のコマンドを実行します。

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

この describe-cluster-operation コマンドの出力は、次の JSON の例のようになります。

```
{
    "ClusterOperationInfo": {
        "ClientRequestId": "62cd41d2-1206-4ebf-85a8-dbb2ba0fe259",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2021-03-11T20:34:59.648000+00:00",
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "UPDATE_IN_PROGRESS",
        "OperationSteps": [
            {
                "StepInfo": {
                    "StepStatus": "IN_PROGRESS"
                },
                "StepName": "INITIALIZE_UPDATE"
            },
            {
                "StepInfo": {
                    "StepStatus": "PENDING"
                },
                "StepName": "UPDATE_APACHE_KAFKA_BINARIES"
            },
            {
                "StepInfo": {
                    "StepStatus": "PENDING"
                },
                "StepName": "FINALIZE_UPDATE"
            }
        ],
        "OperationType": "UPDATE_CLUSTER_KAFKA_VERSION",
        "SourceClusterInfo": {
            "KafkaVersion": "2.4.1"
        },
        "TargetClusterInfo": {
            "KafkaVersion": "2.6.1"
        }
    }
}
```

`OperationState` の値が `UPDATE_IN_PROGRESS` の場合は、しばらく待ってから再度 `describe-cluster-operation` コマンドを実行します。オペレーションが完了すると、`OperationState` の値は `UPDATE_COMPLETE` になります。Amazon MSK がオペレーションを完了するのに必要な時間はさまざまであるため、オペレーションが完了するまで繰り返しチェックする必要がある場合があります。

API を使用して Apache Kafka バージョンをアップグレードする

1. GetCompatibleKafkaVersions オペレーションを呼び出して、クラスターをアップグレードできる Apache Kafka バージョンのリストを取得します。
2. UpdateClusterKafkaVersion オペレーションを呼び出して、クラスターを互換性のある Apache Kafka バージョンのいずれかにアップグレードします。

バージョンアップグレードのベストプラクティス

Kafka バージョンアップグレードプロセスの一環として実行されるローリング更新中にクライアントの継続性を確保するために、クライアントと Apache Kafka トピックの設定について次のように確認します。

- トピックのレプリケーション係数 (RF) の最小値は、2 つの AZ クラスターの場合は 2 に、3 つの AZ クラスターの場合は 3 に設定します。RF 値が 2 の場合、パッチ適用中にオフラインパーティションが発生する可能性があります。
- 最小同期レプリカ (minISR) を、レプリケーション係数 (RF) の より 1 小さい最大値に設定しますminiISR = (RF) - 1。これにより、パーティションレプリカセットは、1 つのレプリカがオフラインまたは過小レプリケーションされることを許容できます。
- 複数のブローカー接続文字列を使用するようにクライアントを設定します。クライアントの接続文字列に複数のブローカーが含まれていると、クライアント I/O をサポートする特定のブローカーにパッチが適用され始めた場合にフェイルオーバーができるようになります。複数のブローカーを含む接続文字列を取得する方法については、「Amazon MSK クラスターのブートストラップブローカーを取得する」を参照してください。
- 新しいバージョンで使用できる機能を活用するために、接続クライアントを推奨バージョン以降にアップグレードすることをお勧めします。クライアントのアップグレードは、MSK クラスターの Kafka バージョンのサポート終了 (EOL) 日に左右されないため、EOL 日までに完了する必要はありません。Apache Kafka は、双方向のクライアント互換性ポリシーを提供しているため、古いクライアントが新しいクラスターで動作することも、その逆も可能になります。

- バージョン 3.x.x を使用する Kafka クライアントには、デフォルトの `acks=all` と `enable.idempotence=true` が付属している可能性があります。`acks=all` は、以前のデフォルトの `acks=1` とは異なり、すべての同期レプリカが生成要求を承認することを保証することで追加の耐久性を提供します。同様に、`enable.idempotence` のデフォルトは以前は `false` でした。`enable.idempotence=true` をデフォルトに変更すると、重複したメッセージが発生する可能性が低くなります。これらの変更はベストプラクティス設定と見なされ、通常のパフォーマンスパラメータの範囲内でわずかな追加遅延が発生する可能性があります。
- 新しい MSK クラスターを作成するときは、推奨される Kafka バージョンを使用します。推奨される Kafka バージョンを使用すると、最新の Kafka 機能と MSK 機能を活用できます。

Amazon MSK クラスターをトラブルシューティングする

次の情報は、お使いの Amazon MSK クラスター に関する問題を解決するために役立ちます。AWS re:Post に問題を投稿することもできます。Amazon MSK Replicator のトラブルシューティングについては、「MSK Replicator のトラブルシューティング」を参照してください。

トピック

- ボリュームを置き換えると、レプリケーションの過負荷によりディスクが飽和します。
- コンシューマーグループが PreparingRebalance 状態から変化しない
- Amazon CloudWatch Logs へのブローカーログの配信エラー
- デフォルトのセキュリティグループがない
- クラスターが [作成中] 状態のまま停止しているように見える
- クラスターの状態が [作成中] から [失敗] に変わる
- クラスターの状態は [アクティブ] であるが、プロデューサーがデータを送信できないか、コンシューマーがデータを受信できない
- AWS CLI は Amazon MSK を認識しません
- パーティションがオフラインになるか、レプリカが同期しない
- ディスク容量が不足している
- メモリが不足している
- プロデューサーが NotLeaderForPartitionException を取得する
- レプリケート不足のパーティション (URP) 数がゼロより大きい
- クラスターには __amazon_msk_canary と __amazon_msk_canary_state というトピックがあります

- パーティションレプリケーションが失敗する
- パブリックアクセスが有効になっているクラスターにアクセスできない
- 内からクラスターにアクセスできない AWS: ネットワークの問題
- 認証の失敗: 接続が多すぎる
- 認証の失敗: セッションが短すぎる
- MSK サーバーレス: クラスターの作成に失敗する
- MSK 設定で KafkaVersionsList を更新できない

ボリュームを置き換えると、レプリケーションの過負荷によりディスクが飽和します。

予期しないボリュームハードウェア障害が発生した場合、Amazon MSK はボリュームを新しいインスタンスに置き換えることがあります。Kafka は、クラスター内の他のブローカーからパーティションをレプリケートすることで、新しいボリュームに再入力します。パーティションがレプリケートされてキャッチアップされると、リーダーシップと同期レプリカ (ISR) メンバーシップの対象となります。

問題

ボリュームの置き換えから復帰するブローカーでは、さまざまなサイズの一部のパーティションが他のパーティションよりも先にオンラインに戻る場合があります。これは、これらのパーティションが、他のパーティションをキャッチアップしている (レプリケートしている) のと同じブローカーからのトラフィックを提供している可能性があるため、問題になる可能性があります。このレプリケーショントラフィックは、基盤となるボリュームスループット制限を飽和させる場合があります。制限は、デフォルトの場合、1 秒あたり 250 MiB です。この飽和状態が発生すると、既にキャッチアップされているパーティションが影響を受け、キャッチアップされたパーティション (リモート Ack `acks=all` によるリーダーパーティションだけでなく) と ISR を共有するブローカーのクラスター全体のレイテンシーが発生します。この問題は、サイズが異なるパーティションの数が多い大規模なクラスターでより一般的です。

推奨事項

- レプリケーション I/O の体制を改善するには、ベストプラクティスのスレッド設定が整っていることを確認します。
- 基になるボリューム飽和の可能性を減らすには、より高いスループットでプロビジョニングされたストレージを有効にします。高スループットレプリケーションの場合、最小スループット値は

500 MiB/秒が推奨されますが、実際に必要な値はスループットとユースケースによって異なります。Amazon MSK クラスター内の標準ブローカーのストレージスループットをプロビジョニングする。

- レプリケーション圧力を最小限に抑えるには、num.replica.fetchers をデフォルト値 2 に下げます。

コンシューマーグループが PreparingRebalance 状態から変化しない

1 つ以上のコンシューマーグループが永続的リバランシング状態から変化しない場合、原因は Apache Kafka の問題である、KAFKA-9752 の可能性があります。これは Apache Kafka バージョン 2.3.1 および 2.4.1 に影響します。

この問題を解決するには、クラスターをこの問題の修正が含まれている Amazon MSK のバグ修正バージョン 2.4.1.1 にアップグレードすることをお勧めします。既存のクラスターを Amazon MSK バグ修正バージョン 2.4.1.1 に更新する方法については、「Apache Kafka バージョンのアップグレード」を参照してください。

クラスターを Amazon MSK バグ修正バージョン 2.4.1.1 にアップグレードせずにこの問題を解決するための回避策は、Kafka クライアントを 静的メンバーシッププロトコル を使用するように設定するか、またはスタックしたコンシューマーグループの調整ブローカーノードを 識別と再起動 します。

静的メンバーシッププロトコルの実装

クライアントに静的メンバーシッププロトコルを実装するには、以下を実行します。

1. Kafka Consumers の設定で group.instance.id のプロパティを、グループ内のコンシューマを識別する静的文字列に設定します。
2. 設定の他のインスタンスが静的文字列を使用できるように更新されていることを確認してください。
3. Kafka コンシューマーに変更をデプロイします。

静的メンバーシッププロトコルの使用は、クライアント構成のセッションタイムアウトが、コンシューマグループのリバランスを早期にトリガーすることなくリカバリできる期間に設定されている場合、より効果的です。例えば、コンシューマーアプリケーションが 5 分間の使用不可を許容できる場合、セッションタイムアウトの妥当な値は、デフォルト値の 10 秒ではなく 4 分になります。

> **Note**
>
> 静的メンバーシッププロトコルを使用すると、この問題が発生する確率を低下させることができます。しかし、静的メンバーシッププロトコルを使用していても、この問題が発生する可能性はあります。

コーディネートブローカーノードの再起動

コーディネートブローカーノードを再起動するには、以下の操作を実行します。

1. 「`kafka-consumer-groups.sh`」コマンドを使用してグループコーディネーターを特定します。
2. スタックしたコンシューマーグループのグループコーディネーターを、API アクションの RebootBroker を使用してリスタートします。

Amazon CloudWatch Logs へのブローカーログの配信エラー

Amazon CloudWatch Logs にブローカーログを送信するようにクラスターを設定する際には、以下の 2 つのうちのどちらかの例外が発生することがあります。

`InvalidInput.LengthOfCloudWatchResourcePolicyLimitExceeded` 例外が発生した場合は、`/aws/vendedlogs/` から始まるロググループを使用して再試行してください。詳細については、「特定の Amazon Web Services からのログ記録を有効にする」を参照してください。

`InvalidInput.NumberOfCloudWatchResourcePoliciesLimitExceeded` 例外が発生した場合は、アカウントで Amazon CloudWatch Logs の既存のポリシーを選択し、次の JSON をそのポリシーに追加してください。

```
{"Sid":"AWSLogDeliveryWrite","Effect":"Allow","Principal":
{"Service":"delivery.logs.amazonaws.com"},"Action":
["logs:CreateLogStream","logs:PutLogEvents"],"Resource":["*"]}
```

上の JSON を既存のポリシーに追加しようとして、選択したポリシーの最大長に達しているというエラーが表示された場合は、Amazon CloudWatch Logs の別のポリシーに JSON を追加してみてください。既存のポリシーに JSON を追加したら、もう一度 Amazon CloudWatch Logs へのブローカーログの配信を設定してください。

デフォルトのセキュリティグループがない

クラスターを作成しようとしたときに、デフォルトのセキュリティグループがないことを示すエラーが表示される場合は、共有された VPC を使用している可能性があります。管理者に、この VPC のセキュリティグループを記述するアクセス許可を付与してもらうよう依頼して、再試行してください。このアクションを許可するポリシーの例については、「Amazon EC2: 特定の VPC に関連付けられた EC2 セキュリティグループの管理をプログラムによりコンソールで許可する」を参照してください。

クラスターが [作成中] 状態のまま停止しているように見える

クラスターの作成には、最大 30 分かかる場合があります。30 分間待ってから、クラスターの状態を再度確認します。

クラスターの状態が [作成中] から [失敗] に変わる

クラスターをもう一度作成してみてください。

クラスターの状態は [アクティブ] であるが、プロデューサーがデータを送信できないか、コンシューマーがデータを受信できない

- クラスターの作成が成功した (クラスターの状態は `ACTIVE`) がデータの送受信ができない場合は、プロデューサーおよびコンシューマーのアプリケーションがクラスターにアクセスできることを確認してください。詳細については、the section called “クライアントマシンを作成する” のガイダンスを参照してください。

- プロデューサーとコンシューマーにクラスターへのアクセスがある場合で、データの生成と消費に問題が生じる場合、原因は KAFKA-7697 である可能性があります。これは、Apache Kafka バージョン 2.1.0 に影響し、1 つ以上のブローカーでデッドロックを引き起こす可能性があります。このバグの影響を受けない Apache Kafka 2.2.1 への移行を検討してください。統合する方法について詳細は、「the section called “Amazon MSK クラスターへの移行”」を参照してください。

AWS CLI は Amazon MSK を認識しません

AWS CLI がインストールされているが、Amazon MSK コマンドを認識しない場合は、 AWS CLI を最新バージョンにアップグレードします。のアップグレード方法の詳細については AWS CLI、「 のインストール AWS Command Line Interface」を参照してください。を使用して Amazon MSK コマ

ンド AWS CLI を実行する方法については、「」を参照してくださいthe section called “主な機能と概念”。

パーティションがオフラインになるか、レプリカが同期しない

これは、ディスクの容量不足の症状である可能性があります。「the section called “ディスク容量が不足している”」を参照してください。

ディスク容量が不足している

ディスク容量の管理に関するベストプラクティスについては、「the section called “ディスク容量のモニタリング”」および「the section called “データ保持パラメータの調整”」を参照してください。

メモリが不足している

`MemoryUsed` メトリックが高くなったり、`MemoryFree` が低くなったりしても、問題があるわけではありません。Apache Kafka はできるだけ多くのメモリを使用し、最適に管理するように設計されています。

プロデューサーが NotLeaderForPartitionException を取得する

これは時折発生する一時的なエラーです。プロデューサーの `retries` の設定パラメータを現在の値よりも大きい値に設定してください。

レプリケート不足のパーティション (URP) 数がゼロより大きい

`UnderReplicatedPartitions` メトリックは監視すべき重要なメトリックの 1 つです。正常な MSK クラスターでは、このメトリックの値は 0 です。ゼロより大きい場合は、次のいずれかの理由が考えられます。

- `UnderReplicatedPartitions` がスパイクの場合、着信トラフィックと発信トラフィックを処理するために適切なサイズでクラスターがプロビジョニングされていないことが問題である可能性があります。「the section called “標準ブローカーのベストプラクティス”」を参照してください。
- トラフィックが少ない期間も含めて `UnderReplicatedPartitions` が一貫して 0 より大きい場合、ブローカーにトピックアクセスを許可しない制限付き ACL が設定されていることが問題である可能性があります。パーティションを複製するには、ブローカーに READ トピックと DESCRIBE トピックの両方を許可する必要があります。DESCRIBE は、デフォルトで READ 権限が付与されます。ACL の設定については、Apache Kafka のドキュメントの「認可と ACL」を参照してください。

クラスターには __amazon_msk_canary と __amazon_msk_canary_state というトピックがあります

MSK クラスターのトピックの中に、`__amazon_msk_canary` および `__amazon_msk_canary_state` という名前のものがあることがあります。これらは Amazon MSK によって作成される、クラスターのヘルスと診断メトリクスに使用される内部トピックです。これらのトピックのサイズはごくわずかで、削除できません。

パーティションレプリケーションが失敗する

CLUSTER_ACTIONS に ACL が設定されていないことを確認してください。

パブリックアクセスが有効になっているクラスターにアクセスできない

クラスターでパブリックアクセスが有効になっていても、インターネットからアクセスできない場合は、以下の手順を実行します。

1. クラスターのセキュリティグループのインバウンドルールで、お使いのIP アドレスとクラスターのポートが許可されていることを確認します。クラスターポート番号のリストについては、the section called “ポート情報” を参照してください。また、セキュリティグループのアウトバウンドルールでアウトバウンド通信が許可されていることを確認してください。セキュリティグループのインバウンドおよびアウトバウンドルールについてのより詳しい情報は、Amazon VPC ユーザーガイドのVPC のセキュリティグループを参照してください。
2. お使いの IP アドレスとクラスターのポートが、クラスターの VPC ネットワーク ACL のインバウンドルールで許可されていることを確認してください。セキュリティグループとは異なり、ネットワーク ACL はステートレスです。つまり、インバウンドルールとアウトバウンドルールの両方を設定する必要があります。アウトバウンドルールでは、IP アドレスへのすべてのトラフィック (ポート範囲: 0 ~ 65535) を許可します。より詳細な情報は、Amazon VPC ユーザーガイドのルールの追加と削除を参照してください。
3. クラスターにアクセスする際は、パブリックアクセスブートストラップブローカーの文字列を使用してください。パブリックアクセスが有効な MSK クラスターには、パブリックアクセス用とAWS内部からのアクセス用の 2 つの異なるブートストラップブローカーの文字列があります。詳細については、「the section called “を使用してブートストラップブローカーを取得する AWS Management Console”」を参照してください。

内からクラスターにアクセスできない AWS: ネットワークの問題

Apache Kafka アプリケーションが MSK クラスターと正常に通信できない場合は 、まず次の接続テストを実行します。

1. 「the section called “ブートストラップブローカーを取得する”」で説明されている方法のいずれかを使用して、ブートストラップブローカーのアドレスを取得します。
2. 次のコマンドで、*bootstrap-broker* を前のステップで取得したブローカーアドレスの 1 つに置き換えます。クラスターが TLS 認証を使用するように設定されている場合は、*port-number* を 9094 に置き換えます。クラスターで TLS 認証を使用しない場合は、*port-number* を 9092 に置き換えます。クライアントマシンからコマンドを実行します。

   ```
   telnet bootstrap-broker port-number
   ```

   ここで、ポート番号は次の値になります。

   - クラスターが TLS 認証を使用するように設定されている場合、9094。
   - クラスターで TLS 認証を使用しない場合、9092。
   - パブリックアクセスが有効になっている場合は、別のポート番号が必要です。

   クライアントマシンからコマンドを実行します。
3. すべてのブートストラップブローカーに対して上記のコマンドを繰り返します。

クライアントマシンがブローカーにアクセスできる場合は、接続に問題はありません。この場合、Apache Kafka クライアントが正しく設定されているかどうかを確認するには、次のコマンドを実行します。*bootstrap-brokers* を取得するには、「the section called “ブートストラップブローカーを取得する”」で説明されているいずれかの方法を使用します。*topic* をトピックの名前に置き換えます。

```
<path-to-your-kafka-installation>/bin/kafka-console-producer.sh --broker-
list bootstrap-brokers --producer.config client.properties --topic topic
```

前のコマンドが成功した場合は、クライアントが正しくセットアップされていることを意味します。それでもアプリケーションから生成および使用できない場合は、アプリケーションレベルで問題をデバッグします。

クライアントマシンがブローカーにアクセスできない場合、以下のサブセクションで、クライアントマシンの設定に基づくガイダンスを参照してください。

同じ VPC 内の Amazon EC2 クライアントおよび MSK クラスター

クライアントマシンが MSK クラスターと同じ VPC にある場合、クラスターのセキュリティグループに、クライアントマシンのセキュリティグループからのトラフィックを受け入れるインバウンドルールがあることを確認します。これらのルールの設定については、セキュリティグループルールを参照してください。クラスターと同じ VPC にある Amazon EC2 インスタンスからクラスターにアクセスする方法の例については、「the section called “はじめに”」を参照してください。

異なる VPC 内の Amazon EC2 クライアントと MSK クラスター

クライアントマシンとクラスターが 2 つの異なる VPC にある場合は、次のことを確認します。

- 2 つの VPC がピア接続されている。
- ピア接続のステータスはアクティブである。
- 2 つの VPC のルートテーブルが正しく設定されている。

VPC ピア接続の詳細については、「VPC ピア接続の使用」を参照してください。

オンプレミスクライアント

を使用して MSK クラスターに接続するように設定されたオンプレミスクライアントの場合は AWS VPN、以下を確認してください。

- VPN 接続のステータスは UP である。VPN 接続のステータスを確認する方法については、「VPN トンネルの現在のステータスを確認するにはどうすればよいですか?」を参照してください。
- クラスターの VPC のルートテーブルには、ターゲットが `Virtual private gateway(vgw-xxxxxxxx)` 形式であるオンプレミス CIDR のルートが含まれています。
- MSK クラスターのセキュリティグループは、ポート 2181、ポート 9092 (クラスターがプレーンテキストのトラフィックを受け入れる場合)、ポート 9094 (クラスターが TLS で暗号化されたトラフィックを受け入れる場合) でトラフィックを許可します。

AWS VPN トラブルシューティングのガイダンスの詳細については、「クライアント VPN のトラブルシューティング」を参照してください。

AWS Direct Connect

クライアントが を使用している場合は AWS Direct Connect、「トラブルシューティング AWS Direct Connect」を参照してください。

上記のトラブルシューティングガイダンスで問題が解決しない場合は、ファイアウォールがネットワークトラフィックをブロックしていないことを確認します。さらにデバッグを行う際は、`tcpdump` や `Wireshark` などのツールを使用してトラフィックを分析し、トラフィックが MSK クラスターに到達していることを確認してください。

認証の失敗: 接続が多すぎる

`Failed authentication ... Too many connects` エラーは、1 つ以上の IAM クライアントが過剰な頻度で接続を試みているため、ブローカーが自身を保護していることを示しています。ブローカーがより高頻度で新しい IAM 接続を受け入れることができるように、`reconnect.backoff.ms` 設定パラメータを増やすことができます。

ブローカーごとの新規接続頻度の制限について詳しくは、「Amazon MSK クォータ」ページを参照してください。

認証の失敗: セッションが短すぎる

`Failed authentication ... Session too short` エラーは、クライアントが期限切れになる IAM 認証情報を使用してクラスターに接続しようとすると発生します。IAM 認証情報の更新方法を確認してください。ほとんどの場合、認証情報がセッションの有効期限に近づいているため、サーバー側で問題が発生し、認証が失敗します。

MSK サーバーレス: クラスターの作成に失敗する

MSK サーバーレスクラスターを作成しようとしてワークフローが失敗した場合、VPC エンドポイントを作成するアクセス許可がない可能性があります。`ec2:CreateVpcEndpoint` アクションを許可して、管理者によって VPC エンドポイントを作成するアクセス許可が付与されていることを確認します。

すべての Amazon MSK アクションを実行するために必要なアクセス許可の完全なリストについては、「AWS マネージドポリシー: AmazonMSKFullAccess」を参照してください。

MSK 設定で KafkaVersionsList を更新できない

AWS::MSK::Configuration リソースの KafkaVersionsList プロパティを更新すると、次のエラーで更新が失敗します。

```
Resource of type 'AWS::MSK::Configuration' with identifier '<identifierName>' already
 exists.
```

KafkaVersionsList プロパティを更新すると、 は古い設定を削除する前に、更新されたプロパティを使用して新しい設定 AWS CloudFormation を再作成します。新しい設定が既存の設定と同じ名前を使用しているため、 AWS CloudFormation スタックの更新は失敗します。このような更新には、リソースの置き換えが必要です。を正常に更新するにはKafkaVersionsList、同じオペレーションで Name プロパティも更新する必要があります。

さらに、設定が AWS Management Console または を使用して作成されたクラスターにアタッチされている場合は AWS CLI、リソースの削除試行の失敗を防ぐために、設定リソースに以下を追加します。

```
UpdateReplacePolicy: Retain
```

更新が成功したら、Amazon MSK コンソールに移動し、古い設定を削除します。MSK 設定の詳細については、「Amazon MSK プロビジョンド設定」を参照してください。

Standard ブローカーと Express ブローカーのベストプラクティス

このセクションでは、スタンダードブローカーと Express ブローカーについて従うべきベストプラクティスについて説明します。Amazon MSK Replicator のベストプラクティスについては、「MSK レプリケーターの使用に関するベストプラクティス」を参照してください。

トピック

- 標準ブローカーのベストプラクティス
- Express ブローカーのベストプラクティス
- Apache Kafka クライアントのベストプラクティス

標準ブローカーのベストプラクティス

このトピックでは、Amazon MSK を使用する際に従うべきいくつかのベストプラクティスの概要を説明します。Amazon MSK Replicator のベストプラクティスについては、「MSK レプリケーターの使用に関するベストプラクティス」を参照してください。

クライアント側の考慮事項

アプリケーションの可用性とパフォーマンスは、サーバー側の設定だけでなく、クライアントの設定にも依存します。

- 高可用性のためにクライアントを設定します。Apache Kafka のような分散システムで、信頼性と耐障害性に優れたメッセージングインフラストラクチャを維持するためには、高可用性の確保が不可欠です。ブローカーは、アップグレード、パッチ適用、ハードウェア障害、ネットワークの問題など、計画的なイベントと計画外のイベントの両方でオフラインになります。Kafka クラスターには、オフラインブローカーに対する寛容性があり、このため Kafka クライアントでは、ブローカーのフェイルオーバーを円滑に処理する必要があります。の詳細については、「」を参照してくださいApache Kafka クライアントのベストプラクティス。
- クライアント接続文字列に、各アベイラビリティーゾーンのブローカーが少なくとも 1 つ含まれていることを確認してください。クライアントの接続文字列に複数のブローカーが含まれていると、特定のブローカーが更新のためにオフラインになった場合にフェイルオーバーができるようになります。複数のブローカーで接続文字列を取得する方法については、「Amazon MSK クラスターのブートストラップブローカーを取得する」を参照してください。
- パフォーマンステストを実行して、クライアント設定でパフォーマンス目標を達成できることを確認します。

サーバー側の考慮事項

クラスターの適切なサイズ設定: 標準ブローカーあたりのパーティション数

次の表は、標準ブローカーあたりの推奨パーティション数 (リーダーレプリカとフォロワーレプリカを含む) を示しています。推奨される数のパーティションは強制されず、プロビジョニングされたすべてのトピックパーティションにトラフィックを送信するシナリオのベストプラクティスです。

ブローカーサイズ	推奨されるブローカーあたりのパーティション数 (リーダーとフォロワーのレプリカを含む)	更新オペレーションをサポートするパーティションの最大数
`kafka.t3.small`	300	300
`kafka.m5.large` 、または `kafka.m5.xlarge`	1,000	1500

ブローカーサイズ	推奨されるブローカーあたりのパーティション数 (リーダーとフォロワーのレプリカを含む)	更新オペレーションをサポートするパーティションの最大数
kafka.m5.2xlarge	2000	3000
kafka.m5.4xlarge 、kafka.m5.8xlarge 、kafka.m5.12xlarge 、kafka.m5.16xlarge 、または kafka.m5.24xlarge	4000	6000
kafka.m7g.large 、または kafka.m7g.xlarge	1,000	1500
kafka.m7g.2xlarge	2000	3000
kafka.m7g.4xlarge 、kafka.m7g.8xlarge 、kafka.m7g.12xlarge 、または kafka.m7g.16xlarge	4000	6000

パーティション数が高く、スループットが低いユースケースで、すべてのパーティションにトラフィックを送信していない場合は、クラスターがパーティション数が多いほど正常であることを検証するために十分なテストとパフォーマンステストを実行している限り、ブローカーごとにより多くのパーティションをパックできます。ブローカーあたりのパーティション数が最大許容値を超え、クラスターが過負荷になった場合、次のオペレーションを実行できなくなります。

- クラスター設定の更新
- クラスターをより小さいブローカータイプに更新する
- SASL/SCRAM 認証を持つクラスターに AWS Secrets Manager シークレットを関連付ける

パーティションの数が多くなると、CloudWatch および Prometheus スクレイピングで Kafka メトリクスの欠落を起こすことがあります。

パーティション数の選択に関するガイダンスについては、「Apache Kafka Supports 200K Partitions Per Cluster」を参照してください。また、独自のテストを実行して、ブローカーに適したサイズを決定することをお勧めします。さまざまなブローカータイプの詳細な情報については、「the section called “ブローカータイプ”」を参照してください。

クラスターの適切なサイズ設定: クラスターあたりの標準ブローカーの数

MSK プロビジョンドクラスターの標準ブローカーの適切な数を決定し、コストを理解するには、MSK サイジングと料金表を参照してください。このスプレッドシートは、同様のセルフマネージド型の EC2-based Apache Kafka クラスターと比較して、MSK プロビジョンドクラスターのサイズと Amazon MSK の関連コストの見積もりを提供します。スプレッドシートの入力パラメータの詳細については、パラメータの説明にカーソルを合わせてください。このシートで提供される見積もりは保守的であり、新しい MSK プロビジョンドクラスターの開始点を提供します。クラスターのパフォーマンス、サイズ、コストはユースケースによって異なるため、実際のテストで検証することが推奨されます。

基盤となるインフラストラクチャが Apache Kafka のパフォーマンスにどのように影響するかを理解するには、 AWS ビッグデータブログの「Apache Kafka クラスターのサイズを適正化してパフォーマンスとコストを最適化するためのベストプラクティス」を参照してください。このブログ記事では、スループット、可用性、レイテンシーの要件を満たすようにクラスターのサイズを設定する方法について説明しています。また、スケールアップとスケールアウトのタイミングなどの質問に対する回答や、本番クラスターのサイズを継続的に検証する方法に関するガイダンスも提供します。階層型ストレージベースのクラスターの詳細については、「Amazon MSK 階層型ストレージを使用して本稼働ワークロードを実行するためのベストプラクティス」を参照してください。

m5.4xl、m7g.4xl、またはそれ以上のインスタンスでクラスタースループットを最適化する

m5.4xl、m7g.4xl 以上のインスタンスを使用する場合、num.io.threads および num.network.threads 設定を調整することで、MSK プロビジョンドクラスターのスループットを最適化できます。

Num.io.threads は、スタンダードブローカーがリクエストの処理に使用するスレッドの数です。スレッド数の追加 (最大値はインスタンスサイズとしてサポートされている CPU コアの数) をすることで、クラスターのスループットを向上させることができます。

Num.network.threads は、Standard ブローカーがすべての受信リクエストを受信し、レスポンスを返すために使用するスレッドの数です。ネットワークスレッドは、着信リクエストをリクエストキューに入れ、io.threads で処理します。num.network.threads をインスタンスサイズとしてサポートされている CPU コア数の半分に設定すると、新しいインスタンスサイズを最大限に活用できます。

> **Important**
>
> num.network.threads を増やす場合は、先に num.io.threads を増やす必要があります。そうしないと、キューの飽和による輻輳が発生する可能性があります。

推奨される設定

インスタンスサイズ	num.io.threads の推奨値	num.network.threads の推奨値
m5.4xl	16	8
m5.8xl	32	16
m5.12xl	48	24
m5.16xl	64	32
m5.24xl	96	48
m7g.4xlarge	16	8
m7g.8xlarge	32	16
m7g.12xlarge	48	24
m7g.16xlarge	64	32

トピック ID の不一致の問題を回避するための最新の Kafka AdminClient の使用

2.8.0 より前の Kafka AdminClient バージョンと フラグを使用して Kafka バージョン 2.8.0 以降を使用する MSK プロビジョニング済みクラスターのトピックパーティション`--zookeeper`を増加または再割り当てすると、トピックの ID が失われます (エラー: はパーティションのトピック ID と一致しません) 。`--zookeeper` フラグは Kafka 2.5 で非推奨になり、Kafka 3.0 以降では削除されているので注意してください。「Upgrading to 2.5.0 from any version 0.8.x through 2.4.x」を参照してください。

トピック ID の不一致を回避するには、Kafka 管理オペレーションに Kafka クライアントバージョン 2.8.0 以降を使用してください。または、2.5 以降のクライアントでは、`--zookeeper` フラグの代わりに `--bootstrap-servers` フラグを使用できます。

高可用性クラスターの構築

更新中 (ブローカーサイズや Apache Kafka バージョンを更新する場合など) または Amazon MSK がブローカーを置き換えるときに、MSK プロビジョンドクラスターを高可用性にするには、次の推奨事項を使用します。

- 3-AZ クラスターを設定します。
- レプリケーション係数 (RF) が 3 以上であることを確認します。RF が 1 の場合、ローリング更新中にパーティションがオフラインになる可能性があり、RF が 2 の場合、データが失われる可能性があることに注意してください。
- 最小同期レプリカ (minISR) を最大で RF - 1 に設定します。RF と同等の minISR は、ローリング更新中のクラスターへの生成を防止できます。minISR が 2 の場合、レプリカが 1 つオフラインになると、3 方向のレプリケートされたトピックを使用できるようになります。

CPU 使用率をモニタリングする

Amazon MSK では、ブローカー (と定義`CPU User + CPU System`) の CPU 使用率を 60% 以下に維持することを強くお勧めします。これにより、クラスターは、ブローカーの障害、パッチ適用、ローリングアップグレードなどの運用イベントを処理するのに十分な CPU ヘッドルームを保持できます。

Apache Kafka は、必要に応じてクラスター内のブローカー間で CPU 負荷を再分散できます。例えば、Amazon MSK がブローカーの障害を検出して復旧すると、パッチ適用などの自動メンテナンスが実行されます。同様に、ユーザーがブローカーサイズの変更またはバージョンアップグレードをリクエストすると、Amazon MSK は一度に 1 つのブローカーをオフラインにするローリングワークフローを開始します。リードパーティションを持つブローカーがオフラインになると、Apache Kafka はパーティションのリーダーシップを再割り当てして、クラスター内の他のブローカーに作業を再配布します。このベストプラクティスに従うことで、これらの運用イベントを許容するのに十分な CPU ヘッドルームを確保できます。

> **Note**
>
> CPU 使用率をモニタリングするときは、合計 CPU 使用率に `CPU User`と を超える が含まれていることに注意してください`CPU System`。、、`iowait`、`irq`、`softirq`などの他のカテゴリも`steal`、全体的な CPU アクティビティに影響します。したがって、CPU アイドルは常に と等しくなるわけではありません`100% - CPU User - CPU System`。

Amazon CloudWatch メトリクス計算を使用して複合メトリクス (CPU User + CPU System) を作成し、平均使用量が 60% を超えたときにトリガーするようにアラームを設定できます。トリガーされたら、次のいずれかのオプションを使用してクラスターをスケーリングすることを検討してください。

- オプション 1 (推奨): 次に大きいサイズにブローカーサイズを更新します。例えば、現在のサイズが kafka.m5.large の場合、kafka.m5.xlarge を使用するようにクラスターを更新します。クラスター内のブローカーサイズを更新すると、Amazon MSK はブローカーをローリング方式でオフラインにし、パーティションのリーダーシップを他のブローカーに一時的に再割り当てることに注意してください。サイズの更新には、通常、ブローカーごとに 10 ～ 15 分かかります。
- オプション 2: ラウンドロビン書き込みを使用するプロデューサーからすべてのメッセージを取り込んでいる (つまり、メッセージにキーが設定されておらず、コンシューマーにとって順序は重要ではない) トピックがある場合は、ブローカーを追加してクラスターを拡張します。また、スループットが最も高い既存のトピックにパーティションを追加します。次に、kafka-topics.sh --describe を使用して、新しく追加されたパーティションが新しいブローカーに割り当てられていることを確認します。前のオプションと比較したこのオプションの主な利点は、リソースとコストをよりきめ細かく管理できることです。さらに、CPU ロードが 60% を大幅に超える場合は、このオプションを使用できます。これは、この形式のスケーリングでは通常、既存のブローカーのロードが増加しないためです。
- オプション 3: ブローカーを追加して MSK プロビジョンドクラスターを拡張し、 という名前のパーティション再割り当てツールを使用して既存のパーティションを再割り当てしますkafka-reassign-partitions.sh。ただし、このオプションを使用する場合、パーティションが再割り当てされた後、クラスターはブローカーからブローカーにデータを複製するためにリソースを費やす必要があります。前の 2 つのオプションと比較すると、これにより、最初はクラスターのロードが大幅に増加する可能性があります。その結果、レプリケーションによって追加の CPU ロードとネットワーク トラフィックが発生するため、CPU 使用率が 70% を超える場合、Amazon MSK はこのオプションの使用を推奨しません。Amazon MSK は、前の2つのオプションが実行可能でない場合にのみ、このオプションを使用することをお勧めします。

その他の推奨事項：

- ロード ディストリビューションのプロキシとして、ブローカーごとの合計 CPU 使用率をモニタリングします。ブローカーの CPU 使用率が一貫して不均一である場合は、ロードがクラスター内で均等に配信されていないことを示している可能性があります。Cruise Control を使用して、パーティション割り当てを介して負荷分散を継続的に管理することをお勧めします。

- 生成および消費レイテンシーをモニタリングします。レイテンシーの生成と消費は、CPU 使用率に比例して増加する可能性があります。
- JMX スクレイプ間隔: Prometheus 機能を使用してオープンモニタリングを有効にする場合は、Prometheus ホスト設定 (prometheus.yml) で 60 秒以上のスクレイプ間隔 (scrape_interval: 60s) を使用することが推奨されます。スクレイプ間隔を短くすると、クラスターの CPU 使用率が高くなる可能性があります。

ディスク容量のモニタリング

メッセージ用のディスクスペースが不足しないようにするには、KafkaDataLogsDiskUsed メトリクスを監視する CloudWatch アラームを作成します。このメトリクスの値が 85% に達するか超える場合は、次の 1 つ以上のアクションを実行します。

- the section called “クラスターの自動スケーリング” を使用します。「the section called “手動スケーリング”」の説明に従って、ブローカーストレージを手動で増やすこともできます。
- メッセージの保持期間またはログサイズを減らします。これを行う方法については、the section called “データ保持パラメータの調整” を参照してください。
- 未使用のトピックを削除します。

アラームの設定と使用方法については、「Amazon CloudWatch アラームの使用」を参照してください。Amazon MSK メトリクスの完全なリストについては「the section called “クラスターをモニタリングする”」を参照してください。

データ保持パラメータの調整

メッセージを消費しても、ログからは削除されません。定期的にディスク容量を解放するには、保持期間 (メッセージをログに保持する期間) を明示的に指定できます。保存ログのサイズを指定することもできます。保持期間または保持ログのサイズのいずれかに達すると、Apache Kafka は、ログから非アクティブなセグメントの削除を開始します。

クラスターレベルで保持ポリシーを指定するには、log.retention.hours、log.retention.minutes、log.retention.ms、または log.retention.bytes のいずれかまたは複数のパラメータを設定します。詳細については、「the section called “カスタム Amazon MSK 設定”」を参照してください。

トピックレベルで保持パラメータを指定することもできます。

- トピックごとに保持期間を指定するには、次のコマンドを使用します。

```
kafka-configs.sh --bootstrap-server $bs --alter --entity-type topics --entity-
name TopicName --add-config retention.ms=DesiredRetentionTimePeriod
```

- トピックごとに保持ログのサイズを指定するには、次のコマンドを使用します。

```
kafka-configs.sh --bootstrap-server $bs --alter --entity-type topics --entity-
name TopicName --add-config retention.bytes=DesiredRetentionLogSize
```

トピックレベルで指定する保持パラメータは、クラスターレベルのパラメータよりも優先されます。

不正シャットダウン後のログ復旧の高速化

不正シャットダウンの後、ブローカーはログ復旧を実行するため、再起動に時間がかかることがあります。デフォルトでは、Kafka はログディレクトリごとに 1 つのスレッドのみを使用してこの復旧を実行します。例えば、数千のパーティションがある場合、ログ復旧が完了するまでに数時間かかる可能性があります。ログ復旧を高速化するには、設定プロパティ `num.recovery.threads.per.data.dir` を使用してスレッド数を増やすことが推奨されます。これを CPU コアの数に設定できます。

Apache Kafka メモリのモニタリング

Apache Kafka が使用するメモリをモニタリングすることが推奨されます。そうしないと、クラスターを使用できなくなる可能性があります。

Apache Kafka が使用するメモリ量を判別するために、`HeapMemoryAfterGC` メトリクスをモニタリングできます。`HeapMemoryAfterGC` は、ガベージコレクション後に使用されている合計ヒープメモリの割合 (%) です。`HeapMemoryAfterGC` が 60% を上回った場合にアクションを実行する CloudWatch アラームを作成することが推奨されます。

メモリ使用量を減らすために実行できるステップはさまざまです。これらは Apache Kafka の設定方法によって異なります。例えば、トランザクションメッセージ配信を使用する場合、Apache Kafka 設定の `transactional.id.expiration.ms` 値を `604800000` ミリ秒から `86400000` ミリ秒に (7 日から 1 日に) 減らすことができます。これにより、各トランザクションのメモリフットプリントが減ります。

MSK 以外のブローカーを追加しない

ZooKeeper ベースの MSK プロビジョンドクラスターの場合、Apache ZooKeeper コマンドを使用してブローカーを追加すると、これらのブローカーは MSK プロビジョンドクラスターに追加され

ず、Apache ZooKeeper にはクラスターに関する誤った情報が含まれます。これにより、データが失われる可能性があります。サポートされている MSK プロビジョンドクラスターオペレーションについては、「」を参照してくださいthe section called “主な機能と概念”。

転送中の暗号化を有効にする

転送中の暗号化とその有効化方法については、「the section called “転送中の Amazon MSK 暗号化”」を参照してください。

パーティションの再割り当て

パーティションを同じ MSK プロビジョンドクラスター上の異なるブローカーに移動するには、 という名前のパーティション再割り当てツールを使用できますkafka-reassign-partitions.sh。安全なオペレーションのために、1 回のkafka-reassign-partitions呼び出しで 10 個を超えるパーティションを再割り当てしないことをお勧めします。例えば、新しいブローカーを追加してクラスターを拡張したり、ブローカーの削除のためにパーティションを移動したりすると、新しいブローカーにパーティションを再割り当てすることで、そのクラスターを再分散できるようになります。MSK プロビジョンドクラスターにブローカーを追加する方法については、「」を参照してくださいthe section called “クラスターを展開する”。MSK プロビジョンドクラスターからブローカーを削除する方法については、「」を参照してくださいthe section called “ブローカーを削除する”。パーティション再割り当てツールの詳細については、Apache Kafka のドキュメントの「クラスターの拡張」を参照してください。

Express ブローカーのベストプラクティス

このトピックでは、Express ブローカーを使用する際に従うべきいくつかのベストプラクティスの概要を説明します。Express ブローカーには、高可用性と耐久性のために事前設定されています。デフォルトでは、データは 3 つのアベイラビリティーゾーンに分散され、レプリケーションは常に 3 に設定され、同期内の最小レプリカは常に 2 に設定されます。ただし、クラスターの信頼性とパフォーマンスを最適化するために考慮すべき要素はいくつかあります。

クライアント側の考慮事項

アプリケーションの可用性とパフォーマンスは、サーバー側の設定だけでなく、クライアント設定にも依存していました。

- 高可用性のためにクライアントを設定します。Apache Kafka のような分散システムで、信頼性と耐障害性に優れたメッセージングインフラストラクチャを維持するためには、高可用性の確保が不可欠です。ブローカーは、アップグレード、パッチ適用、ハードウェア障害、ネットワークの問

題など、計画的なイベントと計画外のイベントの両方でオフラインになります。Kafka クラスターには、オフラインブローカーに対する寛容性があり、このため Kafka クライアントでは、ブローカーのフェイルオーバーを円滑に処理する必要があります。詳細については、「Apache Kafka クライアントのベストプラクティスの推奨事項」を参照してください。

- パフォーマンステストを実行して、ピーク負荷時にブローカーを再起動しても、クライアント設定でパフォーマンス目標を達成できることを確認します。クラスター内のブローカーは、MSK コンソールまたは MSK APIs を使用して再起動できます。

サーバー側の考慮事項

トピック

- クラスターの適切なサイズ設定: クラスターあたりのブローカー数
- CPU 使用率をモニタリングする
- クラスターの適切なサイズ設定: Express ブローカーあたりのパーティション数
- 接続数をモニタリングする
- パーティションの再割り当て

クラスターの適切なサイズ設定: クラスターあたりのブローカー数

Express ベースのクラスターのブローカー数の選択は簡単です。各 Express ブローカーには、入出力用に定義されたスループット容量が付属しています。このスループットキャパシティをクラスターのサイズ設定の主な手段として使用する必要があります (次に、以下で説明するパーティションや接続数などの他の要因を考慮してください)。

たとえば、ストリーミングアプリケーションに 45 MBps のデータ入力 (書き込み) と 90 MBps のデータ出力 (読み取り) 容量が必要な場合は、スループットのニーズを満たすために 3 つの express.m7g.large ブローカーを使用できます。各 express.m7g.large ブローカーは、15 MBps の進入と 30 MBps の進入を処理します。Express ブローカーのサイズごとの推奨スループット制限については、次の表を参照してください。スループットが推奨制限を超えると、パフォーマンスが低下する可能性があるため、トラフィックを減らすか、クラスターをスケールする必要があります。スループットが推奨制限を超え、ブローカーごとのクォータに達すると、MSK はクライアントトラフィックをスロットリングして、それ以上の過負荷を防ぎます。

MSK サイジングと料金表のスプレッドシートを使用して、複数のシナリオを評価し、パーティション数などの他の要因を検討することもできます。

ブローカーあたりの推奨最大スループット

インスタンスサイズ	イングレス (MBps)	出力 (MBps)
express.m7g.large	15.6	31.2
express.m7g.xlarge	31.2	62.5
express.m7g.2xlarge	62.5	125.0
express.m7g.4xlarge	124.9	249.8
express.m7g.8xlarge	250.0	500.0
express.m7g.12xlarge	375.0	750.0
express.m7g.16xlarge	500.0	1000.0

CPU 使用率をモニタリングする

ブローカー (CPU ユーザー + CPU システムとして定義) の合計 CPU 使用率を 60% 以下に維持することをお勧めします。クラスターの合計 CPU の少なくとも 40% が利用可能である場合、Apache Kafka は必要に応じてクラスター内のブローカー間で CPU 負荷を再分散できます。これは、計画的なイベントまたは計画外のイベントが原因で必要になる場合があります。計画されたイベントの例は、MSK がクラスター内のブローカーを一度に 1 つずつ再起動して更新するクラスターバージョンアップグレードです。予期しないイベントの例としては、ブローカーのハードウェア障害、または最悪の場合は AZ 内のすべてのブローカーが影響を受ける AZ 障害などがあります。パーティションリードレプリカを持つブローカーがオフラインになると、Apache Kafka はパーティションリーダーシップを再割り当てして、クラスター内の他のブローカーに作業を再配布します。このベストプラクティスに従うことで、このような運用イベントを許容するのに十分な CPU ヘッドルームをクラスター内に確保できます。

Amazon CloudWatch ユーザーガイドの CloudWatch メトリクスで数式を使用して、CPU ユーザー + CPU システムである複合メトリクスを作成できます。 Amazon CloudWatch 複合メトリクスの平均 CPU 使用率が 60% に達したときにトリガーされるアラームを設定します。このアラームがトリガーされたら、以下のいずれかのオプションを使用してクラスターをスケーリングします。

- オプション 1: ブローカーサイズを次に大きいサイズに更新します。クラスター内のブローカーサイズを更新すると、Amazon MSK はブローカーをローリング方式でオフラインにし、パーティションのリーダーシップを他のブローカーに一時的に再割り当てることに注意してください。

- オプション 2: ブローカーを追加してクラスターを拡張し、 という名前のパーティション再割り当てツールを使用して既存のパーティションを再割り当てしますkafka-reassign-partitions.sh。

その他の推奨事項

- ロード ディストリビューションのプロキシとして、ブローカーごとの合計 CPU 使用率をモニタリングします。ブローカーの CPU 使用率が一貫して一定でない場合、負荷がクラスター内に均等に分散されていない可能性があります。Cruise Control を使用して、パーティション割り当てを介して負荷分散を継続的に管理することをお勧めします。
- 生成および消費レイテンシーをモニタリングします。レイテンシーの生成と消費は、CPU 使用率に比例して増加する可能性があります。
- JMX スクレイプ間隔: Prometheus 機能でオープンモニタリングを有効にする場合は、Prometheus ホスト設定 (scrape_interval: 60s) に 60 秒以上のスクレイプ間隔 () を使用することをお勧めしますprometheus.yml。スクレイプ間隔を短くすると、クラスターの CPU 使用率が高くなる可能性があります。

クラスターの適切なサイズ設定: Express ブローカーあたりのパーティション数

パーティション数が多く、スループットが低いユースケースで、すべてのパーティションにトラフィックを送信していない場合は、クラスターがパーティション数が多いほど正常であることを検証するために十分なテストとパフォーマンステストを実行している限り、ブローカーごとにより多くのパーティションをパックできます。ブローカーあたりのパーティション数が最大許容値を超え、クラスターが過負荷になった場合、次のオペレーションを実行できなくなります。

- クラスター設定の更新
- クラスターをより小さいブローカータイプに更新する
- SASL/SCRAM 認証を持つクラスターに AWS Secrets Manager シークレットを関連付ける

多数のパーティションでクラスターをオーバーロードすると、CloudWatch および Prometheus スクレイピングで Kafka メトリクスが失われる可能性もあります。

パーティション数の選択に関するガイダンスについては、「Apache Kafka Supports 200K Partitions Per Cluster」を参照してください。また、独自のテストを実行して、ブローカーに適したサイズを決定することをお勧めします。さまざまなブローカータイプの詳細な情報については、「Amazon MSK のブローカーサイズ」を参照してください。

各 Express ブローカーの推奨パーティション数 (リーダーレプリカとフォロワーレプリカを含む) については、「」を参照してくださいExpress ブローカーパーティションクォータ。推奨される数のパーティションは強制されず、プロビジョニングされたすべてのトピックパーティションにトラフィックを送信するシナリオのベストプラクティスです。

接続数をモニタリングする

ブローカーへのクライアント接続は、メモリや CPU などのシステムリソースを消費します。認証メカニズムに応じて、 をモニタリングして、該当する制限の範囲内にあることを確認する必要があります。接続に失敗した場合の再試行を処理するには、クライアント側で `reconnect.backoff.ms` 設定パラメータを設定できます。たとえば、クライアントが 1 秒後に接続を再試行する場合は、`reconnect.backoff.ms` を に設定します`1000`。再試行の設定の詳細については、「Apache Kafka ドキュメント」を参照してください。

ディメンション	クォータ
ブローカーあたりの最大 TCP 接続数 (IAM アクセスコントロール)	3000
ブローカーあたりの最大 TCP 接続数 (IAM)	100/秒
ブローカーあたりの最大 TCP 接続数 (非 IAM)	MSK は、IAM 以外の認証の接続制限を適用しません。ただし、CPU やメモリの使用状況などの他のメトリクスをモニタリングして、過剰な接続が原因でクラスターが過負荷にならないようにする必要があります。

パーティションの再割り当て

パーティションを同じ MSK プロビジョンドクラスター上の異なるブローカーに移動するには、 という名前のパーティション再割り当てツールを使用できます`kafka-reassign-partitions.sh`。安全なオペレーションのために、1 回の`kafka-reassign-partitions`呼び出しで 20 個を超えるパーティションを再割り当てしないことをお勧めします。例えば、新しいブローカーを追加してクラスターを拡張したり、ブローカーの削除のためにパーティションを移動したりすると、新しいブローカーにパーティションを再割り当てすることで、そのクラスターを再分散できるようになります。MSK プロビジョンドクラスターにブローカーを追加する方法については、「」を参照してくださいthe section called “クラスターを展開する”。MSK プロビジョンドクラスターからブローカーを削除する方法については、「」を参照してくださいthe section called “ブローカーを削除する”。パー

ティション再割り当てツールの詳細については、Apache Kafka のドキュメントの「クラスターの拡張」を参照してください。

Apache Kafka クライアントのベストプラクティス

Apache Kafka と Amazon MSK を使用する場合、最適なパフォーマンスと信頼性を実現するためには、クライアントとサーバーの両方を最適に設定することが重要です。このガイドでは、Amazon MSK のクライアント側設定におけるベストプラクティスの推奨事項を提供します。

Amazon MSK Replicator のベストプラクティスについては、「MSK レプリケーターの使用に関するベストプラクティス」を参照してください。Standard および Express ブローカーのベストプラクティスについては、「」を参照してくださいStandard ブローカーと Express ブローカーのベストプラクティス。

トピック

- Apache Kafka クライアントの可用性
- Apache Kafka クライアントのパフォーマンス
- Kafka クライアントモニタリング

Apache Kafka クライアントの可用性

Apache Kafka のような分散システムで、信頼性と耐障害性に優れたメッセージングインフラストラクチャを維持するためには、高可用性の確保が不可欠です。ブローカーがオフラインになるイベントには、アップグレード、パッチ適用、ハードウェア障害、ネットワークの問題など、計画的なものと計画的でないものの両方があります。Kafka クラスターには、オフラインブローカーに対する寛容性があり、このため Kafka クライアントでは、ブローカーのフェイルオーバーを円滑に処理する必要があります。Kafka クライアントの高可用性を確保するために、以下のベストプラクティスが推奨されています。

プロデューサーの可用性

- ブローカーのフェイルオーバー中に失敗したメッセージの再送信をプロデューサーに指示するには、`retries` を設定します。ほとんどのユースケースでは、整数の最大値または同様の大きな値が推奨されます。これが適切でない場合、Kafka の高可用性が損なわれます。
- `delivery.timeout.ms` を設定すると、メッセージを送信してブローカーから確認応答を受信する間の、合計時間の上限が指定されます。これには、メッセージの有効期間についての、ビジネス要件を反映する必要があります。フェイルオーバーの操作を完了するのに十分な再試行ができるよ

うに、時間制限を十分高く設定します。ほとんどのユースケースでは、60 秒以上の値にすることをお勧めします。

- request.timeout.ms には、再送信を試みる前に 1 つのリクエストが待機する最大時間を設定します。ほとんどのユースケースでは、10 秒以上の値にすることをお勧めします。
- retry.backoff.ms を設定すると、再試行ストームと可用性への影響を回避するように再試行間の遅延を指定できます。ほとんどのユースケースでは、最小値の 200 ミリ秒にすることをお勧めします。
- acks=all では高い耐障害性を設定します。これは、ISR 内のすべてのパーティションで書き込みが確実に承認されるように、RF=3 および min.isr=2 のサーバー側設定と合わせる必要があります。1 つのブローカーのオフライン中、この min.isr は 2 に設定します。

コンシューマーの可用性

- 新規または再作成されたコンシューマーグループの場合、最初に auto.offset.reset を latest に設定します。これにより、トピック全体を消費することでクラスター負荷が追加されるリスクを回避できます。
- enable.auto.commit を使用する場合は auto.commit.interval.ms を設定します。ほとんどのユースケースでは、追加的な負荷のリスクを避けるために、最小値の 5 秒にすることをお勧めします。
- 一時的なエラー (例えば、サーキットブレーカーや指数バックオフのあるスリープなど) を処理するために、コンシューマーのメッセージ処理コード内に例外処理を実装します。これが適切でない場合、アプリケーションがクラッシュし、過剰な再分散が発生する可能性があります。
- トランザクションメッセージの読み取り方法を制御するには、isolation.level を設定します。

  デフォルトでは、常に暗黙的に read_uncommitted を設定することをお勧めします。これは、一部のクライアント実装には含まれていません。

  階層型ストレージを使用する場合は、read_uncommitted の値にすることをお勧めします。
- 最も近いレプリカリードを使用するには、client.rack を設定します。ネットワークトラフィックのコストとレイテンシーを最小限に抑えるために、az id を設定することをお勧めします。「ラック認識を使用して Amazon MSK コンシューマーのネットワークトラフィックコストを削減する」を参照してください。

コンシューマーの再分散

- session.timeout.ms には、実装された起動ジッターを含め、アプリケーションの起動時間より大きい値を設定します。ほとんどのユースケースでは、値を60 秒にすることをお勧めします。
- heartbeat.interval.ms を設定すると、グループコーディネーターがコンシューマーを正常と見なす方法をファインチューニングできます。ほとんどのユースケースでは、値を 10 秒にすることをお勧めします。
- コンシューマーがグループを離れるタイミングを認識するためにセッションタイムアウトに依存するのではなく、アプリケーション内でシャットダウンフックを設定し、SIGTERM 上でコンシューマーをクリーンに閉じます。Kstream アプリケーションでは、internal.leave.group.on.close の値に true を設定できます。
- group.instance.id には、コンシューマーグループ内の個別の値を設定します。ホスト名、タスク ID、または ポッド ID が理想的です。トラブルシューティング中、より確定的な動作をさせ、クライアント/サーバーログの相関関係を向上させるために、この設定は常に行うことをお勧めします。
- 平均デプロイ時間にに合った値を group.initial.rebalance.delay.ms に設定します。これにより、デプロイ中の継続的な再分散が停止します。
- スティッキーアサイナーを使用するように partition.assignment.strategy を設定します。StickyAssignor または CooperativeStickyAssignor が推奨されます。

Apache Kafka クライアントのパフォーマンス

Kafka クライアントの高パフォーマンスを保証するためには、以下のベストプラクティスが推奨されます。

プロデューサーのパフォーマンス

- バッチが満たされるまでのプロデューサーの待機時間を制御するには、linger.ms を設定します。バッチが小さいほど、一度に変換されるスレッドと I/O 操作が多くなるため、Kafka には計算コストがかかります。推奨される値は、次のようになります。

  低レイテンシーを含むすべてのユースケースについて、最小値は 5 ミリ秒です。

  ほとんどのユースケースでは、25 ミリ秒より大きい値にすることをお勧めします。

  低レイテンシーのユースケースで、ゼロの値を使用することはお勧めしません。(通常、値が 0 の場合には、IO オーバーヘッドが原因でレイテンシーが発生します)。

- クラスターに送信されるバッチサイズを制御するには、batch.size を設定します。これは、64KB または 128KB の値に増やすことをお勧めします。
- より大きなバッチサイズを使用する場合には、buffer.memory を設定します。ほとんどのユースケースでは、値を 64MB にすることをお勧めします。
- バイトの受信に使用される TCP バッファを制御するには send.buffer.bytes を設定します。レイテンシーの高いネットワークでプロデューサーを実行するときに、OS がこのバッファを管理できるようにするには、値を -1 にすることをお勧めします。
- バッチの圧縮を制御するには、compression.type を設定します。高レイテンシーネットワークでは、lz4 または zstd のいずれかでプロデューサーを実行することをお勧めします。

コンシューマーのパフォーマンス

- fetch.min.bytes を設定すると、有効になる最小フェッチサイズを制御して、フェッチ数とクラスター負荷を減らせます。

  すべてのユースケースで、最小値を 32 バイトにすることをお勧めします。

  ほとんどのユースケースでは、より大きな 128 バイトの値にすることをお勧めします。
- fetch.max.wait.ms を設定すると、fetch.min.bytes が無視されるまでコンシューマーが待機する時間が決定されます。ほとんどのユースケースでは、値を 1000ms にすることをお勧めします。
- 並列性と耐障害性を向上させるために、コンシューマーの数はパーティションの数と少なくとも等しくすることをお勧めします。状況によっては、低スループットトピックのパーティション数よりもコンシューマー数を減らすことを選択できます。
- バイトの受信に使用される TCP バッファを制御するには receive.buffer.bytes を設定します。高レイテンシーネットワークでコンシューマーを実行するときに、OS がこのバッファを管理できるようにするには、値を -1 にすることをお勧めします。

クライアント接続

接続ライフサイクルには、Kafka クラスターでの計算コストとメモリコストが発生します。一度に作成された接続が多すぎると負荷が発生し、Kafka クラスターの可用性に影響を与えることがあります。この可用性への影響により、アプリケーションでさらに多くの接続が作成され、それが直列的な障害を発生することにつながり、完全な停止に至ることがよくあります。多数の接続は、合理的なレートで作成した場合に実現できます。

接続作成の高いレートを管理するには、以下の緩和策をお勧めします。

- アプリケーションデプロイメカニズムがすべてのプロデューサー/コンシューマーを一度に再起動するのではなく、可能な限り小さなバッチで再起動するようにします。
- デベロッパーは、アプリケーションレイヤーにおいて管理者クライアント、プロデューサークライアント、またはコンシューマークライアントを作成する前に、ランダムジッター (ランダムスリープ) が実行されていることを保証してください。
- SIGTERM で接続を閉じる際には、ランダムなスリープを実行して、すべての Kafka クライアントが同時に閉じられないようにする必要があります。ランダムスリープのタイムアウトは、SIGKILL が発生する前の範囲である必要があります。

  Example 例 A (Java)

```
sleepInSeconds(randomNumberBetweenOneAndX);
                        this.kafkaProducer = new KafkaProducer<>(this.props);
```

  Example 例 B (Java)

```
Runtime.getRuntime().addShutdownHook(new Thread(() -> {
    sleepInSeconds(randomNumberBetweenOneAndTwentyFive);
    kafkaProducer.close(Duration.ofSeconds(5));
});
```

- デベロッパーは、アプリケーションレイヤーにおいて、クライアントがアプリケーションごとに 1 回だけシングルトンパターンで作成されることを保証してください。例えば、lambda を使用する場合、クライアントはメソッドハンドラーではなくグローバルスコープで作成されるようにします。
- 安定することを目標に、接続数をモニタリングすることをお勧めします。デプロイ時とブローカーフェイルオーバー中は、接続の作成/クローズ/シフトは正常です。

Kafka クライアントモニタリング

Kafka クライアントのモニタリングは、Kafka エコシステムの正常性と効率を維持するために不可欠です。Kafka の管理者、開発者、または運用チームメンバーのいずれであっても、計画的および計画外のイベント中のビジネスへの影響を把握するためには、クライアント側のメトリクスを有効にすることが極めて重要です。

好みのメトリクスキャプチャメカニズムを使用して、次のクライアント側メトリクスをモニタリングすることをお勧めします。

でサポートチケットを発行する場合は AWS、インシデント中に観察された異常値を含めます。また、エラー (警告ではない) を詳述したクライアントアプリケーションログのサンプルも含めます。

プロデューサーメトリクス

- byte-rate (バイトレート)
- record-send-rate (レコード送信レート)
- records-per-request-avg (要求あたりレコードの平均)
- acks-latency-avg (承認遅延の平均)
- request-latency-avg (要求遅延の平均)
- request-latency-max (要求遅延の最大)
- record-error-rate (レコードエラーレート)
- record-retry-rate (レコード再試行レート)
- error-rate (エラーレート)

> **Note**
>
> 再試行に伴う一時的なエラーは、リーダーのフェイルオーバーやネットワーク再送信などの一時的な問題を処理する Kafka のプロトコルの一部であるため、懸念の原因ではありません。 `record-send-rate` は、プロデューサーがまだ再試行を進めているかどうかを確認します。

コンシューマーメトリクス

- records-consumed-rate (レコード使用レート)
- bytes-consumed-rate (バイト使用レート)
- fetch-rate (フェッチレート)
- records-lag-max (レコードの最大時間差)
- record-error-rate (レコードエラーレート)
- fetch-error-rate (フェッチエラーレート)
- poll-rate (ポーリングレート)
- rebalance-latency-avg (リバランス遅延の平均)
- commit-rate (コミットレート)

> **Note**
>
> フェッチレートとコミットレートが高いと、クラスターに不要な負荷が発生します。リクエストの処理には大きなバッチが適しています。

一般的なメトリクス

- connection-close-rate (接続解除レート)
- connection-creation-rate (接続作成レート)
- connection-count (接続数)

> **Note**
>
> 接続の作成/終了が高いと、クラスターに不要な負荷が発生します。

MSK サーバーレスとは

> **Note**
>
> MSK サーバーレスは、米国東部 (オハイオ)、米国東部 (バージニア北部)、米国西部 (オレゴン)、カナダ (中部)、アジアパシフィック (ムンバイ)、アジアパシフィック (シンガポール)、アジアパシフィック (シドニー)、アジアパシフィック (東京)、アジアパシフィック (ソウル)、欧州 (フランクフルト)、欧州 (ストックホルム)、欧州 (アイルランド)、欧州 (パリ)、および欧州 (ロンドン) の各リージョンで利用できます。

MSK サーバーレス は Amazon MSK のクラスタータイプで、クラスターの容量を管理およびスケールすることなく Apache Kafka を実行することができます。トピック内のパーティションを管理しながら容量を自動的にプロビジョニングおよびスケーリングするため、ユーザーはクラスターの適切なサイジングやスケーリングについて考えることなく、データをストリーミングできます。MSK サーバーレスはスループットベースの価格モデルを採用しているため、ご利用分のみのお支払いとなります。ご利用中のアプリケーションで、オンデマンドストリーミング容量の自動的なスケールアップおよびスケールダウンが必要な場合は、サーバーレスクラスターの使用を検討してください。

MSK サーバーレスは Apache Kafka と完全に互換性があるため、互換性のある任意のクライアントアプリケーションを使用してデータを生成および消費できます。また、以下のサービス と統合されます :

- AWS PrivateLink プライベート接続を提供するには
- AWS Identity and Access Management (IAM) Java および Java 以外の言語を使用する認証および認可に使用される IAM 用のクライアントの設定方法については、IAM アクセス制御用にクライアントを設定する を参照してください。
- AWS Glue スキーマ管理用のスキーマレジストリ
- Apache Flink ベースのストリーム処理対応の Amazon Managed Service for Apache Flink
- AWS Lambda イベント処理用

> **Note**
>
> MSK サーバーレスでは、すべてのクラスターに対して IAM アクセス制御が必要です。Apache Kafka アクセスコントロールリスト (ACL) はサポートされていません。詳細については、「the section called “IAM アクセスコントロール”」を参照してください。

MSK サーバーレスに適用される Service Quotas の詳細については、「the section called "サーバーレスクラスターのクォータ"」を参照してください。

サーバーレスクラスターの使用開始に役立つ情報や、構成とモニタリングオプションの詳細について知るには、以下を参照してください。

トピック

- MSK サーバーレスクラスターを使用する
- MSK サーバーレスクラスターの構成プロパティ
- MSK サーバーレスのクラスターをモニタリングする

MSK サーバーレスクラスターを使用する

このチュートリアルでは、MSK サーバーレスクラスターの作成方法と、作成したクラスターにアクセスできるクライアントマシンの作成、またクライアントを使用したクラスター上でのトピックの作成と、それらのトピックにデータを書き込む方法の一例をご説明します。この例は、サーバーレスクラスターを作成するにあたって選択可能なすべてのオプションを示しているわけではありません。この演習中のさまざまな場面で、わかりやすさのためにデフォルトのオプションを選択しています。このことは、サーバーレスクラスターのセットアップにあたって、他の方法がないという意味ではありません。 AWS CLI または Amazon MSK API を使用することもできます。詳細については、Amazon MSK API リファレンス 2.0を参照してください。

トピック

- MSK サーバーレスクラスターを作成する
- MSK サーバーレスクラスターのトピックの IAM ロールを作成する
- MSK サーバーレスクラスターにアクセスするためのクライアントマシンを作成する
- Apache Kafka トピックを作成する
- MSK サーバーレスでデータを生成および消費する
- MSK サーバーレス用に作成したリソースを削除する

MSK サーバーレスクラスターを作成する

このステップでは、2 つのタスクを実行します。まず、デフォルト設定で MSK サーバーレスクラスターを作成します。次に、作成したクラスター に関する情報を収集します。これは、後のステップでクラスターにデータを送信できるクライアントを作成するときに必要となる情報です。

サーバーレスクラスターを作成する

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home で Amazon MSK コンソールを開きます。
2. [クラスターを作成] を選択します。
3. [作成方法] では、[クイック作成] オプションを選択したままにします。[クイック作成] オプションを使用すると、デフォルト設定でサーバーレスクラスターを作成できます。
4. [クラスター名] に、わかりやすい名前 (**msk-serverless-tutorial-cluster** など) を入力します。
5. [全般的なクラスターのプロパティ] では、[クラスタータイプ] として [サーバーレス] を選択します。残りの [全般的なクラスターのプロパティ] にはデフォルト値を使用してください。
6. [すべてのクラスター設定] の下の表に注意してください。この表は、ネットワークや可用性などの重要な設定のデフォルト値のリストと、クラスターの作成後に各設定を変更できるかどうかを示しています。クラスターを作成する前に設定を変更するには、[作成方法] で [カスタム作成] オプションを選択する必要があります。

   > **Note**
   >
   > 最大 5 つの異なる VPC のクライアントを MSK サーバーレスクラスターと接続できます。機能停止時にクライアントアプリケーションが別のアベイラビリティーゾーンに切り替えられるようにするには、各 VPC に少なくとも 2 つのサブネットを指定する必要があります。

7. Create cluster (クラスターの作成) を選択します。

クラスター情報を取得する

1. [クラスターの概要] ページで、[クライアント情報の表示] を選択します。このボタンはAmazon MSK がクラスターの作成を完了するまで、グレーで表示されます。ボタンがアクティブになり使用可能になるまで、数分かかる場合があります。

2. Endpoint (エンドポイント) ラベルの下にある文字列をコピーします。これはブートストラップサーバー文字列です。
3. [プロパティ] タブを選択します。
4. [ネットワーク設定] セクションで、サブネットとセキュリティグループの ID をコピーし、保存します。この情報は後でクライアントマシンを作成する際に必要になります。
5. いずれかのサブネットを選択します。これにより、Amazon VPC コンソールが開きます。サブネットに関連付けられた Amazon VPC の ID を探します。後で使用するため、この Amazon VPC ID を保存します。

次のステップ

MSK サーバーレスクラスターのトピックの IAM ロールを作成する

MSK サーバーレスクラスターのトピックの IAM ロールを作成する

このステップでは、2 つのタスクを実行します。最初のタスクは、クラスターでトピックを作成し、それらのトピックにデータを送信するためのアクセスを許可する IAM ポリシーを作成することです。2 番目のタスクは、IAM ロールを作成し、作成したポリシーをそのロールに関連付けることです。後のステップでは、このロールを引き受けるクライアントマシンを作成し、それを使用してクラスター上にトピックを作成し、そのトピックにデータを送信します。

トピックを作成し、書き込むことを可能にする IAM ポリシーを作成する

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで [Policies] (ポリシー) を選択します。
3. [ポリシーを作成] を選択します。
4. JSON タブを選択し、エディタウィンドウの JSON を次の JSON に置き換えます。

   次の例では、以下を置き換えます。

   - *region* と、クラスターを作成した AWS リージョン のコード。
   - アカウント ID の例: *123456789012*。 AWS アカウント
   - サーバーレスクラスター ID とトピック名を持つ *msk-serverless-tutorial-cluster*/*c07c74ea-5146-4a03-add1-9baa787a5b14-s3* および *msk-serverless-tutorial-cluster*。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:Connect",
                "kafka-cluster:DescribeCluster"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:cluster/msk-serverless-tutorial-cluster/c07c74ea-5146-4a03-add1-9baa787a5b14-s3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:CreateTopic",
                "kafka-cluster:WriteData",
                "kafka-cluster:DescribeTopic"
            ],
            "Resource": [
            "arn:aws:kafka:us-east-1:123456789012:topic/msk-serverless-tutorial-cluster/*"
            ]
        }
    ]
}
```

セキュアポリシーを作成する方法については、「」を参照してくださいthe section called “IAM アクセスコントロール”。

5. [Next: Tags (次へ: タグ)] を選択します。
6. [次へ: レビュー] を選択します。
7. ポリシー名にわかりやすい名前 (**msk-serverless-tutorial-policy** など) を入力します。
8. [Create policy] (ポリシーの作成) を選択します。

IAM ロールを作成し、ポリシーを適用する

1. ナビゲーションペインで [Roles] (ロール) を選択します。
2. Create role (ロールの作成) を選択します。
3. [Common use cases] (一般的なユースケース) で [EC2] を選択し、[Next: Permissions] (次へ: アクセス許可) を選択します。
4. 検索ボックスに、このチュートリアル用に以前に作成したポリシーの名前を入力します。次に、ポリシーの左側にあるボックスをオンにします。
5. [Next: Tags (次へ: タグ)] を選択します。
6. [次へ: レビュー] を選択します。
7. ロール名に、わかりやすい名前 (**`msk-serverless-tutorial-role`** など) を入力します。
8. [ロールの作成] を選択してください。

次のステップ

MSK サーバーレスクラスターにアクセスするためのクライアントマシンを作成する

MSK サーバーレスクラスターにアクセスするためのクライアントマシンを作成する

このステップでは、2 つのタスクを実行します。最初のタスクでは、Apache Kafka クライアントマシンとして使用する Amazon EC2 インスタンスを作成します。2 番目のタスクでは、マシンに Java および Apache Kafka ツールをインストールします。

クライアントマシンを作成するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. [インスタンスを起動] を選択します。
3. クライアントマシンにわかりやすい [名前] (**`msk-serverless-tutorial-client`** など) を入力します。
4. [Amazon マシンイメージ (AMI) のタイプ] については、[Amazon Linux 2 AMI (HVM) - カーネル 5.10、SSD ボリュームタイプ] を選択したままにします。
5. t2.micro インスタンスタイプを選択したままにします。

6. [キーペア (ログイン)] で、[新しいキーペアの作成] を選択します。[キーペア名] に **MSKServerlessKeyPair** と入力します。[キーペアのダウンロード] を選択します。既存のキーペアを使用することもできます。
7. [ネットワーク設定] で、[編集] を選択してください。
8. [VPC] で、サーバーレスクラスターの仮想プライベートクラウド (VPC) の ID を入力します。これは、先ほどクラスターの作成後に保存した ID を持つ Amazon VPC サービスに基づく VPC です。
9. サブネットには、クラスターの作成後に ID を保存したサブネットを選択します。
10. [ファイアウォール (セキュリティグループ)] では、クラスターに関連付けられたセキュリティグループを選択します。この値は、そのセキュリティグループから同じセキュリティグループ内へのトラフィックを許可するインバウンドルールが存在する場合に機能します。このようなルールにより、同じセキュリティグループのメンバー同士が互いに通信できるようになります。詳細については、Amazon VPC デベロッパーガイドのセキュリティグループルールを参照してください。
11. [詳細情報] セクションを展開し、MSK サーバーレスクラスターのトピックの IAM ロールを作成する で作成した IAM ロールを選択します。
12. [Launch] (起動する) を選択します。
13. 左のナビゲーションペインの [インスタンス] を選択してください。新しく作成した Amazon EC2 インスタンスを表す行のチェックボックスをオンにします。これ以降は、このインスタンスのことをクライアントマシンと呼びます。
14. [Connect] (接続) を選択し、指示に従ってクライアントマシンに接続します。

クライアントマシンで Apache Kafka クライアントツールを設定するには

1. クライアントマシンに Java をインストールするには、次のコマンドを実行します。

```
sudo yum -y install java-11
```

2. トピックの作成とデータの送信に必要な Apache Kafka ツールを入手するには、次のコマンドを実行します。

```
wget https://archive.apache.org/dist/kafka/2.8.1/kafka_2.12-2.8.1.tgz
```

```
tar -xzf kafka_2.12-2.8.1.tgz
```

> **Note**
>
> Kafka アーカイブを抽出したら、binディレクトリ内のスクリプトに適切な実行アクセス許可があることを確認します。これを行うには、以下のコマンドを実行します。
>
> ```
> chmod +x kafka_2.12-2.8.1/bin/*.sh
> ```

3. kafka_2.12-2.8.1/libs ディレクトリに移動し、次のコマンドを実行して Amazon MSK IAM JAR ファイルをダウンロードします。Amazon MSK IAM JAR を使用すると、クライアントマシンがクラスターにアクセスできるようになります。

   ```
   wget https://github.com/aws/aws-msk-iam-auth/releases/download/v2.3.0/aws-msk-iam-auth-2.3.0-all.jar
   ```

   このコマンドを使用すると、Amazon MSK IAM JAR ファイルの他のバージョンまたは新しいバージョンをダウンロードすることもできます。

4. kafka_2.12-2.8.1/bin ディレクトリに移動します。次のプロパティ設定をコピーして、新しいファイルに貼り付けます。ファイルに client.properties という名前を付け、保存します。

   ```
   security.protocol=SASL_SSL
   sasl.mechanism=AWS_MSK_IAM
   sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
   sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
   ```

次のステップ

Apache Kafka トピックを作成する

Apache Kafka トピックを作成する

このステップでは、以前に作成したクライアントマシンを使用して、サーバーレスクラスターにトピックを作成します。

トピック

- トピックを作成するための環境のセットアップ

- トピックを作成し、そのトピックにデータを書き込む

トピックを作成するための環境のセットアップ

- トピックを作成する前に、MSK AWS IAM JAR ファイルを Kafka インストールの libs/ ディレクトリにダウンロードしていることを確認してください。これをまだ実行していない場合は、Kafka の libs/ ディレクトリで次のコマンドを実行します。

```
wget https://github.com/aws/aws-msk-iam-auth/releases/download/v2.3.0/aws-msk-iam-auth-2.3.0-all.jar
```

  この JAR ファイルは、MSK Serverless クラスターでの IAM 認証に必要です。
- Kafka コマンドを実行するときは、 に AWS MSK IAM JAR ファイルclasspathが含まれていることを確認する必要がある場合があります。これを行うには、次のいずれかを実行します。
  - 次の例に示すように、Kafka ライブラリを含めるようにCLASSPATH環境変数を設定します。

```
export CLASSPATH=<path-to-your-kafka-installation>/libs/*:<path-to-your-kafka-installation>/libs/aws-msk-iam-auth-2.3.0-all.jar
```

  - 次の例に示すようにclasspath、明示的な で完全な Java コマンドを使用して Kafka コマンドを実行します。

```
java -cp "<path-to-your-kafka-installation>/libs/*:<path-to-your-kafka-installation>/libs/aws-msk-iam-auth-2.3.0-all.jar"
 org.apache.kafka.tools.TopicCommand --bootstrap-server $BS --command-config
 client.properties --create --topic msk-serverless-tutorial --partitions 6
```

トピックを作成し、そのトピックにデータを書き込む

1. 次の export コマンドの *my-endpoint* を、クラスターの作成後に保存したブートストラップサーバー文字列に置き換えます。次に、クライアントマシン上の kafka_2.12-2.8.1/bin ディレクトリに移動し、export コマンドを実行します。

```
export BS=my-endpoint
```

2. 次のコマンドを実行して、msk-serverless-tutorial と呼ばれるトピックを作成します。

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --bootstrap-server $BS
 --command-config client.properties --create --topic msk-serverless-tutorial --
partitions 6
```

次のステップ

MSK サーバーレスでデータを生成および消費する

MSK サーバーレスでデータを生成および消費する

このステップでは、前のステップで作成したトピックを使用してデータを生成および使用します。

メッセージを生成および消費するには

1. 次のコマンドを実行して、コンソールプロデューサーを作成します。

   ```
   <path-to-your-kafka-installation>/bin/kafka-console-producer.sh --broker-list $BS
    --producer.config client.properties --topic msk-serverless-tutorial
   ```

2. 必要なメッセージを入力して、Enter キーを押します。このステップを 2、3 回繰り返します。行を入力して Enter キーを押すたびに、その行は個別のメッセージとしてクラスターに送信されます。
3. クライアントマシンへの接続を開いたままにして、そのマシンへ 2 番目の別の接続を新しいウィンドウで開きます。
4. クライアントマシンへの 2 回目の接続を使用して、以下のコマンドでコンソールコンシューマーを作成します。*my-endpoint* を、クラスターの作成後に保存したブートストラップサーバー文字列に置き換えます。

   ```
   <path-to-your-kafka-installation>/bin/kafka-console-consumer.sh --bootstrap-
   server my-endpoint --consumer.config client.properties --topic msk-serverless-
   tutorial --from-beginning
   ```

   コンソール プロデューサーコマンドを使用したときに、以前に入力したメッセージが表示され始めます。
5. プロデューサーウィンドウにさらにメッセージを入力し、コンシューマーウィンドウに表示されるようにします。

これらのコマンドの実行中にclasspath問題が発生した場合は、正しいディレクトリから実行していることを確認してください。また、MSK AWS IAM JAR が libs ディレクトリにあることを確認します。または、次の例に示すようにclasspath、明示的な で完全な Java コマンドを使用して Kafka コマンドを実行できます。

```
java -cp "kafka_2.12-2.8.1/libs/*:kafka_2.12-2.8.1/libs/aws-msk-iam-auth-2.3.0-
all.jar" org.apache.kafka.tools.ConsoleProducer –broker-list $BS –producer.config
 client.properties –topic msk-serverless-tutorial
```

次のステップ

MSK サーバーレス用に作成したリソースを削除する

MSK サーバーレス用に作成したリソースを削除する

このステップでは、このチュートリアルで作成したリソースを削除します。

クラスターの削除

1. https://console.aws.amazon.com/msk/home で Amazon MSK コンソールを開きます。
2. クラスターの一覧で、このチュートリアルで作成したクラスターを選択します。
3. [Actions] (アクション) で[Delete cluster] (クラスターの削除) を選択します。
4. フィールドに「delete」を入力し、[Delete] (削除) を選択します。

クライアントマシンの停止

1. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。
2. Amazon EC2 インスタンスのリストで、このチュートリアルで作成したクライアントマシンを選択します。
3. [Instance state] (インスタンスの状態) を選択し、[Terminate instance] (インスタンスの終了) をクリックします。
4. [Terminate] (終了) を選択します。

IAM ポリシーとロールを削除するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで [Roles] (ロール) を選択します。

3. 検索ボックスに、このチュートリアル用に作成した IAM ロールの名前を入力します。
4. ロールを選択します。[ロールの削除] を選択し、削除を確定します。
5. ナビゲーションペインで [Policies] (ポリシー) を選択します。
6. 検索ボックスに、このチュートリアル用に作成したポリシーの名前を入力します。
7. ポリシーを選択すると、その概要ページが開きます。ポリシーの[Summary] (概要) ページで [Delete policy] (ポリシーの削除) を選択します。
8. [削除] を選択します。

MSK サーバーレスクラスターの構成プロパティ

Amazon MSK はサーバーレスクラスターのブローカー設定プロパティを設定します。これらのブローカー構成プロパティの設定は変更できません。ただし、次のトピックレベルの構成プロパティは設定または変更できます。その他のトピックレベルの構成プロパティはすべて設定できません。

設定プロパティ	デフォルト	Editable	最大許容値
cleanup.policy	削除	はい。ただし、トピック作成時に限ります	
compression.type	プロデューサー	あり	
max.message.bytes	1048588	あり	8388608 (8MiB)
message.timestamp.difference.max.ms	long.max	あり	
message.timestamp.type	CreateTime	あり	
retention.bytes	250 GiB	あり	無制限。無制限の保持のため -1 に設定します
retention.ms	7 日間	あり	無制限。無制限の保持のため -1 に設定します

これらのトピックレベルの構成プロパティを設定または変更するには、Apache Kafka コマンドラインツールを使用できます。設定方法の詳細と例については、公式のApache Kafka ドキュメントで「3.2 Topic-level Configs」を参照してください。

> **Note**
>
> MSK Serverless のトピックの segment.bytes 設定を変更することはできません。ただし、Kafka Streams アプリケーションは segment.bytes 設定値で内部トピックを作成しようとする場合があります。これは、MSK Serverless で許可されるものとは異なります。MSK Serverless で Kafka Streams を設定する方法については、「」を参照してくださいMSK Express ブローカーと MSK Serverless での Kafka Streams の使用。

Amazon MSK サーバーレス で Apache Kafka コマンドラインツールを使用する場合は、「Amazon MSK サーバーレス入門ドキュメント」、「To set up Apache Kafka client tools on the client machine」のステップ 1~4 を完了していることを確認してください。さらに、コマンドに `--command-config client.properties` パラメータを含める必要があります。

例えば、次のコマンドを使用して retention.bytes トピック設定のプロパティを変更し、無制限の保持を設定できます。

```
<path-to-your-kafka-client-installation>/bin/kafka-configs.sh –bootstrap-
server <bootstrap_server_string> –command-config client.properties --entity-type topics
 --entity-name <topic_name> --alter --add-config retention.bytes=-1
```

この例では、*<bootstrap server string>* を Amazon MSK サーバーレスクラスターのブートストラップサーバーエンドポイントに置き換え、*<topic_name>* を変更するトピックの名前に置き換えます。

`--command-config client.properties` パラメータでは、Kafka コマンドラインツールが適切な設定を使用して Amazon MSK サーバーレスクラスターと通信することを確認します。

MSK サーバーレスのクラスターをモニタリングする

Amazon MSK が Amazon CloudWatch と統合することで、MSK サーバーレスクラスターのメトリクスを収集、表示、および分析できるようになります。次の表に示すメトリクスは、すべてのサーバーレスクラスターで使用できます。これらのメトリクスは、トピックの各パーティションの個別のデータポイントとして公開されるため、トピックレベルの見通しを得るための「SUM」統計としてご覧になることをお勧めします。

Amazon MSK は、1 分に 1 回の頻度で PerSec メトリクスを CloudWatch に公開します。つまり、1 分間の「SUM」統計は、PerSec メトリクスの 1 秒あたりのデータを正確に表しているということです。1 分を超える時間の 1 秒あたりのデータを収集するには、次の CloudWatch 数学式を使用します: `m1 * 60/PERIOD(m1)`。

デフォルトのモニタリングレベルで使用可能なメトリクス

名前	表示可能なタイミング	ディメンション	説明
BytesInPerSec	プロデューサーがトピックに書き込んだ後	クラスター名、トピック	クライアントから受信した 1 秒あたりのバイト数。このメトリクスはトピックで使用できます。
BytesOutPerSec	コンシューマーグループがトピックから消費した後	クラスター名、トピック	クライアントに送信された 1 秒あたりのバイト数。このメトリクスはトピックで使用できます。
FetchMessageConversionsPerSec	コンシューマーグループがトピックから消費した後	クラスター名、トピック	トピックの 1 秒あたりのフェッチメッセージ変換回数。
EstimatedMaxTimeLag	コンシューマーグループがトピックから消費した後	クラスター名、コンシューマーグループ、トピック	MaxOffsetLag メトリクスの推定時間。
MaxOffsetLag	コンシューマーグループがトピックから消費した後	クラスター名、コンシューマーグループ、トピック	トピック内のすべてのパーティションにおける最大オフセットラグ。
MessagesInPerSec	プロデューサーがトピックに書き込んだ後	クラスター名、トピック	トピックの 1 秒あたりの受信メッセージ数。

名前	表示可能なタイミング	ディメンション	説明
ProduceMessageConversionsPerSec	プロデューサーがトピックに書き込んだ後	クラスター名、トピック	トピックの 1 秒あたりの生成メッセージ変換回数。
SumOffsetLag	コンシューマーグループがトピックから消費した後	クラスター名、コンシューマーグループ、トピック	トピック内のすべてのパーティションの集計オフセットラグ。

MSK サーバーレスメトリクスの表示

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Metrics] (メトリクス) から [All metrics] (すべてのメトリクス) を選択します。
3. メトリクスで「**kafka**」を検索します。
4. <shared id="AWS"/>/Kafka / Cluster Name, Topic] (<shared id="AWS"/>/Kafka / クラスター名、トピック) または [<shared id="AWS"/>/Kafka / Cluster Name, Consumer Group, Topic] (<shared id="AWS"/>/Kafka / クラスター名、コンシューマグループ、トピック) を選択すると、各種のメトリクスが表示されます。

MSK Connect を理解する

MSK Connect は Amazon MSK の機能であり、デベロッパーが Apache Kafka クラスターとの間でデータを簡単にストリーミングできるようにします。MSK Connect は Kafka Connect バージョン 2.7.1 または 3.7.x を使用します。これは、Apache Kafka クラスターをデータベース、検索インデックス、ファイルシステムなどの外部システムに接続するためのオープンソースフレームワークです。MSK Connect を使用すると、Kafka Connect 用に構築されたフルマネージドコネクタをデプロイして、Amazon S3 や Amazon OpenSearch Service などの一般的なデータストアにデータを移動したりデータストアからデータをプルしたりできます。データベースから Apache Kafka クラスターに変更ログをストリーミングするために Debezium などのサードパーティによって開発されたコネクタをデプロイするか、コードを変更せずに既存のコネクタをデプロイできます。コネクタは、ロードの変化に合わせてオートスケーリングされ、使用したリソースに対してのみ料金が発生します。

ソースコネクタを使用して、外部システムからトピックにデータをインポートします。シンクコネクタを使用すると、トピックから外部システムにデータをエクスポートできます。

MSK Connect は、MSK クラスターであろうと独立してホストされている Apache Kafka クラスターであろうと、Amazon VPC に接続できる Apache Kafka クラスターのコネクタをサポートします。

MSK Connect は、コネクタのヘルスと配信状態を継続的に監視し、基盤となるハードウェアにパッチを適用して管理し、スループットの変化に合わせてコネクタをオートスケーリングします。

MSK Connect の使用を開始するには、「the section called “入門”」を参照してください。

MSK Connect で作成できる AWS リソースについては、「」、the section called “カスタムプラグインの作成”「」、およびthe section called “コネクタを理解する”「」を参照してくださいthe section called “MSK Connect ワーカーを理解する”。

MSK Connect API の詳細については、「Amazon MSK Connect API リファレンス」を参照してください。

Amazon MSK Connect を使用する利点

Apache Kafka は、リアルタイムデータストリームの取り込みと処理に最も広く採用されているオープンソースストリーミングプラットフォームの 1 つです。Apache Kafka を使用すると、データ生成アプリケーションとデータ消費アプリケーションを分離して個別にスケーリングできます。

Kafka Connect は、Apache Kafka を使用してストリーミングアプリケーションを構築および実行する上で重要なコンポーネントです。Kafka Connect は、Kafka と外部システム間でデータを移動する

標準化された方法を提供します。Kafka Connect はスケーラビリティが高く、大量のデータを処理できます。Kafka Connect は、Kafka トピックと外部システム間でデータを移動するコネクタを設定、デプロイ、およびモニタリングするための強力な API オペレーションとツールのセットを提供します。これらのツールを使用して、ストリーミングアプリケーションの特定のニーズを満たすように Kafka Connect の機能をカスタマイズおよび拡張できます。

Apache Kafka Connect クラスターを単独で運用している場合や、オープンソースの Apache Kafka Connect アプリケーションを AWSに移行しようとしている場合に、課題に遭遇する可能性があります。これらの課題には、インフラストラクチャのセットアップとアプリケーションのデプロイに必要な時間、セルフマネージド型の Apache Kafka Connect クラスターを設定する際のエンジニアリング上の障害、管理運用上のオーバーヘッドが含まれます。

これらの課題に対処するために、Amazon Managed Streaming for Apache Kafka Connect (Amazon MSK Connect) を使用して、オープンソースの Apache Kafka Connect アプリケーションを AWS に移行することをお勧めします。Amazon MSK Connect は、Kafka Connect を使用することで、Apache Kafka クラスターと外部システム (データベース、検索インデックス、ファイルシステムなど) との間のデータのストリーミングを簡素化します。

Amazon MSK Connect に移行する利点をいくつかご紹介します。

- 運用上のオーバーヘッドの排除 – Amazon MSK Connect は、Apache Kafka Connect クラスターのパッチ適用、プロビジョニング、およびスケーリングに関連する運用上の負担を軽減します。Amazon MSK Connect は、Connect クラスターの状態を継続的にモニタリングし、ワークロードの中断を引き起こすことなく、パッチ適用とバージョンアップグレードを自動化します。
- Connect タスクの自動再起動 – Amazon MSK Connect は、失敗したタスクを自動的に復旧するため、本番環境の中断を減らすことができます。タスクの失敗は、Kafka の TCP 接続制限に違反するなどの一時的なエラーや、新しいワーカーがシンクコネクタのコンシューマーグループに参加したときのタスクのリバランス実行によって発生する可能性があります。
- 自動水平スケーリングおよび垂直スケーリング – Amazon MSK Connect により、コネクタアプリケーションが自動的にスケーリングして、より高いスループットをサポートできるようになります。Amazon MSK Connect はユーザーに代わってスケーリングを管理します。ユーザーは、自動スケーリンググループのワーカー数と使用率のしきい値のみを指定する必要があります。Amazon MSK Connect `UpdateConnector` API オペレーションを使用して、vCPU を 1~8 個の範囲で垂直方向にスケールアップまたはスケールダウンすることで、可変スループットのサポートが可能になります。
- プライベートネットワーク接続 — Amazon MSK Connect は、 AWS PrivateLink とプライベート DNS 名を使用してソースシステムとシンクシステムにプライベートに接続します。

MSK Connect の使用開始

これは、 AWS Management Console を使用して MSK クラスターを作成し、クラスターから S3 バケットにデータを送信するシンクコネクタを作成するstep-by-stepチュートリアルです。

トピック

- MSK Connect に必要なリソースを設定する
- カスタムプラグインを作成する
- クライアントマシンと Apache Kafka トピックを作成する
- コネクタの作成
- MSK クラスターにデータを送信する

MSK Connect に必要なリソースを設定する

このステップでは、この入門シナリオに必要な次のリソースを作成します。

- コネクタからデータを受信する送信先として機能する Amazon S3 バケット。
- データの送信先となる MSK クラスター。次に、コネクタはこのクラスターからデータを読み取り、宛先の S3 バケットに送信します。
- 送信先 S3 バケットに書き込むアクセス許可を含む IAM ポリシー。
- コネクタが宛先 S3 バケットに書き込むことを可能にする IAM ロール。作成した IAM ポリシーをこのロールに追加します。
- クラスターとコネクタを備えた Amazon VPC から Amazon S3 にデータを送信できるようにする Amazon VPC エンドポイント。

S3 バケットを作成するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/s3/ で Amazon S3 コンソールを開きます。
2. [バケットを作成] を選択します。
3. バケットの名前には、`amzn-s3-demo-bucket-mkc-tutorial` などのわかりやすい名前を入力します。
4. 下にスクロールして、Create bucket (バケットの作成) を選択します。

5. バケットのリストで、新しく作成されたバケットを選択します。
6. Create folder (フォルダの作成) を選択します。
7. フォルダの名前として `tutorial` と入力し、下にスクロールして Create folder (フォルダの作成) を選択します。

クラスターを作成するには

1. https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. 左側のペインの MSK Clusters (MSK クラスター) で、Clusters (クラスター) を選択します。
3. Create cluster (クラスターの作成) を選択します。
4. 作成 メソッドで、カスタム作成 を選択します。
5. クラスター名には **`mkc-tutorial-cluster`** と入力します。
6. クラスタータイプで、プロビジョニング済み を選択します。
7. [次へ] を選択します。
8. [ネットワーク] で Amazon VPC を選択します。次に、使用するアベイラビリティーゾーンとサブネットを選択します。このチュートリアルの後半で必要になるため、選択した Amazon VPC とサブネットの ID を覚えておいてください。
9. [次へ] を選択します。
10. Access control methods (アクセス制御方法) で、Unauthenticated access (認証されていないアクセス) のみが選択されていることを確認します。
11. Encryption (暗号化) で、Plaintext (プレーンテキスト) のみが選択されていることを確認します。
12. ウィザードを続行し、[クラスターの作成] を選択します。そのクラスターの詳細ページが表示されます。そのページで、[適用されたセキュリティグループ] の下のセキュリティグループ ID を探します。このチュートリアルの後半で必要になるため、その ID を覚えておいてください。

S3 バケットに書き込むアクセス許可を持つ IAM ポリシーを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで [Policies] (ポリシー) を選択します。
3. [Create policy] (ポリシーの作成) を選択します。
4. ポリシーエディタで JSON を選択し、エディタウィンドウの JSON を次の JSON に置き換えます。

次の例では、*#amzn-s3-demo-bucket-my-tutorial>* を S3 バケットの名前に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListBucket",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::<amzn-s3-demo-bucket-my-tutorial>"
    },
    {
      "Sid": "AllowObjectActions",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:AbortMultipartUpload",
        "s3:ListMultipartUploadParts",
        "s3:ListBucketMultipartUploads"
      ],
      "Resource": "arn:aws:s3:::<amzn-s3-demo-bucket-my-tutorial>/*"
    }
  ]
}
```

セキュアポリシーを作成する方法については、「」を参照してくださいthe section called “IAM アクセスコントロール”。

5. [次へ] を選択します。
6. [レビューと作成] ページで、以下の操作を実行します。

   a. ポリシー名には、 などのわかりやすい名前を入力します**mkc-tutorial-policy**。

b. このポリシーで定義されているアクセス許可で、ポリシーで定義されているアクセス許可を確認および/または編集します。

c. (オプション) ポリシーを識別、整理、または検索しやすくするには、新しいタグを追加 を選択してキーと値のペアとしてタグを追加します。たとえば、 **Environment**と のキーと値のペアを使用して、ポリシーにタグを追加します**Test**。

タグの使用の詳細については、IAM ユーザーガイドの「 AWS Identity and Access Management リソースのタグ」を参照してください。

7. [Create policy] (ポリシーの作成) を選択します。

送信先バケットに書き込みことができる IAM ロールを作成するには

1. IAM コンソールのナビゲーションペインで、ロールを選択し、ロールの作成を選択します。
2. [信頼されたエンティティを選択] ページで、以下の操作を実行してください :

a. 信頼できるエンティティタイプ で、AWS のサービス を選択します。

b. サービスまたはユースケースで、S3 を選択します。

c. ユースケースで、S3 を選択します。

3. [次へ] を選択します。
4. [アクセス許可を追加] ページで、以下を実行します。

a. アクセス許可ポリシーの検索ボックスに、このチュートリアル用に以前に作成したポリシーの名前を入力します。例えば、mkc-tutorial-policy などです。次に、ポリシー名の左側にあるボックスを選択します。

b. (オプション) アクセス許可の境界を設定します。このアドバンスド機能は、サービスロールで使用できますが、サービスにリンクされたロールではありません。アクセス許可の境界の設定については、IAM ユーザーガイドの「ロールの作成とポリシーのアタッチ (コンソール) 」を参照してください。

5. [次へ] を選択します。
6. [名前を付けて、レビューし、作成する] ページで、以下の操作を実行します。

a. ロール名には、 などのわかりやすい名前を入力します**mkc-tutorial-role**。

> **Important**
>
> ロールに名前を付けるときは、次のことに注意してください。

- ロール名は 内で一意である必要があり AWS アカウント、大文字と小文字を区別することはできません。

  例えば、**PRODROLE** と **prodrole** の両方の名前でロールを作成することはできません。ロール名がポリシーまたは ARN の一部として使用される場合、ロール名は大文字と小文字が区別されます。ただし、サインインプロセスなど、コンソールにロール名がユーザーに表示される場合、ロール名は大文字と小文字が区別されません。
- 他のエンティティがロールを参照する可能性があるため、ロールを作成した後にロール名を編集することはできません。

b. (オプション) [説明] にロールの説明を入力します。

c. (オプション) ロールのユースケースとアクセス許可を編集するには、「ステップ 1: 信頼できるエンティティを選択する」または「ステップ 2: アクセス許可を追加する」セクションで、「編集」を選択します。

d. (オプション) ロールを識別、整理、または検索しやすくするには、新しいタグを追加 を選択して、キーと値のペアとしてタグを追加します。たとえば、 **ProductManager**と のキーと値のペアを使用して、ロールにタグを追加します**John**。

   タグの使用の詳細については、IAM ユーザーガイドの「 AWS Identity and Access Management リソースのタグ」を参照してください。

7. ロールを確認したら、[ロールを作成] を選択します。

MSK Connect がそのロールを引き受けることができるようにするには

1. IAM コンソールの左側のペインのAccess management (アクセス管理) で、Roles (ロール) を選択します。
2. `mkc-tutorial-role` を見つけて選択します。
3. ロールの Summary (概要) で、Trust relationships (信頼関係) タブを選択します。
4. Edit trust relationship (信頼関係の編集) を選択します。
5. 既存の信頼ポリシーを次の JSON に置き換えます。

   JSON

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kafkaconnect.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

6. 信頼ポリシーの更新 を選択します。

クラスターの VPC から Amazon S3 への Amazon VPC エンドポイントを作成するには

1. Amazon VPC コンソールhttps://console.aws.amazon.com/vpc/を開きます。
2. 左側のペインで、Endpoints (エンドポイント) を選択します。
3. Create endpoint (エンドポイントの作成) を選択します。
4. Service Name (サービス名) で、com.amazonaws.us-east-1.s3 サービスとGateway (ゲートウェイ) タイプを選択します。
5. クラスターの VPC を選択してから、クラスターのサブネットに関連付けられているルートテーブルの左側にあるボックスを選択します。
6. Create endpoint (エンドポイントの作成) を選択します。

次のステップ

カスタムプラグインを作成する

カスタムプラグインを作成する

プラグインには、コネクタのロジックを定義するコードが含まれています。このステップでは、Lenses Amazon S3 Sink Connector のコードを含むカスタムプラグインを作成します。後のステップで、MSK コネクタを作成するときに、そのコードがこのカスタム プラグインにあることを指定します。同じプラグインを使用して、構成が異なる複数の MSK コネクタを作成できます。

カスタムプラグインを作成するには

1. S3 コネクタをダウンロードします。

2. アクセスできる S3 バケットに ZIP ファイルをアップロードします。Amazon S3 にファイルをアップロードする方法については、Amazon S3 ユーザーガイドのオブジェクトのアップロードを参照してください。
3. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
4. 左側のペインで MSK Connect を展開し、Custom plugins (カスタムプラグイン) を選択します。
5. Create custom plugin (カスタムプラグインの作成) を選択します。
6. Browse S3 (S3 の参照) を選択します。
7. バケットのリストで、ZIP ファイルをアップロードしたバケットを見つけて、そのバケットを選択します。
8. バケット内のオブジェクトのリストで、ZIP ファイルの左側にあるラジオボタンを選択し、選択というラベル付けたボタンを選択します。
9. カスタムプラグイン名に `mkc-tutorial-plugin` と入力し、Create custom plugin (カスタムプラグインの作成) を選択します。

カスタムプラグインの作成が完了するまで AWS に数分かかる場合があります。作成プロセスが完了すると、ブラウザウィンドウの上部にあるバナーに次のメッセージが表示されます。

```
Custom plugin mkc-tutorial-plugin was successfully created
The custom plugin was created. You can now create a connector using this custom plugin.
```

次のステップ

クライアントマシンと Apache Kafka トピックを作成する

クライアントマシンと Apache Kafka トピックを作成する

このステップでは、Apache Kafka クライアントインスタンスとして使用する Amazon EC2 インスタンスを作成します。次に、このインスタンスを使用して、クラスター上にトピックを作成します。

クライアントマシンを作成するには

1. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。
2. [Launch Instances] (インスタンスの起動) を選択してください。
3. クライアントマシンの [名前] (**`mkc-tutorial-client`** など) を入力します。
4. [Amazon マシンイメージ (AMI) のタイプ] については、[Amazon Linux 2 AMI (HVM) - カーネル 5.10、SSD ボリューム タイプ] を選択したままにします。

5. [t2.xlarge] インスタンスタイプを選択します。
6. [キーペア (ログイン)] で、[新しいキーペアの作成] を選択します。[キーペア名] に **mkc-tutorial-key-pair** を入力し、[キーペアのダウンロード] を選択します。既存のキーペアを使用することもできます。
7. [インスタンスを起動] を選択します。
8. [インスタンスの表示] を選択します。次に、[セキュリティグループ] 列で、新しいインスタンスに関連付けられているセキュリティグループを選択します。セキュリティグループの ID をコピーし、後で使用できるように保存します。

新しく作成されたクライアントがクラスターにデータを送信するのを許可するには

1. Amazon VPC コンソールhttps://console.aws.amazon.com/vpc/を開きます。
2. 左側のペインのSECURITY (セキュリティ) で、Security Groups (セキュリティグループ) を選択します。Security group ID (セキュリティグループID) 列で、クラスターのセキュリティグループを見つけます。the section called “MSK Connect に必要なリソースを設定する” でクラスターを作成したときに、このセキュリティグループのIDを保存しました。行の左側にあるボックスを選択して、このセキュリティグループを選択します。他のセキュリティグループが同時に選択されていないことを確認してください。
3. 画面の下半分で、Inbound rules (インバウンドルール) タブを選択します。
4. Edit inbound rules (インバウンドルールの編集) を選択します。
5. 画面の左下で、Add rule (ルールの追加) を選択します。
6. 新しいルールで、Type (タイプ) 列の All traffic (すべてのトラフィック) を選択します。[ソース] 列の右側のフィールドに、クライアントマシンのセキュリティグループの ID を入力します。これは、クライアントマシンを作成した後に保存したセキュリティグループ ID です。
7. Save rules (ルールの保存) を選択します。これで、MSK クラスターは、前の手順で作成したクライアントからのすべてのトラフィックを受け入れます。

トピックを作成する

1. Amazon EC2 コンソールhttps://console.aws.amazon.com/ec2/を開きます。
2. インスタンスのテーブルで `mkc-tutorial-client` を選択します。
3. 画面上部のConnect (接続) を選択し、指示に従ってインスタンスに接続します。
4. 次のコマンドを実行して、クライアントインスタンスに Java をインストールします。

```
sudo yum install java-1.8.0
```

5. 次のコマンドを実行して、Apache Kafka をダウンロードします。

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

> **Note**
>
> このコマンドで使用されているもの以外のミラーサイトを使用する場合は、Apache ウェブサイトで別のサイトを選択できます。

6. 前のステップで TAR ファイルをダウンロードしたディレクトリで次のコマンドを実行します。

```
tar -xzf kafka_2.12-2.2.1.tgz
```

7. kafka_2.12-2.2.1 ディレクトリに移動します。
8. https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
9. 左側のペインで Clusters (クラスター) を選択してから、`mkc-tutorial-cluster` という名前を選択します。
10. View client information (ライアント情報の表示) を選択します。
11. プレーンテキストの接続文字列をコピーします。
12. [完了] をクリックします。
13. クライアントインスタンス (`mkc-tutorial-client`) で次のコマンドを実行し、*`bootstrapServerString`* を、クラスターのクライアント情報を表示したときに保存した値に置き換えます。

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --create --bootstrap-
server bootstrapServerString --replication-factor 2 --partitions 1 --topic mkc-
tutorial-topic
```

コマンドが成功すると、次のメッセージが表示されます : `Created topic mkc-tutorial-topic.`

次のステップ

コネクタの作成

コネクタの作成

この手順では、 AWS Management Consoleを使用してコネクタを作成する方法について説明します。

コネクタを作成するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. 左側のペインで、MSK Connect を展開し、Connectors (コネクタ) を選択します。
3. Create connector (コネクタの作成) を選択します。
4. プラグインのリストで、`mkc-tutorial-plugin` を選択し、次にNext (次へ) を選択します。
5. コネクタ名に `mkc-tutorial-connector` と入力します。
6. クラスターのリストから、`mkc-tutorial-cluster` を選択します。
7. 次の構成をコピーして、コネクタ構成フィールドに貼り付けます。

   region は、コネクタ AWS リージョン を作成する のコードに置き換えてください。また、Amazon S3 バケット名 *#amzn-s3-demo-bucket-my-tutorial>* を次の例のバケット名に置き換えます。

```
connector.class=io.confluent.connect.s3.S3SinkConnector
s3.region=us-east-1
format.class=io.confluent.connect.s3.format.json.JsonFormat
flush.size=1
schema.compatibility=NONE
tasks.max=2
topics=mkc-tutorial-topic
partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartitioner
storage.class=io.confluent.connect.s3.storage.S3Storage
s3.bucket.name=<amzn-s3-demo-bucket-my-tutorial>
topics.dir=tutorial
```

8. Access permissions (アクセス許可) で `mkc-tutorial-role` を選択します。
9. Next (次へ) を選択します。Security (セキュリティ) ページで、もう一度 Next (次へ) を選択します。
10. Logs (ログ) ページで、Next (次へ) を選択します。

11. Review and create (確認して作成) で、Create connector (コネクタの作成) を選択します。

次のステップ

MSK クラスターにデータを送信する

MSK クラスターにデータを送信する

このステップでは、前に作成した Apache Kafka トピックにデータを送信し、宛先 S3 バケットで同じデータを探します。

MSK クラスターにデータを送信するには

1. クライアントインスタンスの Apache Kafka インストールの bin フォルダに、次の内容の client.properties という名前のテキストファイルを作成します。

```
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
```

2. 次のコマンドを実行して、コンソールプロデューサーを作成します。*BootstrapBrokerString* を前のコマンドを実行したときに取得した値に置き換えます。

```
<path-to-your-kafka-installation>/bin/kafka-console-producer.sh --broker-
list BootstrapBrokerString --producer.config client.properties --topic mkc-
tutorial-topic
```

3. 必要なメッセージを入力して、Enter キーを押します。このステップを 2、3 回繰り返します。行を入力して Enter キーを押すたびに、その行は個別のメッセージとして Apache Kafka クラスターに送信されます。
4. 宛先の Amazon S3 バケットを調べて、前のステップで送信したメッセージを見つけます。

コネクタを理解する

コネクタは、ストリーミングデータをデータソースから Apache Kafka クラスターに継続的にコピーするか、クラスターからデータシンクにデータを継続的にコピーすることにより、外部システムと Amazon サービスを Apache Kafka と統合します。コネクタは、データを宛先に配信する前に、変換、フォーマット変換、データのフィルタリングなどの軽量ロジックを実行することもできます。ソースコネクタはデータソースからデータをプルしてこのデータをクラスターにプッシュし、シンクコネクタはクラスターからデータをプルしてこのデータをデータシンクにプッシュします。

次の図表は、コネクタのアーキテクチャを示しています。ワーカーは、コネクタロジックを実行する Java 仮想マシン (JVM) プロセスです。各ワーカーは、並列スレッドで実行され、データをコピーする作業を行う一連のタスクを作成します。タスクは状態を保存しないため、復元力のあるスケーラブルな Data Pipeline を提供するために、いつでも開始、停止、または再開できます。

コネクタ容量を理解する

コネクタの総容量は、コネクタが持つワーカーの数、およびワーカーごとの MSK 接続ユニット (MCU) の数によって異なります。各 MCU は、1 vCPU のコンピューティングと 4 GiB のメモリを表します。MCU メモリは、使用中のヒープメモリではなく、ワーカーインスタンスの合計メモリに関係します。

MSK Connect ワーカーは、カスタマー提供のサブネット内の IP アドレスを消費します。各ワーカーは、カスタマー提供のサブネットの 1 つに含まれている 1 つの IP アドレスを使用します。CreateConnector リクエストに応答して提供されたサブネットに使用可能な IP アドレスが十分にあり、指定した容量を満たしていることを確認する必要があります。特に、ワーカー数が変動する可能性のあるコネクタを自動スケーリングする場合は注意が必要です。

コネクタを作成するには、次の 2 つの容量モードのいずれかを選択する必要があります。

- プロビジョニング済み: コネクタの容量要件がわかっている場合は、このモードを選択します。次の 2 つの値を指定します。
  - ワーカー数
  - ワーカーあたりの MCU の数。
- オートスケーリング: コネクタの容量要件が可変である場合、またはアドバンスにそれらを知らない場合は、このモードを選択します。自動スケーリングモードを使用すると、Amazon MSK Connect はコネクタの `tasks.max` プロパティを、コネクタで実行されているワーカーの数とワーカーあたりの MCU の数に比例した値で上書きします。

  次の 3 つの値のセットを指定します。
  - ワーカーの最小数と最大数。
  - CPU 使用率のスケールインおよびスケールアウトのパーセンテージ。これは、`CpuUtilization` メトリクスによって決定されます。コネクタの `CpuUtilization` メトリクスがスケールアウトのパーセンテージを超えると、MSK Connect はコネクタで実行されているワーカーの数を増やします。`CpuUtilization` メトリクスがスケールインのパーセンテージを下回ると、MSK Connect はワーカーの数を減らします。ワーカーの数は、コネクタの作成時に指定した最小数と最大数の範囲内に常に留まります。

- ワーカーあたりの MCU の数。

ワーカーの詳細については、「the section called “MSK Connect ワーカーを理解する”」を参照してください。MSK Connect メトリクスについては、「the section called “MSK Connect のモニタリング”」を参照してください。

コネクタを作成する

この手順では、 AWS Management Consoleを使用してコネクタを作成する方法について説明します。

を使用したコネクタの作成 AWS Management Console

1. https://console.aws.amazon.com/msk/ でAmazon MSK コンソールを開きます。
2. 左側のペインの MSK Connect で、Connectors (コネクタ) を選択します。
3. Create connector (コネクタの作成) を選択します。
4. 既存のカスタムプラグインを使用してコネクタを作成するか、最初に新しいカスタムプラグインを作成するかを選択できます。カスタムプラグインとその作成方法については、「the section called “カスタムプラグインの作成”」を参照してください。この手順では、使用するカスタムプラグインがあると仮定します。カスタムプラグインのリストで、使用するプラグインを見つけ、左側のボックスを選択して、Next (次へ) を選択します。
5. 名前と、オプションで説明を入力します。
6. 接続するクラスターを選択します。
7. コネクタ構成を指定します。指定する必要のある構成パラメータは、作成するコネクタのタイプによって異なります。ただし、`connector.class` パラメータや`tasks.max` パラメータなど、一部のパラメータはすべてのコネクタに共通です。以下は、Confluent Amazon S3 Sink Connectorの設定例です。

```
connector.class=io.confluent.connect.s3.S3SinkConnector
tasks.max=2
topics=my-example-topic
s3.region=us-east-1
s3.bucket.name=amzn-s3-demo-bucket
flush.size=1
storage.class=io.confluent.connect.s3.storage.S3Storage
format.class=io.confluent.connect.s3.format.json.JsonFormat
partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartitioner
key.converter=org.apache.kafka.connect.storage.StringConverter
```

```
value.converter=org.apache.kafka.connect.storage.StringConverter
schema.compatibility=NONE
```

8. 次に、コネクタ容量を設定します。プロビジョニングモードとオートスケーリングの 2 つの容量モードから選択できます。これら 2 つのオプションの詳細については、「the section called "コネクタ容量を理解する"」を参照してください。
9. デフォルトのワーカー構成またはカスタムワーカー構成のいずれかを選択します。カスタムワーカー構成の作成については、「the section called "MSK Connect ワーカーを理解する"」を参照してください。
10. 次に、サービス実行ロールを指定します。これは、MSK Connect が引き受けることができ、必要な AWS リソースにアクセスするために必要なすべてのアクセス許可をコネクタに付与する IAM ロールである必要があります。これらの権限は、コネクタのロジックによって異なります。このロールを作成する方法については、「the section called "サービス実行ロールを理解する"」を参照してください。
11. Next (次へ) を選択し、セキュリティ情報を確認してから、もう一度 Next (次へ) を選択します。
12. 必要なログオプションを指定し、Next (次へ) を選択します。ログ作成の詳細については、「the section called "ロギング"」を参照してください。
13. Create connector (コネクタの作成) を選択します。

MSK Connect API を使用してコネクタを作成するには、CreateConnectorを参照してください。

UpdateConnector API を使用してコネクタの設定を変更できます。詳細については、「the section called "コネクタを更新する"」を参照してください。

コネクタを更新する

この手順では、 を使用して既存の MSK Connect コネクタの設定を更新する方法について説明しますAWS Management Console。

を使用したコネクタ設定の更新 AWS Management Console

1. https://console.aws.amazon.com/msk/ でAmazon MSK コンソールを開きます。
2. 左側のペインの MSK Connect で、Connectors (コネクタ) を選択します。
3. 既存のコネクタを選択します。
4. コネクタ設定の編集を選択します。
5. コネクタ設定を更新します。UpdateConnector `connector.class`を使用して上書きすることはできません。次の例は、Confluent Amazon S3 シンクコネクタの設定例を示しています。

```
connector.class=io.confluent.connect.s3.S3SinkConnector
tasks.max=2
topics=my-example-topic
s3.region=us-east-1
s3.bucket.name=amzn-s3-demo-bucket
flush.size=1
storage.class=io.confluent.connect.s3.storage.S3Storage
format.class=io.confluent.connect.s3.format.json.JsonFormat
partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartitioner
key.converter=org.apache.kafka.connect.storage.StringConverter
value.converter=org.apache.kafka.connect.storage.StringConverter
schema.compatibility=NONE
```

6. [送信] を選択します。
7. その後、コネクタの Operations タブでオペレーションの現在の状態をモニタリングできます。

MSK Connect API を使用してコネクタの設定を更新するには、UpdateConnector」を参照してください。

コネクタからの接続

以下のベストプラクティスにより、Amazon MSK Connect への接続パフォーマンスを向上させることができます。

Amazon VPC ピアリングまたは Transit Gateway の IP が重複しないようにする

Amazon VPC ピアリングまたは Transit Gateway を Amazon MSK Connect と共に使用している場合は、ピアリングされた VPC リソースに到達するのに以下の CIDR 範囲内の IP を使用するようにコネクタを設定しないでください。

- "10.99.0.0/16"
- "192.168.0.0/16"
- "172.21.0.0/16"

カスタムプラグインの作成

プラグインは、コネクタロジックを定義するコードを含む AWS リソースです。JAR ファイル (または 1 つ以上の JAR ファイルを含む ZIP ファイル) を S3 バケットにアップロードし、プラグインを

作成するときにバケットの場所を指定します。コネクタを作成するときは、MSK Connect で使用するプラグインを指定します。プラグインとコネクタの関係は 1 対多です。同じプラグインから 1 つ以上のコネクタを作成できます。

コネクタのコードを開発する方法については、Apache Kafka ドキュメントのコネクタ開発ガイドを参照してください。

を使用したカスタムプラグインの作成 AWS Management Console

1. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
2. 左側のペインの MSK Connect で、Custom plugins (カスタムプラグイン) を選択します。
3. Create custom plugin (カスタムプラグインの作成) を選択します。
4. Browse S3 (S3 の参照) を選択します。
5. S3 バケットのリストで、プラグインの JAR または ZIP ファイルを含むバケットを選択します。
6. オブジェクトのリストで、プラグインの JAR または ZIP ファイルの左側にあるボックスを選択し、Choose (選択) を選択します。
7. Create custom plugin (カスタムプラグインの作成) を選択します。

MSK Connect API を使用してカスタムプラグインを作成するには、CreateCustomPluginを参照してください。

MSK Connect ワーカーを理解する

ワーカーは、コネクタロジックを実行する Java 仮想マシン (JVM) プロセスです。各ワーカーは、並列スレッドで実行され、データをコピーする作業を行う一連のタスクを作成します。タスクは状態を保存しないため、復元力のあるスケーラブルな Data Pipeline を提供するために、いつでも開始、停止、または再開できます。スケーリングイベントまたは予期しない障害によるワーカー数の変更は、残りのワーカーによって自動的に検出され、残りのワーカーのセット全体でタスクのバランスを取り直すように調整されます。Connect ワーカーは、Apache Kafka のコンシューマーグループを使用して、調整とリバランスを行います。

コネクタの容量要件が可変するか、見積もりが難しい場合は、MSK Connect に、指定した下限と上限の間で必要に応じてワーカー数をスケールさせることができます。または、コネクタロジックを実行するワーカーの正確な数を指定することもできます。詳細については、「the section called “コネクタ容量を理解する”」を参照してください。

MSK Connect ワーカーによる IP アドレスの消費

MSK Connect ワーカーは、カスタマー提供のサブネット内の IP アドレスを消費します。各ワーカーは、カスタマー提供のサブネットの 1 つに含まれている 1 つの IP アドレスを使用します。CreateConnector リクエストに応答して提供されたサブネットに使用可能な IP アドレスが十分にあり、指定した容量を満たしていることを確認する必要があります。特に、ワーカー数が変動する可能性のあるコネクタを自動スケーリングする場合は注意が必要です。

デフォルトのワーカー構成

MSK Connect は、次のデフォルトのワーカー構成を提供します。

```
key.converter=org.apache.kafka.connect.storage.StringConverter
value.converter=org.apache.kafka.connect.storage.StringConverter
```

サポートされているワーカー設定プロパティ

MSK Connect は、デフォルトのワーカー構成を提供します。コネクタで使用するカスタムワーカー設定を作成することもできます。次のリストには、Amazon MSK Connect がサポートする、またはサポートしないワーカー設定プロパティに関する情報が含まれています。

- `key.converter` プロパティと `value.converter` プロパティが必要です。
- MSK Connect は、次の `producer.` 設定プロパティをサポートしています。

```
producer.acks
producer.batch.size
producer.buffer.memory
producer.compression.type
producer.enable.idempotence
producer.key.serializer
producer.linger.ms
producer.max.request.size
producer.metadata.max.age.ms
producer.metadata.max.idle.ms
producer.partitioner.class
producer.reconnect.backoff.max.ms
producer.reconnect.backoff.ms
producer.request.timeout.ms
producer.retry.backoff.ms
producer.value.serializer
```

- MSK Connect は、次の consumer. 設定プロパティをサポートしています。

```
consumer.allow.auto.create.topics
consumer.auto.offset.reset
consumer.check.crcs
consumer.fetch.max.bytes
consumer.fetch.max.wait.ms
consumer.fetch.min.bytes
consumer.heartbeat.interval.ms
consumer.key.deserializer
consumer.max.partition.fetch.bytes
consumer.max.poll.interval.ms
consumer.max.poll.records
consumer.metadata.max.age.ms
consumer.partition.assignment.strategy
consumer.reconnect.backoff.max.ms
consumer.reconnect.backoff.ms
consumer.request.timeout.ms
consumer.retry.backoff.ms
consumer.session.timeout.ms
consumer.value.deserializer
```

- 次のプロパティを除いて、producer. または consumer. プレフィックスでスタートしないすべての設定プロパティが許可されます。

```
access.control.
admin.
admin.listeners.https.
client.
connect.
inter.worker.
internal.
listeners.https.
metrics.
metrics.context.
rest.
sasl.
security.
socket.
ssl.
topic.tracking.
worker.
bootstrap.servers
```

```
config.storage.topic
connections.max.idle.ms
connector.client.config.override.policy
group.id
listeners
metric.reporters
plugin.path
receive.buffer.bytes
response.http.headers.config
scheduled.rebalance.max.delay.ms
send.buffer.bytes
status.storage.topic
```

ワーカー設定プロパティとその表現については、Apache Kafka ドキュメントの「Kafka Connect Config」を参照してください。

カスタムワーカー設定の作成

この手順では、 AWS Management Consoleを使用してカスタムワーカー設定を作成する方法について説明します。

を使用したカスタムワーカー設定の作成 AWS Management Console

1. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
2. 左側のペインの MSK Connect で、Worker configurations (ワーカー構成) を選択します。
3. Create worker configuration (ワーカー構成の作成) を選択します。
4. 名前とオプションの説明を入力し、設定するプロパティと値を追加します。
5. Create worker configuration (ワーカー構成の作成) を選択します。

MSK Connect API を使用してワーカー構成を作成するには、CreateWorkerConfigurationを参照してください。

`offset.storage.topic` を使用してソースコネクタオフセットを管理する

このセクションでは、「オフセットストレージトピック」を使用してソースコネクタオフセットを管理するのに役立つ情報を提供します。オフセットストレージトピックは、Kafka Connect がコネクタとタスク設定のオフセットを保存するために使用する内部トピックです。

考慮事項

ソースコネクタオフセットを管理するときは、次の点を考慮してください。

- オフセットストレージトピックを指定するには、ワーカー設定の `offset.storage.topic` の値として、コネクタオフセットが保存される Kafka トピックの名前を指定します。
- コネクタ設定を変更する場合は注意が必要です。ソースコネクタがキーオフセットレコードに対して設定の値を使用している場合、設定値を変更すると、コネクタが意図しない動作をする可能性があります。プラグインのドキュメントを参照することをお勧めします。
- デフォルトのパーティション数のカスタマイズ — `offset.storage.topic` の追加によるワーカー設定のカスタマイズに加えて、オフセットとステータスストレージのトピックのパーティション数をカスタマイズできます。内部トピックのデフォルトパーティションは次のとおりです。
  - `config.storage.topic`: 1、設定不可、単一パーティションのトピックである必要がある
  - `offset.storage.topic`: 25、`offset.storage.partitions` を指定することで設定可能
  - `status.storage.topic`: 5、`status.storage.partitions` を指定することで設定可能
- トピックの手動削除 — Amazon MSK Connect は、コネクタをデプロイするたびに新しい Kafka Connect 内部トピック (トピック名が `__amazon_msk_connect` で始まる) を作成します。`offset.storage.topic` などの内部トピックはコネクタ間で再利用される可能性があるため、削除されたコネクタにアタッチされた古いトピックは自動的に削除されません。ただし、MSK Connect によって作成された未使用の内部トピックは手動で削除できます。内部トピックには `__amazon_msk_connect_<offsets|status|configs>_`*`connector_name`*`_`*`connector_id`* 形式に従って名前が付けられます。

  正規表現 `__amazon_msk_connect_<offsets|status|configs>_`*`connector_name`*`_`*`connector_id`* を使用して内部トピックを削除できます。実行中のコネクタによって現在使用されている内部トピックは削除しないでください。
- MSK Connect によって作成された内部トピックに同じ名前を使用する — オフセットストレージトピックを再利用して以前に作成したコネクタのオフセットを利用する場合は、新しいコネクタに古いコネクタと同じ名前を付ける必要があります。ワーカー設定を使用して `offset.storage.topic` プロパティを設定し、`offset.storage.topic` に同じ名前を割り当て、異なるコネクタ間で再利用することができます。この設定については、「コネクタオフセットを管理する」で説明しています。MSK Connect では、異なるコネクタが `config.storage.topic` と `status.storage.topic` を共有することはできません。これらのトピックは、MSKC で新しいコネクタを作成するたびに作成されます。`__amazon_msk_connect_<status|configs>_`*`connector_name`*`_`*`connector_id`* 形式に従って自動的に名前が付けられるため、作成するコネクタによって名前が異なります。

デフォルトのオフセットストレージトピックを使用する

デフォルトでは、Amazon MSK Connect は、作成したコネクタごとに Kafka クラスターに新しいオフセットストレージトピックを生成します。MSK は、コネクタ ARN の一部を使用してデフォルトのトピック名を作成します。例えば、`__amazon_msk_connect_offsets_my-mskc-connector_12345678-09e7-4abc-8be8-c657f7e4ff32-2`。

カスタムオフセットストレージトピックを使用する

ソースコネクタ間のオフセットの連続性を提供するために、デフォルトトピックの代わりに任意のオフセットストレージトピックを使用できます。オフセットストレージトピックを指定すると、前のコネクタの最後のオフセットから読み取りを再開するソースコネクタを作成するといったタスクを実行しやすくなります。

オフセットストレージトピックを指定するには、コネクタを作成する前にワーカー設定で `offset.storage.topic` プロパティの値を指定します。オフセットストレージトピックを再利用して以前に作成したコネクタのオフセットを利用する場合は、新しいコネクタに古いコネクタと同じ名前を付ける必要があります。カスタムオフセットストレージトピックを作成する場合は、トピック設定で `cleanup.policy` を `compact` に設定する必要があります。

> **Note**
>
> シンクコネクタの作成時にオフセットストレージトピックを指定すると、トピックがまだ存在しない場合は MSK Connect によってそのトピックが作成されます。ただし、このトピックはコネクタオフセットの保存には使用されません。
> 代わりに、シンクコネクタオフセットは Kafka コンシューマーグループプロトコルを使用して管理されます。各シンクコネクタは `connect-{CONNECTOR_NAME}` という名前のグループを作成します。コンシューマーグループが存在する限り、同じ `CONNECTOR_NAME` 値で連続して作成されるシンクコネクタは、最後にコミットされたオフセットから継続されます。

Example : オフセットストレージトピックを指定し、更新された設定を使用してソースコネクタを再作成する

変更データキャプチャ (CDC) コネクタがあり、CDC ストリーム内での位置を見失うことなくコネクタ設定を変更するとします。既存のコネクタ設定を更新することはできませんが、そのコネクタを削除して同じ名前で新しいコネクタ設定を作成することはできます。CDC ストリームのどこから読み取りを開始するかを新しいコネクタに伝えるには、ワーカー設定で古いコネクタのオフセットストレージトピックを指定します。次のステップでこのタスクのやり方を説明します。

1. クライアントマシンで、次のコマンドを実行してコネクタのオフセットストレージトピックの名前を検索します。*<bootstrapBrokerString>* をクラスターのブートストラップブローカー文字列に置き換えます。ブートストラップブローカー文字列を取得する手順については、「Amazon MSK クラスターのブートストラップブローカーを取得する」を参照してください。

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --list --bootstrap-
server <bootstrapBrokerString>
```

次の出力は、デフォルトの内部コネクタトピックを含むすべてのクラスタートピックのリストを示しています。この例では、既存の CDC コネクタは MSK Connect によって作成されたデフォルトのオフセットストレージトピックを使用します。オフセットストレージトピックが __amazon_msk_connect_offsets_my-mskc-connector_12345678-09e7-4abc-8be8-c657f7e4ff32-2 と呼ばれるのはこれが理由です。

```
__consumer_offsets
__amazon_msk_canary
__amazon_msk_connect_configs_my-mskc-connector_12345678-09e7-4abc-8be8-
c657f7e4ff32-2
__amazon_msk_connect_offsets_my-mskc-connector_12345678-09e7-4abc-8be8-
c657f7e4ff32-2
__amazon_msk_connect_status_my-mskc-connector_12345678-09e7-4abc-8be8-
c657f7e4ff32-2
my-msk-topic-1
my-msk-topic-2
```

2. https://console.aws.amazon.com/msk/ で Amazon MSK コンソールを開きます。
3. [コネクタ] リストからコネクタを選択します。[コネクタ設定] フィールドの内容をコピーして保存し、内容を変更して新しいコネクタを作成できるようにします。
4. コネクタを削除するには、[削除] を選択します。テキスト入力フィールドにコネクタ名を入力して、削除を確定します。
5. 実際のシナリオに合った値を使用してカスタムワーカー設定を作成します。手順については、「カスタムワーカー設定の作成」を参照してください。

ワーカー設定では、以下の設定のように、以前に `offset.storage.topic` の値として取得したオフセットストレージトピックの名前を指定する必要があります。

```
config.providers.secretManager.param.aws.region=eu-west-3
key.converter=<org.apache.kafka.connect.storage.StringConverter>
value.converter=<org.apache.kafka.connect.storage.StringConverter>
```

```
config.providers.secretManager.class=com.github.jcustenborder.kafka.config.aws.SecretsManag
config.providers=secretManager
offset.storage.topic=__amazon_msk_connect_offsets_my-mskc-
connector_12345678-09e7-4abc-8be8-c657f7e4ff32-2
```

6.

> **Important**
>
> 新しいコネクタには古いコネクタと同じ名前を付ける必要があります。

前のステップで設定したワーカー設定を使用して、新しいコネクタを作成します。手順については、「コネクタを作成する」を参照してください。

チュートリアル: 設定プロバイダーを用いた機密情報の外部化

この例は、オープンソースの設定プロバイダーを使用して Amazon MSK Connect の機密情報を外部化する方法を示しています。設定プロバイダーを使用すると、コネクタまたはワーカー設定でプレーンテキストの代わりに変数を指定でき、コネクタで実行されているワーカーは実行時にこれらの変数を解決します。これにより、認証情報やその他のシークレットがプレーンテキストで保存されるのを防ぐことができます。この例の設定プロバイダーは、 AWS Secrets Manager、Amazon S3、Systems Manager (SSM) からの設定パラメータの取得をサポートしています。ステップ 2 では、設定するサービスの機密情報の保存と取得をセットアップする方法を確認できます。

考慮事項

Amazon MSK Connect で MSK 設定プロバイダーを使用する際には、次の点を考慮してください。

- 設定プロバイダーを使用するときは、IAM サービス実行ロールに適切なアクセス許可を割り当てます。
- ワーカー設定で設定プロバイダーを定義し、コネクタ設定でその実装を定義します。
- プラグインで機密設定値をシークレットとして定義していない場合、機密設定値がコネクタログに表示される可能性があります。Kafka Connect は、未定義の設定値を他のプレーンテキスト値と同じように扱います。詳細についてはシークレットがコネクタログに表示されないようにするを参照してください。
- デフォルトでは、コネクタが設定プロバイダーを使用するときに、MSK Connect はコネクタを頻繁に再起動します。この再起動動作を無効にするには、コネクタ設定で `config.action.reload` の値を `none` に設定します。

カスタムプラグインを作成して S3 にアップロードする

カスタムプラグインを作成するには、ローカルマシンで以下のコマンドを実行して、コネクタと msk-config-provider を含む zip ファイルを作成します。

ターミナルウィンドウと Debezium をコネクタとして使用してカスタムプラグインを作成するには

CLI を使用して、 AWS S3 AWS バケットへのアクセスを許可する認証情報を持つスーパーユーザーとしてコマンドを実行します。CLI のインストールとセットアップの詳細については、 AWS「ユーザーガイド」の「CLI AWS の開始方法」を参照してください。 AWS Command Line Interface Amazon S3 で AWS CLI を使用する方法については、 AWS Command Line Interface ユーザーガイドの「CLI での Amazon S3 AWS の使用」を参照してください。

1. ターミナルウィンドウで、以下のコマンドを使用して custom-plugin という名前のフォルダをワークスペースに作成します。

   ```
   mkdir custom-plugin && cd custom-plugin
   ```

2. 次のコマンドを使用して、Debezium サイトから MySQL コネクタプラグインの最新の安定版リリースをダウンロードします。

   ```
   wget https://repo1.maven.org/maven2/io/debezium/debezium-connectormysql/
   2.2.0.Final/debezium-connector-mysql-2.2.0.Final-plugin.tar.gz
   ```

   次のコマンドを使用して、ダウンロードした gzip ファイルを custom-plugin フォルダに解凍します。

   ```
   tar xzf debezium-connector-mysql-2.2.0.Final-plugin.tar.gz
   ```

3. 次のコマンドを使用して、MSK 設定プロバイダーの zip ファイルをダウンロードします。

   ```
   wget https://github.com/aws-samples/msk-config-providers/releases/download/r0.4.0/
   msk-config-providers-0.4.0-with-dependencies.zip
   ```

   次のコマンドを使用して、ダウンロードした zip ファイルを custom-plugin フォルダに解凍します。

   ```
   unzip msk-config-providers-0.4.0-with-dependencies.zip
   ```

4. 上記のステップで取得した MSK 設定プロバイダーとカスタムコネクタの内容を、`custom-plugin.zip` という名前を付けた単一のファイルに圧縮します。

```
zip -r ../custom-plugin.zip *
```

5. このファイルを後で参照できるように S3 にアップロードします。

```
aws s3 cp ../custom-plugin.zip s3:<S3_URI_BUCKET_LOCATION>
```

6. Amazon MSK コンソールの [MSK Connect] セクションで [カスタムプラグイン] を選択し、次に [カスタムプラグインの作成] を選択して s3:<***S3_URI_BUCKET_LOCATION***> S3 バケットを参照し、アップロードしたカスタムプラグインの ZIP ファイルを選択します。

7. プラグイン名には **debezium-custom-plugin** と入力します。オプションで説明を入力し、[カスタムプラグインの作成] を選択します。

さまざまなプロバイダーのパラメータとアクセス許可を設定する

次の 3 つのサービスでパラメータ値を設定できます。

- Secrets Manager
- Systems Manager Parameter Store
- S3 - Simple Storage Service

以下のタブのいずれかを選択して、そのサービスのパラメータと関連するアクセス許可を設定する手順を確認してください。

Configure in Secrets Manager

Secrets Manager でパラメータ値を設定するには

1. Secrets Manager コンソールを開きます。
2. 認証情報またはシークレットを保存する新しいシークレットを作成します。手順については、「 AWS Secrets Manager ユーザーガイド」の「 AWS Secrets Manager シークレットの作成」を参照してください。
3. シークレットの ARN をコピーします。

4. 以下のサンプルポリシーの Secrets Manager のアクセス許可をサービス実行ロールに追加します。サンプル ARN、 をarn:aws:secretsmanager:*us-east-1*:*123456789012*:secret:*MySecret-1234*シークレットの ARN に置き換えます。
5. ワーカー設定とコネクタの指示を追加します。

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "secretsmanager:GetResourcePolicy",
                    "secretsmanager:GetSecretValue",
                    "secretsmanager:DescribeSecret",
                    "secretsmanager:ListSecretVersionIds"
                ],
                "Resource": [
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:MySecret-1234"
                ]
            }
        ]
    }
```

6. Secrets Manager 設定プロバイダーを使用するには、ステップ 3 のワーカー設定テキストボックスに次のコード行をコピーします。

```
# define name of config provider:

config.providers = secretsmanager

# provide implementation classes for secrets manager:

config.providers.secretsmanager.class =
 com.amazonaws.kafka.config.providers.SecretsManagerConfigProvider

# configure a config provider (if it needs additional initialization), for
 example you can provide a region where the secrets or parameters are located:

config.providers.secretsmanager.param.region = us-east-1
```

7. Secrets Manager 設定プロバイダーで、ステップ 4 のコネクタ設定にある次のコード行をコピーします。

```
#Example implementation for secrets manager variable
database.user=${secretsmanager:MSKAuroraDBCredentials:username}
database.password=${secretsmanager:MSKAuroraDBCredentials:password}
```

上記のステップは、他の設定プロバイダーでも使用できます。

Configure in Systems Manager Parameter Store

Systems Manager のパラメータストアでパラメータ値を設定するには

1. Systems Manager コンソールを開きます。
2. ナビゲーションペインで、[パラメータストア] を選択します。
3. Systems Manager に保存する新しいパラメータを作成します。手順については、「 AWS Systems Manager ユーザーガイド」の「Systems Manager パラメータの作成 (コンソール) 」を参照してください。
4. パラメータの ARN をコピーします。
5. 以下のサンプルポリシーの Secrets Manager のアクセス許可をサービス実行ロールに追加します。*<arn:aws:ssm:us-east-1:123456789000:parameter/MyParameterName>* をパラメータの ARN で置き換えます。

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "VisualEditor0",
                "Effect": "Allow",
                "Action": [
                    "ssm:GetParameterHistory",
                    "ssm:GetParametersByPath",
                    "ssm:GetParameters",
                    "ssm:GetParameter"
                ],
                "Resource": "arn:aws:ssm:us-east-1:123456789000:parameter/
MyParameterName"
            }
        ]
```

```
    }
```

6. パラメータストア設定プロバイダーを使用するには、ステップ 3 のワーカー設定テキストボックスに次のコード行をコピーします。

```
# define name of config provider:

config.providers = ssm

# provide implementation classes for parameter store:

config.providers.ssm.class =
 com.amazonaws.kafka.config.providers.SsmParamStoreConfigProvider

# configure a config provider (if it needs additional initialization), for
 example you can provide a region where the secrets or parameters are located:

config.providers.ssm.param.region = us-east-1
```

7. パラメータストア設定プロバイダーで、ステップ 5 のコネクタ設定にある次のコード行をコピーします。

```
#Example implementation for parameter store variable
schema.history.internal.kafka.bootstrap.servers=
${ssm::MSKBootstrapServerAddress}
```

上記の 2 つのステップを他の設定プロバイダーにバンドルすることもできます。

Configure in Amazon S3

Amazon S3 のオブジェクト/ファイルを設定するには

1. Amazon S3 コンソールを開きます。
2. オブジェクトを S3 のバケットにアップロードします。手順については、「オブジェクトのアップロード」を参照してください。
3. オブジェクトの ARN をコピーします。
4. 以下のサンプルポリシーの Amazon S3 オブジェクト読み取りアクセス許可をサービス実行ロールに追加します。サンプル ARN、 `arn:aws:s3:::<MY_S3_BUCKET/path/to/custom-plugin.zip>`を オブジェクトの ARN に置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
            {
                "Sid": "VisualEditor0",
                "Effect": "Allow",
                "Action": "s3:GetObject",
                "Resource": "arn:aws:s3:::<MY_S3_BUCKET/path/to/custom-
plugin.zip>"
            }
        ]
    }
```

5. Amazon S3 設定プロバイダーを使用するには、ステップ 3 のワーカー設定テキストボックスに次のコード行をコピーします。

```
# define name of config provider:

config.providers = s3import
# provide implementation classes for S3:

config.providers.s3import.class =
 com.amazonaws.kafka.config.providers.S3ImportConfigProvider
```

6. Amazon S3 設定プロバイダーで、次のコード行をステップ 4 のコネクタ設定にコピーします。

```
#Example implementation for S3 object

database.ssl.truststore.location = ${s3import:us-west-2:my_cert_bucket/path/to/
trustore_unique_filename.jks}
```

上記の 2 つのステップを他の設定プロバイダーにバンドルすることもできます。

設定プロバイダーに関する情報を使用してカスタムワーカー設定を作成します。

1. [Amazon MSK Connect] セクションで [ワーカー設定] を選択します。

2. [ワーカー設定の作成] を選択します。
3. [ワーカー設定名] テキストボックスに SourceDebeziumCustomConfig を入力します。説明はオプションです。
4. 必要なプロバイダーに基づいて関連する設定コードをコピーし、[ワーカー設定] テキストボックスに貼り付けます。
5. 3 つのプロバイダーすべてのワーカー設定の例を以下に示します。

```
key.converter=org.apache.kafka.connect.storage.StringConverter
key.converter.schemas.enable=false
value.converter=org.apache.kafka.connect.json.JsonConverter
value.converter.schemas.enable=false
offset.storage.topic=offsets_my_debezium_source_connector

# define names of config providers:

config.providers=secretsmanager,ssm,s3import

# provide implementation classes for each provider:

config.providers.secretsmanager.class    =
 com.amazonaws.kafka.config.providers.SecretsManagerConfigProvider
config.providers.ssm.class               =
 com.amazonaws.kafka.config.providers.SsmParamStoreConfigProvider
config.providers.s3import.class          =
 com.amazonaws.kafka.config.providers.S3ImportConfigProvider

# configure a config provider (if it needs additional initialization), for example
 you can provide a region where the secrets or parameters are located:


config.providers.secretsmanager.param.region = us-east-1
config.providers.ssm.param.region = us-east-1
```

6. [ワーカー設定の作成] をクリックします。

コネクタを作成する

1. 「Create a new connector」の手順に従って新しいコネクタを作成します。
2. ??? で S3 バケットにアップロードした custom-plugin.zip ファイルをカスタムプラグインのソースとして選択します。

3. 必要なプロバイダーに基づいて関連する設定コードをコピーし、[コネクタの設定] フィールドに貼り付けます。
4. 3 つのプロバイダーすべてのコネクタ設定の例を以下に示します。

```
#Example implementation for parameter store variable
schema.history.internal.kafka.bootstrap.servers=${ssm::MSKBootstrapServerAddress}

#Example implementation for secrets manager variable
database.user=${secretsmanager:MSKAuroraDBCredentials:username}
database.password=${secretsmanager:MSKAuroraDBCredentials:password}

#Example implementation for Amazon S3 file/object
database.ssl.truststore.location = ${s3import:us-west-2:my_cert_bucket/path/to/
trustore_unique_filename.jks}
```

5. [ワーカー設定] ドロップダウンから [カスタム設定を使用] を選択し、[SourceDebeziumCustomConfig] を選択します。
6. 「Create connector」の残りの手順に従います。

MSK Connect の IAM のロールとポリシー

このセクションでは、 AWS 環境内で Amazon MSK Connect を安全にデプロイおよび管理するための適切な IAM ポリシーとロールを設定するのに役立ちます。以下のセクションでは、MSK Connect で使用する必要があるサービス実行ロールについて説明します。これには、IAM 認証 MSK クラスターに接続するときに必要な信頼ポリシーと追加のアクセス許可が含まれます。このページでは、MSK Connect 機能へのフルアクセスを許可するための包括的な IAM ポリシーの例と、サービスで使用できる AWS マネージドポリシーの詳細も示します。

トピック

- サービス実行ロールを理解する
- MSK Connect の IAM ポリシーの例
- サービス間での混乱した代理問題を防止する
- AWS MSK Connect の マネージドポリシー
- MSK Connect のサービスにリンクされたロールの使用

サービス実行ロールを理解する

> **Note**
>
> Amazon MSK Connect は、サービスにリンクされたロールをサービス実行ロールとして使用することをサポートしていません。サービス実行ロールは別途作成する必要があります。カスタム IAM ロールを作成する手順については、IAM ユーザーガイドの「 AWS サービスにアクセス許可を委任するロールの作成」を参照してください。

MSK Connect でコネクタを作成するときは、それで使用する AWS Identity and Access Management (IAM) ロールを指定する必要があります。MSK Connect が継承できるように、サービス実行ロールには次の信頼ポリシーが必要です。このポリシーの条件コンテキストキーの詳細については、「the section called “サービス間での混乱した代理問題を防止する”」を参照してください。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kafkaconnect.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:kafkaconnect:us-
east-1:123456789012:connector/myConnector/abc12345-abcd-4444-a8b9-123456f513ed-2"
        }
      }
    }
  ]
}
```

コネクタで使用する Amazon MSK クラスターが IAM 認証を使用するクラスターである場合は、次の許可ポリシーをコネクタのサービス実行ロールに追加する必要があります。クラスターの UUID を検索する方法とトピック ARNs「」を参照してくださいthe section called “認可ポリシーリソース”。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:Connect",
                "kafka-cluster:DescribeCluster"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:000000000001:cluster/
testClusterName/300d0000-0000-0005-000f-00000000000b-1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:ReadData",
                "kafka-cluster:DescribeTopic"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:topic/
myCluster/300a0000-0000-0003-000a-00000000000b-6/__amazon_msk_connect_read"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:WriteData",
                "kafka-cluster:DescribeTopic"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:topic/
testCluster/300f0000-0000-0008-000d-00000000000m-7/__amazon_msk_connect_write"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:CreateTopic",
                "kafka-cluster:WriteData",
                "kafka-cluster:ReadData",
                "kafka-cluster:DescribeTopic"
            ],
            "Resource": [
                "arn:aws:kafka:us-
east-1:123456789012:topic/testCluster/300f0000-0000-0008-000d-00000000000m-7/
__amazon_msk_connect_*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:AlterGroup",
                "kafka-cluster:DescribeGroup"
            ],
            "Resource": [
                "arn:aws:kafka:us-
east-1:123456789012:group/testCluster/300d0000-0000-0005-000f-00000000000b-1/
__amazon_msk_connect_*",
                "arn:aws:kafka:us-
east-1:123456789012:group/testCluster/300d0000-0000-0005-000f-00000000000b-1/
connect-*"
            ]
        }
    ]
}
```

コネクタの種類によっては、 AWS リソースへのアクセスを許可するアクセス許可ポリシーをサービス実行ロールにアタッチする必要もあります。例えば、コネクタが S3 バケットにデータを送信する必要がある場合、サービス実行ロールには、そのバケットへの書き込み許可を付与する許可ポリシーが必要です。テストの目的で、arn:aws:iam::aws:policy/AmazonS3FullAccess のように、フルアクセスを提供する事前に構築された IAM ポリシーの 1 つを使用できます。ただし、セキュリティ上の理由から、コネクタが AWS ソースから読み取るか、 AWS シンクに書き込むことを許可する、最も制限の厳しいポリシーを使用することをお勧めします。

MSK Connect の IAM ポリシーの例

管理者以外のユーザーにすべての MSK Connect 機能へのフルアクセスを許可するには、次のようなポリシーをユーザーの IAM ロールにアタッチします。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MSKConnectFullAccess",
      "Effect": "Allow",
      "Action": [
        "kafkaconnect:CreateConnector",
        "kafkaconnect:DeleteConnector",
        "kafkaconnect:DescribeConnector",
        "kafkaconnect:ListConnectors",
        "kafkaconnect:UpdateConnector",
        "kafkaconnect:CreateCustomPlugin",
        "kafkaconnect:DeleteCustomPlugin",
        "kafkaconnect:DescribeCustomPlugin",
        "kafkaconnect:ListCustomPlugins",
        "kafkaconnect:CreateWorkerConfiguration",
        "kafkaconnect:DeleteWorkerConfiguration",
        "kafkaconnect:DescribeWorkerConfiguration",
        "kafkaconnect:ListWorkerConfigurations"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAMPassRole",
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/MSKConnectServiceRole",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "kafkaconnect.amazonaws.com"
        }
      }
    },
    {
```

```
      "Sid": "EC2NetworkAccess",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DeleteNetworkInterface",
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups"
      ],
      "Resource": "*"
    },
    {
      "Sid": "MSKClusterAccess",
      "Effect": "Allow",
      "Action": [
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka:GetBootstrapBrokers"
      ],
      "Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/myCluster/"
    },
    {
      "Sid": "MSKLogGroupAccess",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:log-group:/aws/msk-connect/*"
      ]
    },
    {
      "Sid": "S3PluginAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject"
      ],
```

```
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1-custom-plugins",
        "arn:aws:s3:::amzn-s3-demo-bucket1-custom-plugins/*"
      ]
    }
  ]
}
```

サービス間での混乱した代理問題を防止する

混乱した代理問題は、アクションを実行する許可を持たないエンティティが、より特権のあるエンティティにアクションを実行するように強制できるセキュリティの問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐため、 AWS では、アカウントのリソースへのアクセス権が付与されたサービスプリンシパルで、すべてのサービスのデータを保護するために役立つツールを提供しています。

リソースポリシーで aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストキーを使用して、MSK Connect が別のサービスに付与するアクセス許可をそのリソースに制限することをお勧めします。aws:SourceArn 値にアカウント ID が含まれていない (例: Amazon S3 バケット ARN にアカウント ID が含まれていない) 場合、アクセス許可を制限するため、両方のグローバル条件コンテキストキーを使用する必要があります。同じポリシーステートメントでこれらのグローバル条件コンテキストキーの両方を使用し、アカウント ID にaws:SourceArn の値が含まれていない場合、aws:SourceAccount 値と aws:SourceArn 値の中のアカウントには、同じアカウント ID を使用する必要があります。クロスサービスのアクセスにリソースを 1 つだけ関連付けたい場合は、aws:SourceArn を使用します。そのアカウント内のリソースをクロスサービスの使用に関連付けることを許可する場合は、aws:SourceAccount を使用します。

MSK Connect の場合、aws:SourceArn の値は MSK コネクタである必要があります。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して aws:SourceArn グローバル条件コンテキストキーを使用することです。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合は、aws:SourceArn グローバルコンテキスト条件キーを使用して、ARN の未知部分をワイルドカード (*) で表します。例え

ば、*arn:aws:kafkaconnect:us-east-1:123456789012:connector/** は、米国東部 (バージニア北部) リージョンの ID 123456789012 のアカウントに属するすべてのコネクタを表します。

次の例では、MSK Connect で aws:SourceArn および aws:SourceAccount グローバル条件コンテキストキーを使用して、混乱した代理問題を回避する方法を示します。*123456789012* と arn:aws:kafkaconnect:*us-east-1*：*123456789012*:connector/*my-S3-Sink-Connector*/*abcd1234-5678-90ab-cdef-1234567890ab* を AWS アカウント および コネクタ情報に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": " kafkaconnect.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
        "aws:SourceArn": "arn:aws:kafkaconnect:us-
east-1:123456789012:connector/my-S3-Sink-Connector/abcd1234-5678-90ab-
cdef-1234567890ab"
        }
      }
    }
  ]
}
```

AWS MSK Connect の マネージドポリシー

AWS 管理ポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。 AWS 管理ポリシーは、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できるように、多くの一般的なユースケースにアクセス許可を付与するように設計されています。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることに注意してください。ユースケースに固有のカスタマー管理ポリシーを定義して、アクセス許可を絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS 管理ポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) に影響します。 AWS は、新しい が起動されるか、新しい API オペレーション AWS のサービス が既存のサービスで使用できるようになったときに、 AWS 管理ポリシーを更新する可能性が高くなります。

詳細については「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWS マネージドポリシー: AmazonMSKConnectReadOnlyAccess

このポリシーは、MSK Connect リソースを一覧表示および説明するために必要なアクセス許可をユーザーに付与します。

AmazonMSKConnectReadOnlyAccess ポリシーを IAM アイデンティティにアタッチできます。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kafkaconnect:ListConnectors",
                "kafkaconnect:ListCustomPlugins",
                "kafkaconnect:ListWorkerConfigurations"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafkaconnect:DescribeConnector"
            ],
            "Resource": [
                "arn:aws:kafkaconnect:*:*:connector/*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "kafkaconnect:DescribeCustomPlugin"
            ],
            "Resource": [
                "arn:aws:kafkaconnect:*:*:custom-plugin/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafkaconnect:DescribeWorkerConfiguration"
            ],
            "Resource": [
                "arn:aws:kafkaconnect:*:*:worker-configuration/*"
            ]
        }
    ]
}
```

AWS マネージドポリシー: KafkaConnectServiceRolePolicy

このポリシーは、タグ `AmazonMSKConnectManaged:true` を持つネットワークインターフェイスを作成および管理するために必要なアクセス許可を MSK Connect サービスに付与します。これらのネットワークインターフェースにより、MSK Connect ネットワークは Apache Kafka クラスターやソースまたはシンクなどの Amazon VPC 内のリソースにアクセスできます。

KafkaConnectServiceRolePolicy を IAM エンティティにアタッチすることはできません。このポリシーは、MSK Connect がユーザーに代わってアクションを実行できるようにするサービスにリンクされたロールに関連付けられています。

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterface"
```

```
    ],
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
     "StringEquals": {
      "aws:RequestTag/AmazonMSKConnectManaged": "true"
     },
     "ForAllValues:StringEquals": {
      "aws:TagKeys": "AmazonMSKConnectManaged"
     }
    }
   },
   {
    "Effect": "Allow",
    "Action": [
     "ec2:CreateNetworkInterface"
    ],
    "Resource": [
     "arn:aws:ec2:*:*:subnet/*",
     "arn:aws:ec2:*:*:security-group/*"
    ]
   },
   {
    "Effect": "Allow",
    "Action": [
     "ec2:CreateTags"
    ],
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
     "StringEquals": {
      "ec2:CreateAction": "CreateNetworkInterface"
     }
    }
   },
   {
    "Effect": "Allow",
    "Action": [
     "ec2:DescribeNetworkInterfaces",
     "ec2:CreateNetworkInterfacePermission",
     "ec2:AttachNetworkInterface",
     "ec2:DetachNetworkInterface",
     "ec2:DeleteNetworkInterface"
    ],
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
```

```
    "StringEquals": {
     "ec2:ResourceTag/AmazonMSKConnectManaged": "true"
    }
   }
  }
 ]
}
```

MSK Connect による AWS 管理ポリシーの更新

このサービスがこれらの変更の追跡を開始してからの MSK Connect の AWS マネージドポリシーの更新に関する詳細を表示します。

変更	説明	日付
MSK Connect の更新された読み取り専用ポリシー	MSK Connect は、AmazonMSKConnectReadOnlyAccess ポリシーを更新して、リストオペレーションの制限を削除しました。	2021 年 10 月 13 日
MSK Connect が変更の追跡をスタートしました	MSK Connect は AWS 、管理ポリシーの変更の追跡を開始しました。	2021 年 9 月 14 日

MSK Connect のサービスにリンクされたロールの使用

Amazon MSK Connect は AWS Identity and Access Management (IAM) サービスにリンクされたロールを使用します。サービスにリンクされたロールは、MSK Connect に直接リンクされている一意のタイプの IAM ロールです。サービスにリンクされたロールは MSK Connect によって事前定義されており、サービスがユーザーに代わって他の AWS サービスを呼び出すために必要なすべてのアクセス許可が含まれています。

サービスにリンクされたロールにより、必要な許可を手動で追加する必要がないため、MSK Connect の設定が簡単になります。MSK Connect は、サービスにリンクされたロールの許可を定

義します。特に定義されていない限り、MSK Connect のみがその口ールを引き受けることができます。定義されたアクセス許可には、信頼ポリシーとアクセス許可ポリシーが含まれ、そのアクセス許可ポリシーを他の IAM エンティティに添付することはできません。

サービスリンクロールをサポートする他のサービスについては、「IAM と連携するAWS のサービス」を参照して、サービスリンクロール列がはいになっているサービスを探してください。そのサービスのサービスにリンクされたロールのドキュメントを表示するには、リンク付きのはいを選択します。

MSK Connect のサービスにリンクされたロールのアクセス許可

MSK Connect は、AWSServiceRoleForKafkaConnect という名前のサービスにリンクされたロールを使用します – Amazon MSK Connect がユーザーに代わって Amazon リソースにアクセスできるようにします。

AWSServiceRoleForKafkaConnect サービスにリンクされたロール
は、`kafkaconnect.amazonaws.com` サービスを信頼してロールを引き受けます。

ロールが使用する権限ポリシーについては、「the section called “KafkaConnectServiceRolePolicy”」を参照してください。

サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、アクセス許可を設定する必要があります。詳細については、「IAMユーザーガイド」のサービスにリンクされたロールのアクセス許可を参照してください。

MSK Connect のサービスにリンクされたロールの作成

サービスリンクロールを手動で作成する必要はありません。 AWS Management Console、 AWS CLI、または AWS API でコネクタを作成すると、MSK Connect によってサービスにリンクされたロールが作成されます。

このサービスにリンクされたロールを削除してから再度作成する必要がある場合は、同じプロセスを使用してアカウントにロールを再作成できます。コネクタを作成すると、MSK Connect はサービスにリンクされたロールを再度作成します。

MSK Connect のサービスにリンクされたロールの編集

MSK Connect では、AWSServiceRoleForKafkaConnect サービスにリンクされたロールを編集することはできません。サービスにリンクされたロールを作成した後は、さまざまなエンティティがロールをリファレンスする可能性があるため、ロールの名前を変更することはできません。ただし、IAM

を使用したロール記述の編集はできます。詳細については、「IAM ユーザーガイド」のサービスにリンクされたロールの編集を参照してください。

MSK Connect のサービスにリンクされたロールの削除

IAM コンソール、、 AWS CLI または AWS API を使用して、サービスにリンクされたロールを手動で削除できます。これを行うには、最初にすべての MSK Connect コネクタを手動で削除する必要があります。次に、ロールを手動で削除できます。詳細については、「IAM ユーザーガイド」のサービスにリンクされたロールの削除を参照してください。

MSK Connect のサービスにリンクされたロールがサポートされているリージョン

MSK Connect は、サービスが利用可能なすべてのリージョンでサービスにリンクされたロールの使用をサポートします。詳細については、「AWS リージョンとエンドポイント」を参照してください。

Amazon MSK Connect のインターネットアクセスを有効にする

Amazon MSK Connect のコネクタがインターネットにアクセスする必要がある場合は、次の Amazon Virtual Private Cloud (VPC) 設定を使用してそのアクセスを有効にすることをお勧めします。

- コネクタをプライベートサブネットで設定します。
- パブリックサブネットの VPC 用にパブリック NAT ゲートウェイまたは NAT インスタンスを作成します。詳細については、「Amazon Virtual Private Cloudユーザーガイド」の「Connect subnets to the internet or other VPCs using NAT devices」ページを参照してください。
- プライベートサブネットから NAT ゲートウェイまたはインスタンスへのアウトバウンドトラフィックを許可します。

Amazon MSK Connect 用の NAT ゲートウェイを設定する

次のステップは、NAT ゲートウェイをセットアップしてコネクタのインターネットアクセスを有効にする方法を示しています。プライベートサブネットにコネクタを作成する前に、次のステップを完了する必要があります。

NAT ゲートウェイを設定するための完全な前提条件

以下があることを確認します。

- クラスターに関連付けられている Amazon Virtual Private Cloud (VPC) の ID。例えば、vpc-123456ab などです。
- VPC 内のプライベートサブネットの ID。例えば、subnet-a1b2c3de、subnet-f4g5h6ij などです。コネクタにはプライベートサブネットを設定する必要があります。

コネクタのインターネットアクセスを有効にするための手順

コネクタのインターネットアクセスを有効にするには

1. https://console.aws.amazon.com/vpc/ で Amazon Virtual Private Cloud コンソールを開きます。
2. わかりやすい名前を付けて NAT ゲートウェイのパブリックサブネットを作成し、サブネット ID を書き留めます。詳細な手順については、「VPC にサブネットを作成する」を参照してください。
3. VPC がインターネットと通信できるようにインターネットゲートウェイを作成し、ゲートウェイ ID を書き留めます。VPC にインターネットゲートウェイをアタッチします。手順については、「インターネットゲートウェイの作成とアタッチ」を参照してください。
4. プライベートサブネット内のホストがパブリックサブネットにアクセスできるように、パブリック NAT ゲートウェイをプロビジョニングします。NAT ゲートウェイを作成するときに、前に作成したパブリックサブネットを選択します。手順については、「NAT ゲートウェイの作成」を参照してください。
5. ルートテーブルを設定します。この設定を完了するには、合計で 2 つのルートテーブルが必要です。VPC と同時に自動的に作成されたメインのルートテーブルが既にあるはずです。このステップでは、パブリックサブネット用の追加のルートテーブルを作成します。

   a. 次の設定を使用して VPC のメインルートテーブルを変更し、プライベートサブネットがトラフィックを NAT ゲートウェイにルーティングするようにします。手順については、「Amazon Virtual Private Cloudユーザーガイド」の「ルートテーブルの使用」を参照してください。

      プライベート MSKC ルートテーブル

プロパティ	値
名前タグ	このルートテーブルには、識別しやすいようにわかりやすい名前タグを付けることをお勧めします。例えば、Private MSKC などです。

プロパティ	値
関連付けられたサブネット	プライベートサブネット
MSK Connect のインターネットアクセスを有効にするためのルート	• 送信先: 0.0.0.0/0 • ターゲット: NAT ゲートウェイ ID。例えば、nat-12a345bc6789efg1h です。
内部トラフィックのルート	• 送信先: 10.0.0.0/16 この値は VPC の CIDR ブロックによって異なる場合があります。 • ターゲット: ローカル

b. 「カスタムルートテーブルを作成する」の手順に従って、パブリックサブネットのルートテーブルを作成します。テーブルを作成するときは、そのテーブルがどのサブネットに関連付けられているかを識別しやすいように、[名前タグ] フィールドにわかりやすい名前を入力します。例えば、パブリック MSKC と入力します。

c. 以下の設定を使用してパブリック MSKC のルートテーブルを設定します。

プロパティ	値
名前タグ	パブリック MSKC または選択した別のわかりやすい名前を使用してください。
関連付けられたサブネット	NAT ゲートウェイを使用するパブリックサブネット
MSK Connect のインターネットアクセスを有効にするためのルート	• 送信先: 0.0.0.0/0 • ターゲット: インターネットゲートウェイ ID。例えば igw-1a234bc5 です。
内部トラフィックのルート	• 送信先: 10.0.0.0/16 この値は VPC の CIDR ブロックによって異なる場合があります。 • ターゲット: ローカル

プライベート DNS ホスト名を理解する

MSK Connect のプライベート DNS ホスト名のサポートにより、パブリックドメイン名またはプライベートドメイン名を参照するようにコネクタを設定できます。サポートは VPC DHCP オプションセットで指定されている DNS サーバーによって異なります。

DHCP オプションセットは、EC2 インスタンスが VPC で VPC ネットワーク経由で通信するために使用するネットワーク構成のグループです。各 VPC にはデフォルトの DHCP オプションセットがありますが、例えば、VPC 内のインスタンスで Amazon が提供する DNS サーバーではない別の DNS サーバーを使用してドメイン名解決を行う場合は、カスタム DHCP オプションセットを作成できます。「Amazon VPC の DHCP オプションセット」を参照してください。

プライベート DNS 解決機能が MSK Connect に組み込まれる前は、コネクタでは顧客のコネクタからの DNS クエリにサービス VPC DNS リゾルバーが使用されていました。コネクタが、顧客の VPC DHCP オプションセットで定義されている DNS サーバーを DNS 解決に使用することはありませんでした。

コネクタが参照できるのは、顧客のコネクタ設定内のホスト名またはパブリックに解決可能なプラグインのホスト名のみでした。プライベートホストゾーンで定義されたプライベートホスト名を解決したり、別の顧客ネットワークの DNS サーバーを使用したりすることはできませんでした。

顧客が自社の VPC 内のデータベース、データウェアハウス、Secrets Manager などのシステムをインターネットアクセス不可とすることを選択した場合、プライベート DNS がなければ MSK コネクタと連携できません。顧客は、企業のセキュリティ体制に準拠するためにプライベート DNS ホスト名を使用することがよくあります。

コネクタの VPC DHCP オプションセットを設定する

コネクタは、コネクタの作成時に VPC DHCP オプションセットで定義されている DNS サーバーを自動的に使用します。コネクタを作成する前に、コネクタの DNS ホスト名の解決要件に応じて VPC DHCP オプションセットを設定してください。

プライベート DNS ホスト名機能が MSK Connect で使用できるようになる前に作成したコネクタは、以前の DNS 解決設定を変更不要で引き続き使用します。

パブリックで解決可能な DNS ホスト名解決のみがコネクタで必要な場合は、セットアップを簡単にするために、コネクタの作成時にアカウントのデフォルト VPC を使用することをお勧めします。アマゾンが提供する DNS サーバーまたは Amazon Route 53 Resolver の詳細については、「Amazon VPC ユーザーガイド」の「Amazon DNS サーバー」を参照してください。

プライベート DNS ホスト名を解決する必要がある場合は、コネクタの作成時に渡される VPC の DHCP オプションが正しく設定されていることを確認してください。詳細については、「Amazon VPC ユーザーガイド」の「DHCP オプションセットの使用」を参照してください。

プライベート DNS ホスト名解決用に DHCP オプションセットを設定する場合、コネクタが DHCP オプションセットで設定したカスタム DNS サーバーにアクセスできることを確認してください。アクセスできない場合、コネクタの作成は失敗します。

VPC DHCP オプションセットをカスタマイズすると、その VPC でその後作成されるコネクタでは、オプションセットで指定した DNS サーバーが使用されます。コネクタを作成した後にオプションセットを変更すると、コネクタは数分以内に新しいオプションセットの設定を採用します。

VPC の DNS 属性を設定する

「Amazon VPC ユーザーガイド」の「VPC 内の DNS 属性」と「DNS ホスト名」の説明に従って、VPC DNS 属性が正しく設定されていることを確認します。

インバウンドとアウトバウンドのリゾルバーエンドポイントを使用して他のネットワークを VPC に接続してコネクタと連携させる方法については、「Amazon Route 53 開発者ガイド」の「VPC とネットワークの間における DNS クエリの解決」を参照してください。

コネクタ作成の失敗を処理する

このセクションでは、DNS 解決に関連して発生する可能性のあるコネクタ作成の失敗と、問題を解決するための推奨処置について説明します。

失敗	推奨されるアクション
DNS 解決クエリが失敗した場合、またはコネクタから DNS サーバーにアクセスできない場合、コネクタの作成は失敗します。	DNS 解決クエリの失敗によるコネクタ作成の失敗は、CloudWatch ログに記録されます (コネクタでこれらのログを設定している場合)。 DNS サーバーの設定を確認し、コネクタから DNS サーバーへのネットワーク接続を確認します。
コネクタの実行中に VPC DHCP オプションセットの DNS サーバー設定を変更すると、コネクタからの DNS 解決クエリが失敗する可能	DNS 解決クエリの失敗によるコネクタ作成の失敗は、CloudWatch ログに記録されます (コネクタでこれらのログを設定している場合)。

失敗	推奨されるアクション
性があります。DNS 解決に失敗すると、コネクタタスクの一部が失敗状態になる可能性があります。	失敗したタスクは自動的に再開され、コネクタが復旧するはずです。そうならない場合は、サポートに連絡して失敗したコネクタのタスクを再開するか、コネクタを再作成してください。

MSK Connect のセキュリティ

Amazon VPC と Amazon MSK-Connect 互換 APIs 間のトラフィックが Amazon ネットワークから出るのを防ぐために AWS PrivateLink、 を搭載したインターフェイス VPC エンドポイントを使用できます。インターフェイス VPC エンドポイントには、インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続は必要ありません。詳細については、「インターフェイス VPC エンドポイントで Amazon MSK APIs を使用する」を参照してください。

MSK Connect のロギング

MSK Connect は、コネクタのデバッグに使用できるログイベントを書き込むことができます。コネクタを作成するときに、次のログの宛先を 0 個以上指定できます。

- Amazon CloudWatch Logs: MSK Connect がコネクタのログイベントを送信するロググループを指定します。ロググループを作成する方法については、「CloudWatch Logs ユーザーガイド」の「ロググループの作成」を参照してください。
- Amazon S3: MSK Connect がコネクタのログイベントを送信する S3 バケットを指定します。S3 バケットを作成する方法については、「Amazon S3 ユーザーガイド」の「バケットの作成」を参照してください。
- Amazon Data Firehose: MSK Connect がコネクタのログイベントを送信する配信ストリームを指定します。配信ストリームを作成する方法については、「Firehose ユーザーガイド」の「Amazon Data Firehose 配信ストリームの作成」を参照してください。

ロギングの設定について詳しくは、「Amazon CloudWatch Logs ユーザーガイド」の「特定の AWS サービスからのログの記録を有効にする」を参照してください。

MSK Connect は、次のタイプのログイベントを発行します。

レベル	説明
INFO	スタートアップとシャットダウン時の対象となるランタイムイベント。
WARN	エラーではないが、望ましくない、または予期しないランタイムの状況。
FATAL	早期終了の原因となる重大なエラー。
ERROR	致命的ではない予期しない状態とランタイムエラー。

以下は、CloudWatch Logs に送信されるログイベントの例です。

```
[Worker-0bb8afa0b01391c41] [2021-09-06 16:02:54,151] WARN [Producer
 clientId=producer-1] Connection to node 1 (b-1.my-test-cluster.twwhtj.c2.kafka.us-
east-1.amazonaws.com/INTERNAL_IP) could not be established. Broker may not be
 available. (org.apache.kafka.clients.NetworkClient:782)
```

シークレットがコネクタログに表示されないようにする

> **Note**
>
> プラグインで機密設定値をシークレットとして定義していない場合、機密設定値がコネクタログに表示される可能性があります。Kafka Connect は、未定義の設定値を他のプレーンテキスト値と同じように扱います。

プラグインがプロパティをシークレットとして定義する場合、Kafka Connect はコネクタログからプロパティの値を削除します。例えば、以下のコネクタログは、プラグインによって `aws.secret.key` が PASSWORD タイプとして定義されている場合、その値が **[hidden]** に置き換えられることを示しています。

```
   2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b] [2022-01-11
15:18:55,150] INFO SecretsManagerConfigProviderConfig values:
   2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b] aws.access.key =
my_access_key
```

```
    2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b] aws.region = us-east-1
    2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b] aws.secret.key
= [hidden]
    2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b] secret.prefix =
    2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b] secret.ttl.ms = 300000
    2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b]
(com.github.jcustenborder.kafka.config.aws.SecretsManagerConfigProviderConfig:361)
```

シークレットがコネクタログファイルに表示されないようにするには、プラグイン開発者は Kafka Connect の列挙定数 `ConfigDef.Type.PASSWORD` を使用して機密プロパティを定義する必要があります。プロパティがタイプ `ConfigDef.Type.PASSWORD` の場合、値がプレーンテキストとして送信された場合でも、Kafka Connect はその値をコネクタログから除外します。

Amazon MSK Connect のモニタリング

モニタリングは、MSK Connect やその他の AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。Amazon CloudWatch は、 AWS リソースと で実行されるアプリケーションを AWS リアルタイムでモニタリングします。メトリクスを収集および追跡し、カスタマイズされたダッシュボードを作成し、指定されたメトリクスが指定したしきい値に達したときに通知またはアクションを実行するアラームを設定できます。例えば、CloudWatch でコネクタの CPU 使用率やその他のメトリクスを追跡できるため、必要に応じて容量を増やすことができます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

次の API オペレーションを使用できます。

- `DescribeConnectorOperation`: コネクタの更新オペレーションのステータスをモニタリングします。
- `ListConnectorOperations`: コネクタで実行された以前の更新を追跡します。

次の表は、MSK Connect が`ConnectorName` ディメンションで CloudWatch に送信するメトリクスを示しています。MSK Connect は、これらのメトリクスをデフォルトで追加コストなしで提供します。CloudWatch はこれらのメトリクスを 15 か月間保持するため、履歴情報にアクセスして、コネクタのパフォーマンスをより正確に把握できます。また、特定のしきい値を監視するアラームを設定し、それらのしきい値に達したときに通知を送信したり、アクションを実行したりすることもできます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

メトリクス名	説明
CpuUtilization	システムおよびユーザーによる CPU 消費のパーセンテージ。
ErroredTaskCount	エラーが発生したタスクの数。
MemoryUtilization	現在使用中の Java 仮想マシン (JVM) ヒープメモリだけでなく、ワーカーインスタンスの合計メモリに占める割合。JVM は通常、メモリを運用システムに解放しません。そのため、JVM ヒープサイズ (MemoryUtilization) は通常、最小ヒープサイズから始まり、徐々に増加して約 80 ~ 90% の安定した最大値になります。JVM ヒープ使用量は、コネクタの実際のメモリ使用量の変化に合わせて増減する場合があります。
RebalanceCompletedTotal	このコネクタによって完了したリバランスの総数。
RebalanceTimeAvg	コネクタがリバランスに費やした平均時間 (ミリ秒単位)。
RebalanceTimeMax	コネクタがリバランスに費やした最大時間 (ミリ秒単位)。
RebalanceTimeSinceLast	このコネクタが最新のリバランスを完了してからのミリ秒単位の時間。
RunningTaskCount	コネクタで実行中のタスクの数。
SinkConsumerByteRate	変換がデータに適用される前に Kafka Connect フレームワークのシンクコンシューマーが消費した 1 秒あたりの平均バイト数。
SinkRecordReadRate	Apache Kafka または Amazon MSK クラスターから読み取られた 1 秒あたりの平均レコード数。

メトリクス名	説明
SinkRecordSendRate	変換から出力され、宛先に送信される 1 秒あたりの平均レコード数。この数には、フィルタリングされたレコードは含まれていません。
SourceRecordPollRate	生成またはポーリングされた 1 秒あたりの平均レコード数。
SourceProducerByteRate	変換がデータに適用された後に Kafka Connect フレームワークのソースプロデューサーによって 1 秒あたりに生成された平均バイト数。
SourceRecordWriteRate	変換から出力され、Apache Kafka または Amazon MSK クラスターに書き込まれる 1 秒あたりの平均レコード数。
TaskStartupAttemptsTotal	コネクタが試行したタスクの起動の総数。このメトリクスを使用して、タスクのスタートアップの異常を識別できます。
TaskStartupSuccessPercentage	成功したタスクの平均パーセンテージは、コネクタでスタートされます。このメトリクスを使用して、タスクのスタートアップ試行の異常を識別できます。
WorkerCount	コネクタで実行されているワーカーの数。
BytesInPerSec	ワーカー間の通信のために Kafka Connect フレームワークに転送されたメタデータのバイト数。
BytesOutPerSec	ワーカー間の通信のために Kafka Connect フレームワークから転送されたメタデータのバイト数。

Amazon MSK Connect リソースの設定例

このセクションには、一般的なサードパーティー製コネクタや設定プロバイダーなどの Amazon MSK Connect リソースのセットアップに役立つ例が含まれています。

トピック

- Amazon S3 シンクコネクタを設定する
- MSK Connect の EventBridge Kafka シンクコネクタを設定する
- Debezium ソースコネクタ (設定プロバイダー付き) を使用する

Amazon S3 シンクコネクタを設定する

この例では、Confluent Amazon S3 シンクコネクタと を使用して AWS CLI MSK Connect で Amazon S3 シンクコネクタを作成する方法を示します。

1. 次の JSON をコピーして、新しいファイルに貼り付けます。プレースホルダー文字列を、Amazon MSK クラスターのブートストラップサーバー接続文字列とクラスターのサブネットおよびセキュリティグループ ID に対応する値に置き換えます。サービス実行ロールの設定方法については、「the section called “IAM ロールとポリシー”」を参照してください。

```
{
    "connectorConfiguration": {
        "connector.class": "io.confluent.connect.s3.S3SinkConnector",
        "s3.region": "us-east-1",
        "format.class": "io.confluent.connect.s3.format.json.JsonFormat",
        "flush.size": "1",
        "schema.compatibility": "NONE",
        "topics": "my-test-topic",
        "tasks.max": "2",
        "partitioner.class":
 "io.confluent.connect.storage.partitioner.DefaultPartitioner",
        "storage.class": "io.confluent.connect.s3.storage.S3Storage",
        "s3.bucket.name": "amzn-s3-demo-bucket"
    },
    "connectorName": "example-S3-sink-connector",
    "kafkaCluster": {
        "apacheKafkaCluster": {
            "bootstrapServers": "<cluster-bootstrap-servers-string>",
            "vpc": {
                "subnets": [
```

```
                    "<cluster-subnet-1>",
                    "<cluster-subnet-2>",
                    "<cluster-subnet-3>"
                ],
                "securityGroups": ["<cluster-security-group-id>"]
            }
        }
    },
    "capacity": {
        "provisionedCapacity": {
            "mcuCount": 2,
            "workerCount": 4
        }
    },
    "kafkaConnectVersion": "2.7.1",
    "serviceExecutionRoleArn": "<arn-of-a-role-that-msk-connect-can-assume>",
    "plugins": [
        {
            "customPlugin": {
                "customPluginArn": "<arn-of-custom-plugin-that-contains-connector-
code>",
                "revision": 1
            }
        }
    ],
    "kafkaClusterEncryptionInTransit": {"encryptionType": "PLAINTEXT"},
    "kafkaClusterClientAuthentication": {"authenticationType": "NONE"}
}
```

2. 前のステップで JSON ファイルを保存したフォルダで、次の AWS CLI コマンドを実行します。

```
aws kafkaconnect create-connector --cli-input-json file://connector-info.json
```

以下は、コマンドを正常に実行したときに得られる出力の例です。

```
{
    "ConnectorArn": "arn:aws:kafkaconnect:us-east-1:123450006789:connector/example-
S3-sink-connector/abc12345-abcd-4444-a8b9-123456f513ed-2",
    "ConnectorState": "CREATING",
    "ConnectorName": "example-S3-sink-connector"
}
```

MSK Connect の EventBridge Kafka シンクコネクタを設定する

このトピックでは、MSK Connect 用の EventBridge Kafka シンクコネクタを設定する方法について説明します。このコネクタを使用すると、MSK クラスターから EventBridge イベントバスにイベントを送信できます。このトピックでは、必要なリソースを作成し、Kafka と EventBridge 間のシームレスなデータフローを可能にするようにコネクタを設定するプロセスについて説明します。

トピック

- 前提条件
- MSK Connect に必要なリソースを設定する
- コネクタを作成する
- Kafka にメッセージを送信する

前提条件

コネクタをデプロイする前に、次のリソースがあることを確認してください。

- Amazon MSK クラスター: Kafka メッセージを生成および消費するためのアクティブな MSK クラスター。
- Amazon EventBridge イベントバス: Kafka トピックからイベントを受信するための EventBridge イベントバス。
- IAM ロール: MSK Connect と EventBridge コネクタに必要なアクセス許可を持つ IAM ロールを作成します。
- MSK Connect または MSK クラスターの VPC とサブネットで作成された EventBridge の VPC インターフェイスエンドポイントからパブリックインターネットにアクセスします。 EventBridge これにより、NAT ゲートウェイを必要とせずにパブリックインターネットを経由することを回避できます。
- Amazon EC2 インスタンスや などのクライアントマシンは、トピックを作成しAWS CloudShell、Kafka にレコードを送信します。

MSK Connect に必要なリソースを設定する

コネクタの IAM ロールを作成してから、コネクタを作成します。また、EventBridge ルールを作成して、EventBridge イベントバスに送信された Kafka EventBridge イベントをフィルタリングします。

トピック

- コネクタの IAM ロール
- 受信イベントの EventBridge ルール

コネクタの IAM ロール

コネクタに関連付ける IAM ロールには、EventBridge へのイベントの送信を許可する PutEvents アクセス許可が必要です。次の IAM ポリシーの例では、 という名前のイベントバスにイベントを送信する許可を付与しますexample-event-bus。次の例のリソース ARN をイベントバスの ARN に置き換えてください。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/example-event-bus"
    }
  ]
}
```

さらに、コネクタの IAM ロールに次の信頼ポリシーが含まれていることを確認する必要があります。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "kafkaconnect.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

受信イベントの EventBridge ルール

受信イベントをイベントパターンと呼ばれるイベントデータ基準と照合するルールを作成します。イベントパターンを使用すると、受信イベントをフィルタリングする基準を定義し、どのイベントが特定のルールをトリガーし、その後指定されたターゲットにルーティングするかを決定できます。次のイベントパターンの例は、EventBridge イベントバスに送信された Kafka イベントと一致します。

```
{
  "detail": {
    "topic": ["msk-eventbridge-tutorial"]
  }
}
```

Kafka シンクコネクタを使用して Kafka から EventBridge に送信されるイベントの例を次に示します。

```
{
  "version": "0",
  "id": "dbc1c73a-c51d-0c0e-ca61-ab9278974c57",
  "account": "123456789012",
  "time": "2025-03-26T10:15:00Z",
  "region": "us-east-1",
  "detail-type": "msk-eventbridge-tutorial",
  "source": "kafka-connect.msk-eventbridge-tutorial",
  "resources": [],
  "detail": {
    "topic": "msk-eventbridge-tutorial",
    "partition": 0,
    "offset": 0,
    "timestamp": 1742984100000,
    "timestampType": "CreateTime",
    "headers": [],
    "key": "order-1",
    "value": {
```

```
      "orderItems": [
        "item-1",
        "item-2"
      ],
      "orderCreatedTime": "Wed Mar 26 10:15:00 UTC 2025"
    }
  }
}
```

EventBridge コンソールで、このパターン例を使用してイベントバスにルールを作成し、CloudWatch Logs グループなどのターゲットを指定します。EventBridge コンソールは、CloudWatch Logs グループに必要なアクセスポリシーを自動的に設定します。

コネクタを作成する

次のセクションでは、 を使用して EventBridge Kafka シンクコネクタを作成してデプロイします AWS Management Console。

トピック

- ステップ 1: コネクタをダウンロードする
- ステップ 1: Amazon S3 バケットを作成する
- ステップ 3: MSK Connect でプラグインを作成する
- ステップ 4: コネクタを作成する

ステップ 1: コネクタをダウンロードする

EventBridge Kafka コネクタの GitHub リリースページから最新の EventBridge コネクタシンク JAR をダウンロードします。たとえば、バージョン v1.4.1 をダウンロードするには、JAR ファイルリンク を選択してコネクタ`kafka-eventbridge-sink-with-dependencies.jar`をダウンロードします。次に、ファイルをマシン上の任意の場所に保存します。

ステップ 1: Amazon S3 バケットを作成する

1. MSK Connect で使用する JAR ファイルを Amazon S3 に保存するには、 を開き AWS Management Console、Amazon S3 を選択します。
2. Amazon S3 コンソールで、バケットの作成を選択し、一意のバケット名を入力します。例えば、**`amzn-s3-demo-bucket1-eb-connector`**。
3. Amazon S3 バケットに適したリージョンを選択します。MSK クラスターがデプロイされているリージョンと一致することを確認してください。

4. バケット設定では、デフォルトの選択を保持するか、必要に応じて調整します。
5. [バケットを作成] を選択します。
6. JAR ファイルを Amazon S3 バケットにアップロードします。

ステップ 3: MSK Connect でプラグインを作成する

1. を開き AWS Management Console、MSK Connect に移動します。
2. 左側のナビゲーションペインで、カスタムプラグインを選択します。
3. プラグインの作成を選択し、プラグイン名を入力します。例えば、**eventbridge-sink-plugin**。
4. カスタムプラグインの場所の場合は、S3 オブジェクト URL を貼り付けます。
5. プラグインのオプションの説明を追加します。
6. プラグインの作成 を選択します。

プラグインを作成したら、それを使用して MSK Connect で EventBridge Kafka コネクタを設定およびデプロイできます。

ステップ 4: コネクタを作成する

コネクタを作成する前に、コネクタエラーを避けるために必要な Kafka トピックを作成することをお勧めします。トピックを作成するには、クライアントマシンを使用します。

1. MSK コンソールの左側のペインで、コネクタを選択し、コネクタの作成を選択します。
2. プラグインのリストで、eventbridge-sink-plugin を選択し、次へを選択します。
3. コネクタ名には、「」と入力します**EventBridgeSink**。
4. クラスターのリストで、MSK クラスターを選択します。
5. コネクタの次の設定をコピーし、コネクタ設定フィールドに貼り付けます。

   必要に応じて、次の設定のプレースホルダーを置き換えます。

   - MSK クラスターにパブリックインターネットアクセス`aws.eventbridge.endpoint.uri`がある場合は、 を削除します。
   - PrivateLink を使用して MSK から EventBridge に安全に接続する場合は、 の DNS 部分を、前に作成した EventBridge の (オプション) VPC インターフェイスエンドポイントの`https://`正しいプライベート DNS 名に置き換えます。

- 次の設定の EventBridge イベントバス ARN をイベントバスの ARN に置き換えます。
- リージョン固有の値を更新します。

```
{
  "connector.class":
 "software.amazon.event.kafkaconnector.EventBridgeSinkConnector",
  "aws.eventbridge.connector.id": "msk-eventbridge-tutorial",
  "topics": "msk-eventbridge-tutorial",
  "tasks.max": "1",
  "aws.eventbridge.endpoint.uri": "https://events.us-east-1.amazonaws.com",
  "aws.eventbridge.eventbus.arn": "arn:aws:events:us-east-1:123456789012:event-bus/
example-event-bus",
  "value.converter.schemas.enable": "false",
  "value.converter": "org.apache.kafka.connect.json.JsonConverter",
  "aws.eventbridge.region": "us-east-1",
  "auto.offset.reset": "earliest",
  "key.converter": "org.apache.kafka.connect.storage.StringConverter"
}
```

コネクタ設定の詳細については、「eventbridge-kafka-connector」を参照してください。

必要に応じて、ワーカーと自動スケーリングの設定を変更します。また、ドロップダウンから利用可能な最新の (推奨) Apache Kafka Connect バージョンを使用することをお勧めします。アクセス許可で、前に作成したロールを使用します。また、オブザーバビリティとトラブルシューティングのために CloudWatch へのログ記録を有効にすることをお勧めします。必要に応じて、タグなどの他のオプション設定を調整します。次に、コネクタをデプロイし、ステータスが実行中状態になるまで待ちます。

Kafka にメッセージを送信する

Apache Avro や JSON などのメッセージエンコーディングを設定するには、 `value.converter` と、オプションで Kafka Connect で使用できる`key.converter`設定を使用して、さまざまなコンバーターを指定します。

このトピックconnector exampleの は、 `org.apache.kafka.connect.json.JsonConverter`の の使用で示されているように、JSON エンコードされたメッセージで動作するように設定されていますvalue converter。コネクタが実行中状態になったら、クライアントマシンから `msk-eventbridge-tutorial` Kafka トピックにレコードを送信します。

Debezium ソースコネクタ (設定プロバイダー付き) を使用する

この例は、MySQL 互換の Amazon Aurora データベースをソースとして Debezium MySQL コネクタプラグインを使用する方法を示しています。この例では、 AWS Secrets Managerのデータベースの認証情報を外部化するために、オープンソースの AWS Secrets Manager Config プロバイダーも設定しています。設定プロバイダーの詳細については、「チュートリアル: 設定プロバイダーを用いた機密情報の外部化」を参照してください。

> **Important**
>
> Debezium MySQL コネクタプラグインは 1 つのタスクのみをサポートし、Amazon MSK Connect の自動スケーリングキャパシティモードでは動作しません。代わりにプロビジョニングキャパシティモードを使用し、`workerCount` をコネクタ設定の値と等しい値に設定してください。MSK Connect のキャパシティモードの詳細については、「コネクタ容量を理解する」を参照してください。

Debezium ソースコネクタを使用するための完全な前提条件

コネクタは、 の外部 AWS Secrets Manager にある などのサービスとやり取りできるように、インターネットにアクセスできる必要があります Amazon Virtual Private Cloud。このセクションの手順は、インターネットアクセスを有効にするための次のタスクを実行するのに役立ちます。

- NAT ゲートウェイをホストし、VPC 内のインターネットゲートウェイにトラフィックをルーティングするパブリックサブネットを設定します。
- プライベートサブネットのトラフィックを NAT ゲートウェイに送るデフォルトルートを作成します。

詳細については、「Amazon MSK Connect のインターネットアクセスを有効にする」を参照してください。

前提条件

インターネットアクセスを有効にするには、以下のものが必要です。

- クラスターに関連付けられた Amazon Virtual Private Cloud (VPC) の ID。例えば、vpc-123456ab などです。

- VPC 内のプライベートサブネットの ID。例えば、subnet-a1b2c3de、subnet-f4g5h6ij などです。コネクタにはプライベートサブネットを設定する必要があります。

コネクタのインターネットアクセスを有効にするには

1. https://console.aws.amazon.com/vpc/ で Amazon Virtual Private Cloud コンソールを開きます。
2. わかりやすい名前を付けて NAT ゲートウェイのパブリックサブネットを作成し、サブネット ID を書き留めます。詳細な手順については、「VPC にサブネットを作成する」を参照してください。
3. VPC がインターネットと通信できるようにインターネットゲートウェイを作成し、ゲートウェイ ID を書き留めます。VPC にインターネットゲートウェイをアタッチします。手順については、「インターネットゲートウェイの作成とアタッチ」を参照してください。
4. プライベートサブネット内のホストがパブリックサブネットにアクセスできるように、パブリック NAT ゲートウェイをプロビジョニングします。NAT ゲートウェイを作成するときに、前に作成したパブリックサブネットを選択します。手順については、「NAT ゲートウェイの作成」を参照してください。
5. ルートテーブルを設定します。この設定を完了するには、合計で 2 つのルートテーブルが必要です。VPC と同時に自動的に作成されたメインのルートテーブルが既にあるはずです。このステップでは、パブリックサブネット用の追加のルートテーブルを作成します。

   a. 次の設定を使用して VPC のメインルートテーブルを変更し、プライベートサブネットがトラフィックを NAT ゲートウェイにルーティングするようにします。手順については、「Amazon Virtual Private Cloudユーザーガイド」の「ルートテーブルの使用」を参照してください。

      プライベート MSKC ルートテーブル

プロパティ	値
名前タグ	このルートテーブルには、識別しやすいようにわかりやすい名前タグを付けることをお勧めします。例えば、Private MSKC などです。
関連付けられたサブネット	プライベートサブネット
MSK Connect のインターネットアクセスを有効にするためのルート	• 送信先: 0.0.0.0/0

プロパティ	値
	• ターゲット: NAT ゲートウェイ ID。例えば、nat-12a345bc6789efg1h です。
内部トラフィックのルート	• 送信先: 10.0.0.0/16 この値は VPC の CIDR ブロックによって異なる場合があります。 • ターゲット: ローカル

b. 「カスタムルートテーブルを作成する」の手順に従って、パブリックサブネットのルートテーブルを作成します。テーブルを作成するときは、そのテーブルがどのサブネットに関連付けられているかを識別しやすいように、[名前タグ] フィールドにわかりやすい名前を入力します。例えば、パブリック MSKC と入力します。

c. 以下の設定を使用してパブリック MSKC のルートテーブルを設定します。

プロパティ	値
名前タグ	パブリック MSKC または選択した別のわかりやすい名前を使用してください。
関連付けられたサブネット	NAT ゲートウェイを使用するパブリックサブネット
MSK Connect のインターネットアクセスを有効にするためのルート	• 送信先: 0.0.0.0/0 • ターゲット: インターネットゲートウェイ ID。例えば igw-1a234bc5 です。
内部トラフィックのルート	• 送信先: 10.0.0.0/16 この値は VPC の CIDR ブロックによって異なる場合があります。 • ターゲット: ローカル

Amazon MSK Connect のインターネットアクセスが有効になり、コネクタを作成する準備が整いました。

Debezium ソースコネクタを作成する

この手順では、Debezium ソースコネクタを作成する方法について説明します。

1. カスタムプラグインを作成する

   a. Debezium サイトから最新の安定版リリース用の MySQL コネクタプラグインをダウンロードしてください。ダウンロードした Debezium リリースバージョン (バージョン 2.x、または古いシリーズ 1.x) を書き留めます。この手順の後半で、Debezium のバージョンに基づいてコネクタを作成します。

   b. AWS Secrets Manager 設定プロバイダーをダウンロードして解凍します。

   c. 以下のアーカイブを同じディレクトリに置きます。

      - debezium-connector-mysql フォルダ
      - jcusten-border-kafka-config-provider-aws-0.1.1 フォルダ

   d. 前のステップで作成したディレクトリを ZIP ファイルに圧縮し、その ZIP ファイルを S3 バケットにアップロードします。手順については、「Amazon S3 ユーザーガイド」の「オブジェクトのアップロード」を参照してください。

   e. 次の JSON をコピーして、ファイルに貼り付けます。例えば、debezium-source-custom-plugin.json。*<example-custom-plugin-name>* をプラグインに含める名前に、*#amzn-s3-demo-bucket-arn>* を ZIP ファイルをアップロードした Amazon S3 バケットの ARN に、 を S3 にアップロードした ZIP オブジェクトのファイルキー*<file-key-of-ZIP-object>*に置き換えます。

```
{
    "name": "<example-custom-plugin-name>",
    "contentType": "ZIP",
    "location": {
        "s3Location": {
            "bucketArn": "<amzn-s3-demo-bucket-arn>",
            "fileKey": "<file-key-of-ZIP-object>"
        }
    }
}
```

   f. JSON ファイルを保存したフォルダから次の AWS CLI コマンドを実行して、プラグインを作成します。

```
aws kafkaconnect create-custom-plugin --cli-input-json file://<debezium-source-custom-plugin.json>
```

以下のような出力が表示されます。

```
{
    "CustomPluginArn": "arn:aws:kafkaconnect:us-east-1:012345678901:custom-plugin/example-custom-plugin-name/abcd1234-a0b0-1234-c1-12345678abcd-1",
    "CustomPluginState": "CREATING",
    "Name": "example-custom-plugin-name",
    "Revision": 1
}
```

g. 次のコマンドを実行して、プラグインの状態を確認します。状態は CREATING から ACTIVE に変わります。ARN プレースホルダーを前のコマンドの出力で取得した ARN に置き換えます。

```
aws kafkaconnect describe-custom-plugin --custom-plugin-arn "<arn-of-your-custom-plugin>"
```

2. データベース認証情報のシークレットを設定 AWS Secrets Manager および作成する

a. Secrets Manager のコンソール (https://console.aws.amazon.com/secretsmanager/) を開きます。

b. データベースのサインイン認証情報を保存する新しいシークレットを作成します。手順については、「AWS Secrets Managerユーザーガイド」の「シークレットを作成する」を参照してください。

c. シークレットの ARN をコピーします。

d. 以下のサンプルポリシーの Secrets Manager のアクセス許可を サービス実行ロールを理解する に追加します。*<arn:aws:secretsmanager:us-east-1:123456789000:secret:MySecret-1234>* をシークレットの ARN で置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetResourcePolicy",
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret",
        "secretsmanager:ListSecretVersionIds"
      ],
      "Resource": [
      "arn:aws:secretsmanager:us-
east-1:123456789012:secret:MySecret-1234"
      ]
    }
  ]
}
```

IAM のアクセス許可を追加する手順については、「IAM ユーザーガイド」の「IAM ID のアクセス許可の追加と削除」を参照してください。

3. 設定プロバイダーに関する情報を使用してカスタムワーカー設定を作成します。

   a. 次のワーカー設定プロパティをファイルにコピーして、プレースホルダー文字列をシナリオに対応する値に置き換えます。 AWS Secrets Manager 設定プロバイダーの設定プロパティの詳細については、プラグインのドキュメントの「SecretsManagerConfigProvider」を参照してください。

```
key.converter=<org.apache.kafka.connect.storage.StringConverter>
value.converter=<org.apache.kafka.connect.storage.StringConverter>
config.providers.secretManager.class=com.github.jcustenborder.kafka.config.aws.SecretsM
config.providers=secretManager
config.providers.secretManager.param.aws.region=<us-east-1>
```

   b. 次の AWS CLI コマンドを実行して、カスタムワーカー設定を作成します。

      以下の値を置き換えます:

      - *<my-worker-config-name>* - カスタムワーカー設定のわかりやすい名前
      - *<encoded-properties-file-content-string>* - 前のステップでコピーしたプレーンテキストプロパティの base64 でエンコードされたバージョン

```
aws kafkaconnect create-worker-configuration --name <my-worker-config-name> --
properties-file-content <encoded-properties-file-content-string>
```

4. コネクタを作成する

   a. Debezium のバージョン (2.x または 1.x) に対応する次の JSON をコピーして、新しいファイルに貼り付けます。*<placeholder>* 文字列をシナリオに対応する値に置き換えます。サービス実行ロールの設定方法については、「the section called “IAM ロールとポリシー”」を参照してください。

   この設定では、データベースの認証情報を指定するのにプレーンテキストではなく ${secretManager:MySecret-1234:dbusername} のような変数を使用していることに注意してください。*MySecret-1234* をシークレットの名前に置き換えてから、取得したいキーの名前を入力します。また、*<arn-of-config-provider-worker-configuration>* をカスタムワーカー設定の ARN に置き換える必要があります。

   Debezium 2.x

   Debezium 2.x バージョンでは、次の JSON をコピーして、新しいファイルに貼り付けます。*<placeholder>* 文字列をシナリオに対応する値に置き換えます。

```
{
 "connectorConfiguration": {
  "connector.class": "io.debezium.connector.mysql.MySqlConnector",
  "tasks.max": "1",
  "database.hostname": "<aurora-database-writer-instance-endpoint>",
  "database.port": "3306",
  "database.user": "<${secretManager:MySecret-1234:dbusername}>",
  "database.password": "<${secretManager:MySecret-1234:dbpassword}>",
  "database.server.id": "123456",
  "database.include.list": "<list-of-databases-hosted-by-specified-server>",
  "topic.prefix": "<logical-name-of-database-server>",
  "schema.history.internal.kafka.topic": "<kafka-topic-used-by-debezium-to-
track-schema-changes>",
  "schema.history.internal.kafka.bootstrap.servers": "<cluster-bootstrap-
servers-string>",
  "schema.history.internal.consumer.security.protocol": "SASL_SSL",
  "schema.history.internal.consumer.sasl.mechanism": "AWS_MSK_IAM",
  "schema.history.internal.consumer.sasl.jaas.config":
 "software.amazon.msk.auth.iam.IAMLoginModule required;",
```

```
  "schema.history.internal.consumer.sasl.client.callback.handler.class":
 "software.amazon.msk.auth.iam.IAMClientCallbackHandler",
  "schema.history.internal.producer.security.protocol": "SASL_SSL",
  "schema.history.internal.producer.sasl.mechanism": "AWS_MSK_IAM",
  "schema.history.internal.producer.sasl.jaas.config":
 "software.amazon.msk.auth.iam.IAMLoginModule required;",
  "schema.history.internal.producer.sasl.client.callback.handler.class":
 "software.amazon.msk.auth.iam.IAMClientCallbackHandler",
  "include.schema.changes": "true"
 },
 "connectorName": "example-Debezium-source-connector",
 "kafkaCluster": {
  "apacheKafkaCluster": {
   "bootstrapServers": "<cluster-bootstrap-servers-string>",
   "vpc": {
    "subnets": [
     "<cluster-subnet-1>",
     "<cluster-subnet-2>",
     "<cluster-subnet-3>"
    ],
    "securityGroups": ["<id-of-cluster-security-group>"]
   }
  }
 },
 "capacity": {
  "provisionedCapacity": {
   "mcuCount": 2,
   "workerCount": 1
  }
 },
 "kafkaConnectVersion": "2.7.1",
 "serviceExecutionRoleArn": "<arn-of-service-execution-role-that-msk-
connect-can-assume>",
 "plugins": [{
  "customPlugin": {
   "customPluginArn": "<arn-of-msk-connect-plugin-that-contains-connector-
code>",
   "revision": 1
  }
 }],
 "kafkaClusterEncryptionInTransit": {
  "encryptionType": "TLS"
 },
 "kafkaClusterClientAuthentication": {
```

```
  "authenticationType": "IAM"
 },
 "workerConfiguration": {
  "workerConfigurationArn": "<arn-of-config-provider-worker-configuration>",
  "revision": 1
 }
}
```

Debezium 1.x

Debezium 1.x バージョンでは、次の JSON をコピーして、新しいファイルに貼り付けます。*<placeholder>* 文字列をシナリオに対応する値に置き換えます。

```
{
 "connectorConfiguration": {
  "connector.class": "io.debezium.connector.mysql.MySqlConnector",
  "tasks.max": "1",
  "database.hostname": "<aurora-database-writer-instance-endpoint>",
  "database.port": "3306",
  "database.user": "<${secretManager:MySecret-1234:dbusername}>",
  "database.password": "<${secretManager:MySecret-1234:dbpassword}>",
  "database.server.id": "123456",
  "database.server.name": "<logical-name-of-database-server>",
  "database.include.list": "<list-of-databases-hosted-by-specified-server>",
  "database.history.kafka.topic": "<kafka-topic-used-by-debezium-to-track-
schema-changes>",
  "database.history.kafka.bootstrap.servers": "<cluster-bootstrap-servers-
string>",
  "database.history.consumer.security.protocol": "SASL_SSL",
  "database.history.consumer.sasl.mechanism": "AWS_MSK_IAM",
  "database.history.consumer.sasl.jaas.config":
 "software.amazon.msk.auth.iam.IAMLoginModule required;",
  "database.history.consumer.sasl.client.callback.handler.class":
 "software.amazon.msk.auth.iam.IAMClientCallbackHandler",
  "database.history.producer.security.protocol": "SASL_SSL",
  "database.history.producer.sasl.mechanism": "AWS_MSK_IAM",
  "database.history.producer.sasl.jaas.config":
 "software.amazon.msk.auth.iam.IAMLoginModule required;",
  "database.history.producer.sasl.client.callback.handler.class":
 "software.amazon.msk.auth.iam.IAMClientCallbackHandler",
  "include.schema.changes": "true"
 },
 "connectorName": "example-Debezium-source-connector",
```

```
 "kafkaCluster": {
  "apacheKafkaCluster": {
   "bootstrapServers": "<cluster-bootstrap-servers-string>",
   "vpc": {
    "subnets": [
     "<cluster-subnet-1>",
     "<cluster-subnet-2>",
     "<cluster-subnet-3>"
    ],
    "securityGroups": ["<id-of-cluster-security-group>"]
   }
  }
 },
 "capacity": {
  "provisionedCapacity": {
   "mcuCount": 2,
   "workerCount": 1
  }
 },
 "kafkaConnectVersion": "2.7.1",
 "serviceExecutionRoleArn": "<arn-of-service-execution-role-that-msk-
connect-can-assume>",
 "plugins": [{
  "customPlugin": {
   "customPluginArn": "<arn-of-msk-connect-plugin-that-contains-connector-
code>",
   "revision": 1
  }
 }],
 "kafkaClusterEncryptionInTransit": {
  "encryptionType": "TLS"
 },
 "kafkaClusterClientAuthentication": {
  "authenticationType": "IAM"
 },
 "workerConfiguration": {
  "workerConfigurationArn": "<arn-of-config-provider-worker-configuration>",
  "revision": 1
 }
}
```

b. 前のステップで JSON ファイルを保存したフォルダで、次の AWS CLI コマンドを実行します。

```
aws kafkaconnect create-connector --cli-input-json file://connector-info.json
```

以下は、コマンドを正常に実行したときに得られる出力の例です。

```
{
    "ConnectorArn": "arn:aws:kafkaconnect:us-east-1:123450006789:connector/
example-Debezium-source-connector/abc12345-abcd-4444-a8b9-123456f513ed-2",
    "ConnectorState": "CREATING",
    "ConnectorName": "example-Debezium-source-connector"
}
```

Debezium コネクタ設定を更新する

Debezium コネクタの設定を更新するには、次の手順に従います。

1. 次の JSON をコピーし、新しいファイルに貼り付けます。<placeholder> 文字列をシナリオに対応する値に置き換えます。

```
{
   "connectorArn": <connector_arn>,
   "connectorConfiguration": <new_configuration_in_json>,
   "currentVersion": <current_version>
}
```

2. 前のステップで JSON ファイルを保存したフォルダで、次の AWS CLI コマンドを実行します。

```
aws kafkaconnect update-connector --cli-input-json file://connector-info.json
```

コマンドを正常に実行したときの出力の例を次に示します。

```
{
    "connectorArn": "arn:aws:kafkaconnect:us-east-1:123450006789:connector/example-
Debezium-source-connector/abc12345-abcd-4444-a8b9-123456f513ed-2",
    "connectorOperationArn": "arn:aws:kafkaconnect:us-
east-1:123450006789:connector-operation/example-Debezium-source-connector/abc12345-
abcd-4444-a8b9-123456f513ed-2/41b6ad56-3184-479b-850a-a8bedd5a02f3",
    "connectorState": "UPDATING"
}
```

3. 次のコマンドを実行して、オペレーションの現在の状態をモニタリングできるようになりました。

```
aws kafkaconnect describe-connector-operation --connector-operation-arn
 <operation_arn>
```

詳細なステップを含む Debezium コネクタの例については、「Amazon MSK Connect の紹介 – マネージドコネクタを使用した Apache Kafka クラスターとの間のデータのストリーミング」を参照してください。

Amazon MSK Connect への移行

このセクションでは、Apache Kafka コネクタアプリケーションを Amazon Managed Streaming for Apache Kafka Connect (Amazon MSK Connect) に移行する方法について説明します。Amazon MSK Connect に移行する利点の詳細については、「???」を参照してください。

このセクションでは、Kafka Connect と Amazon MSK Connect で使用される状態管理のトピックだけでなく、ソースコネクタとシンクコネクタの移行手順についても説明します。

Kafka Connect で使用される内部トピックを理解する

分散モードで実行されている Apache Kafka Connect アプリケーションは、Kafka クラスターとグループメンバーの内部トピックを使用して、その状態を保存します。以下は、Kafka Connect アプリケーションで使用される内部トピックに対応する設定値です。

- `config.storage.topic` で指定された設定トピック

  Kafka Connect は、ユーザーが開始したすべてのコネクタとタスクの設定を設定トピックに保存します。ユーザーがコネクタの設定を更新するたびに、またはコネクタが再設定の要求を送信するたびに (コネクタがより多くのタスクを開始できることを検出した場合など)、このトピックにレコードが出力されます。このトピックは圧縮が有効になっているため、常に各エンティティの最後の状態を維持します。

- `offset.storage.topic` で指定されたオフセットトピック

  Kafka Connect は、ソースコネクタのオフセットをオフセットトピックに保存します。設定トピックと同様に、オフセットトピックは圧縮が有効になっています。このトピックは、外部システムから Kafka にデータを生成するソースコネクタのソース位置のみを記述するために使用されま

す。Kafka からデータを読み込んで外部システムに送信するシンクコネクタは、通常の Kafka コンシューマーグループを使用してコンシューマーオフセットを保存します。

- `status.storage.topic` で指定されたステータストピック

  Kafka Connect は、コネクタとタスクの現在の状態をステータストピックに保存します。このトピックは、REST API のユーザーがクエリを実行するデータを一元化する場所として使用されます。このトピックを使用すると、ユーザーは任意のワーカーにクエリを実行し、実行中のすべてのプラグインのステータスを取得できます。設定トピックやオフセットのトピックと同様に、ステータストピックも圧縮が有効になっています。

これらのトピックに加えて、Kafka Connect は Kafka のグループメンバー API を幅広く活用しています。グループはコネクタ名にちなんで命名されます。例えば、file-sink という名前のコネクタの場合、グループは connect-file-sink という名前になります。グループ内の各コンシューマーは、単一のタスクにレコードを提供します。これらのグループとそのオフセットは、`Kafka-consumer-group.sh` などの通常のコンシューマーグループのツールを使用して取得できます。各シンクコネクタに対して、Connect ランタイムは Kafka からレコードを抽出する通常のコンシューマーグループを実行します。

Amazon MSK Connect アプリケーションの状態管理

デフォルトでは、Amazon MSK Connect は、Amazon MSK コネクタごとに Kafka クラスターに 3 つの別個のトピックを作成し、コネクタの設定、オフセット、およびステータスを保存します。デフォルトのトピック名は、次のような構造になっています。

- __msk_connect_configs_*connector-name_connector-id*
- __msk_connect_status_*connector-name_connector-id*
- __msk_connect_offsets_*connector-name_connector-id*

> **Note**
>
> ソースコネクタ間のオフセットの連続性を提供するために、デフォルトトピックの代わりに任意のオフセットストレージトピックを使用できます。オフセットストレージトピックを指定すると、前のコネクタの最後のオフセットから読み取りを再開するソースコネクタを作成するといったタスクを実行しやすくなります。オフセットストレージトピックを指定するには、コネクタを作成する前に Amazon MSK Connect ワーカー設定で offset.storage.topic プロパティの値を指定します。

ソースコネクタを Amazon MSK Connect に移行する

ソースコネクタは、外部システムから Kafka にレコードをインポートする Apache Kafka Connect アプリケーションです。このセクションでは、 で実行されているオンプレミスまたはセルフマネージド Kafka Connect クラスターを実行している Apache Kafka Connect ソースコネクタアプリケーションを AWS Amazon MSK Connect に移行するプロセスについて説明します。

Kafka Connect ソースコネクタアプリケーションは、設定プロパティ `offset.storage.topic` に設定された値で名前が付けられたトピックにオフセットを保存します。以下は、`movies` と `shows` という 2 つの異なるテーブルからデータをインポートする 2 つのタスクを実行している JDBC コネクタのオフセットメッセージの例です。テーブル movies からインポートされた最新の行のプライマリ ID は 18343 です。テーブル shows からインポートされた最新の行のプライマリ ID は 732 です。

```
["jdbcsource",{"protocol":"1","table":"sample.movies"}] {"incrementing":18343}
["jdbcsource",{"protocol":"1","table":"sample.shows"}] {"incrementing":732}
```

ソースコネクタを Amazon MSK Connect に移行するには、次の手順を実行します。

1. オンプレミスまたはセルフマネージド型の Kafka Connect クラスターからコネクタライブラリをプルして、Amazon MSK Connect カスタムプラグインを作成します。
2. Amazon MSK Connect ワーカープロパティを作成し、プロパティ `key.converter`、`value.converter`、および `offset.storage.topic` を、既存の Kafka Connect クラスターで実行されている Kafka コネクタに設定された値と同じ値に設定します。
3. 既存の Kafka Connect クラスターに PUT `/connectors/`*`connector-name`*`/pause` リクエストを送信して、既存のクラスターのコネクタアプリケーションを一時停止します。
4. コネクタアプリケーションのすべてのタスクが完全に停止していることを確認します。タスクを停止するには、既存の Kafka Connect クラスターに GET `/connectors/`*`connector-name`*`/status` リクエストを送信するか、プロパティ `status.storage.topic` に設定されているトピック名からのメッセージを消費します。
5. 既存のクラスターからコネクタ設定を取得します。コネクタ設定を取得するには、既存のクラスターに GET `/connectors/`*`connector-name`*`/config/` リクエストを送信するか、プロパティ `config.storage.topic` に設定されているトピック名からのメッセージを消費します。
6. 既存のクラスターと同じ名前の新しい Amazon MSK コネクタを作成します。ステップ 1 で作成したコネクタカスタムプラグイン、ステップ 2 で作成したワーカープロパティ、およびステップ 5 で抽出したコネクタ設定を使用して、このコネクタを作成します。

7. Amazon MSK コネクタのステータスが `active` の場合、ログを表示して、コネクタがソースシステムからデータのインポートを開始したことを確認します。
8. `DELETE /connectors/connector-name` リクエストを送信して、既存のクラスター内のコネクタを削除します。

シンクコネクタを Amazon MSK Connect に移行する

シンクコネクタは、Kafka から外部システムにデータをエクスポートする Apache Kafka Connect アプリケーションです。このセクションでは、オンプレミスで実行されている Apache Kafka Connect シンクコネクタアプリケーション、または で実行されているセルフマネージド Kafka Connect クラスター AWS を Amazon MSK Connect に移行するプロセスについて説明します。

Kafka Connect シンクコネクタは Kafka グループメンバー API を使用し、一般的なコンシューマーアプリケーションと同じ `__consumer_offset` トピックにオフセットを保存します。この動作により、シンクコネクタのセルフマネージド型クラスターから Amazon MSK Connect への移行が簡素化されます。

シンクコネクタを Amazon MSK Connect に移行するには、次の手順を実行します。

1. オンプレミスまたはセルフマネージド型の Kafka Connect クラスターからコネクタライブラリをプルして、Amazon MSK Connect カスタムプラグインを作成します。
2. Amazon MSK Connect ワーカープロパティを作成し、プロパティ `key.converter` と `value.converter` を、既存の Kafka Connect クラスターで実行されている Kafka コネクタに設定された値と同じ値に設定します。
3. 既存の Kafka Connect クラスターに `PUT /connectors/connector-name/pause` リクエストを送信して、既存のクラスターでコネクタアプリケーションを一時停止します。
4. コネクタアプリケーションのすべてのタスクが完全に停止していることを確認します。タスクを停止するには、既存の Kafka Connect クラスターに `GET /connectors/connector-name/status` リクエストを送信するか、プロパティ `status.storage.topic` に設定されているトピック名からのメッセージを消費します。
5. 既存のクラスターからコネクタ設定を取得します。コネクタ設定を取得するには、既存のクラスターに `GET /connectors/connector-name/config` リクエストを送信するか、プロパティ `config.storage.topic` に設定されているトピック名からのメッセージを消費します。
6. 既存のクラスターと同じ名前の新しい Amazon MSK コネクタを作成します。ステップ 1 で作成したコネクタカスタムプラグイン、ステップ 2 で作成したワーカープロパティ、およびステップ 5 で抽出したコネクタ設定を使用して、このコネクタを作成します。

7. Amazon MSK コネクタのステータスが `active` の場合、ログを表示して、コネクタがソースシステムからデータのインポートを開始したことを確認します。
8. `DELETE /connectors/connector-name` リクエストを送信して、既存のクラスター内のコネクタを削除します。

Amazon MSK Connect の問題のトラブルシューティング

次の情報は、MSK Connect の使用時に発生する問題を解決するために役立ちます。AWS re:Post に問題を投稿することもできます。

パブリックインターネット上でホストされているリソースにコネクタがアクセスできない

「Amazon MSK Connect のインターネットアクセスを有効にする」を参照してください。

コネクタで実行中のタスクの数が tasks.max で指定されたタスクの数と等しくない

コネクタが使用するタスクの数が指定された tasks.max 設定よりも少ない理由は次のとおりです。

- コネクタの実装によっては、使用できるタスクの数が制限されている場合があります。例えば、MySQL 用の Debezium コネクタは 1 つのタスクしか使用できません。
- 自動スケーリングキャパシティモードを使用する場合、Amazon MSK Connect は、コネクタで実行されているワーカーの数とワーカーあたりの MCU の数に比例する値でコネクタの tasks.max プロパティを上書きします。
- シンクコネクタの場合、並列処理レベル (タスクの数) はトピックパーティションの数を超えることはできません。tasks.max をそれより大きい値に設定することはできますが、1 つのパーティションが同時に複数のタスクによって処理されることはありません。
- Kafka Connect 2.7.x では、デフォルトのコンシューマーパーティションのアサイナーは `RangeAssignor` です。このアサイナーの動作は、各トピックの最初のパーティションを単一のコンシューマーに、各トピックの 2 番目のパーティションを単一のコンシューマーに割り当てるというものです。つまり、`RangeAssignor` を使用するシンクコネクタのアクティブなタスクの最大数は、消費されている単一のトピックに含まれるパーティションの最大数と同じになります。これがユースケースに適さない場合は、`consumer.partition.assignment.strategy` プロパティをより適切なコンシューマーパーティションのアサイナーに設定したワーカー設定を作成する必要があります。「Kafka 2.7 Interface ConsumerPartitionAssignor: All Known Implementing Classes」を参照してください。

Amazon MSK Replicator とは

Amazon MSK Replicator は、異なるまたは同じ Amazon MSK クラスター間でデータを確実にレプリケートできるようにする Amazon MSK 機能です AWS リージョン。ただし、ソースクラスターとターゲットクラスターの両方が同じ に存在する必要があります AWS アカウント。MSK レプリケーターを使用すると、リージョンの障害に耐性のあるストリーミングアプリケーションを簡単に構築することができ、可用性と事業継続性を向上させることができます。MSK レプリケーターには、MSK クラスター間で自動非同期レプリケーションを行う機能が備わっているため、カスタムコードの作成、インフラストラクチャの管理、クロスリージョンネットワークの設定が不要になります。

MSK レプリケーターは、基盤となるリソースを自動的にスケーリングするため、容量のモニタリングやスケーリングを行わなくてもオンデマンドでデータをレプリケートできます。MSK レプリケーターは、トピック設定、アクセスコントロールリスト (ACL)、コンシューマーグループのオフセットなど、必要な Kafka メタデータもレプリケートします。リージョンで予期しないイベントが発生した場合は、他の AWS リージョンにフェイルオーバーして、シームレスに処理を再開できます。

MSK レプリケーターは、クロスリージョンレプリケーション (CRR) と同一リージョンレプリケーション (SRR) の両方をサポートします。クロスリージョンレプリケーションでは、ソース MSK クラスターとターゲット MSK クラスターは異なる AWS リージョンにあります。同じリージョンのレプリケーションでは、ソース MSK クラスターとターゲット MSK クラスターの両方が同じ AWS リージョンにあります。ソースとターゲットの MSK クラスターは、MSK レプリケーターで使用する前に作成する必要があります。

> **Note**
>
> MSK レプリケーターは、次の AWS リージョンをサポートしています。 米国東部 (us-east-1、 バージニア北部)、 米国東部 (us-east-2、 オハイオ)。 米国西部 (us-west-2、 オレゴン)。 欧州 (eu-west-1、 アイルランド)。 欧州 (eu-central-1、 フランクフルト)。 アジアパシフィック (ap-southeast-1、 シンガポール)。 アジアパシフィック (ap-southeast-2、 シドニー)、 欧州 (eu-north-1、 ストックホルム)、 アジアパシフィック (ap-south-1、 ムンバイ)、 欧州 (eu-west-3、 パリ)、 南米 (sa-east-1、 サンパウロ)、 アジアパシフィック (ap-northeast-2、 ソウル)、 欧州 (eu-west-2、 ロンドン)、 アジアパシフィック (ap-northeast-1、 東京)、 米国西部 (us-west-1、 N. California)、 カナダ (ca-central-1、 中央)。

Amazon MSK Replicator の一般的な使用法をいくつかご紹介します。

- マルチリージョンストリーミングアプリケーションの構築: 高い可用性と耐障害性を備えたストリーミングアプリケーションを構築し、カスタムソリューションの設定を必要とせずに、耐障害性の向上を実現します。
- 低レイテンシーのデータアクセス: さまざまな地理的リージョンのコンシューマーに低レイテンシーのデータアクセスを提供します。
- パートナーへのデータ配布: 1 つの Apache Kafka クラスターから多数の Apache Kafka クラスターにデータをコピーすることで、さまざまなチームやパートナーがそれぞれのデータコピーを持てるようにします。
- 分析用データの集約: 複数の Apache Kafka クラスターから 1 つのクラスターにデータをコピーすることで、集約されたリアルタイムデータに関するインサイトを簡単に生成できます。
- ローカルに書き込み、データにグローバルにアクセスする: マルチアクティブレプリケーションを設定して、1 つの AWS リージョンで実行された書き込みを他のリージョンに自動的に伝達し、低レイテンシーと低コストでデータを提供します。

Amazon MSK Replicator の仕組み

MSK レプリケーターの使用を開始するには、ターゲットクラスターの AWS リージョンに新しいレプリケーターを作成する必要があります。MSK レプリケーターは、ソースと呼ばれるプライマリ AWS リージョンのクラスターからターゲットと呼ばれる送信先リージョンのクラスターに、すべてのデータを自動的にコピーします。ソースクラスターとターゲットクラスターは、同じリージョンでも異なる AWS リージョンでもかまいません。ターゲットクラスターがまだ存在しない場合は、作成する必要があります。

レプリケーターを作成すると、MSK レプリケーターはターゲットクラスターの AWS リージョンに必要なすべてのリソースをデプロイして、データレプリケーションのレイテンシーを最適化します。レプリケーションレイテンシーは、MSK クラスターの AWS リージョン間のネットワーク距離、ソースクラスターとターゲットクラスターのスループットキャパシティ、ソースクラスターとターゲットクラスターのパーティション数など、多くの要因によって異なります。MSK レプリケーターは、基盤となるリソースを自動的にスケーリングするため、容量のモニタリングやスケーリングを行わなくてもオンデマンドでデータをレプリケートできます。

データレプリケーション

デフォルトでは、MSK Replicator は、すべてのデータを、ソースクラスターのトピックパーティション内の最新のオフセットからターゲットクラスターに非同期にコピーします。[新しいトピックを検出してコピーする] 設定がオンになっている場合、MSK Replicator は新しいトピックまたはト

ピックパーティションを自動的に検出してターゲットクラスターにコピーします。ただし、レプリケーターがターゲットクラスターの新しいトピックまたはトピックパーティションを検出して作成するまでに、最大 30 秒かかる場合があります。ターゲットクラスターでトピックが作成される前にソーストピックに生成されたメッセージはレプリケートされません。あるいは、トピックの既存のメッセージをターゲットクラスターにレプリケートする場合は、ソースクラスターのトピックパーティション内の最も古いオフセットからレプリケーションを開始するように、作成時にレプリケーターを設定することができます。

MSK Replicator はデータを保存しません。データはソースクラスターから消費され、インメモリでバッファリングされ、ターゲットクラスターに書き込まれます。バッファは、データが正常に書き込まれるか、あるいは再試行後に失敗すると自動的にクリアされます。MSK Replicator とクラスター間のすべての通信とデータは、転送中では常に暗号化されます。`DescribeClusterV2`、 などのすべての MSK レプリケーター API コールDescribeTopicDynamicConfigurationは`CreateTopic`、 AWS CloudTrail にキャプチャされます。MSK ブローカーログにも同じ内容が反映されます。

MSK Replicator は、レプリケーション係数が 3 のターゲットクラスターにトピックを作成します。必要に応じて、ターゲットクラスターのレプリケーション係数を直接変更できます。

メタデータレプリケーション

MSK Replicator は、ソースクラスターからターゲットクラスターへのメタデータのコピーもサポートしています。メタデータにはトピック設定、アクセスコントロールリスト (ACL)、およびコンシューマーグループのオフセットが含まれます。データレプリケーションと同様に、メタデータレプリケーションも非同期的に行われます。パフォーマンスを向上させるために、MSK Replicator はメタデータレプリケーションよりもデータレプリケーションを優先します。

次の表は、MSK Replicator がコピーできるアクセスコントロールリスト (ACL) の一覧です。

Operation	リソース	許可される API
変更	トピック	CreatePartitions
AlterConfigs	トピック	AlterConfigs
作成	トピック	CreateTopics、Metadata
削除	トピック	DeleteRecords、DeleteTopics

Operation	リソース	許可される API
説明	トピック	ListOffsets、Metadata、OffsetFetch、OffsetForLeaderEpoch
DescribeConfigs	トピック	DescribeConfigs
読み取り	トピック	Fetch、OffsetCommit、TxnOffsetCommit
書き込み (deny のみ)	トピック	Produce、AddPartitionsToTxn

MSK Replicator は、リソースタイプがトピックの場合にのみ、LITERAL パターンタイプの ACL をコピーします。PREFIXED パターンタイプの ACL、およびその他のリソースタイプの ACL はコピーされません。MSK Replicator は、ターゲットクラスターの ACL を削除することもありません。ソースクラスターの ACL を削除する場合は、ターゲットクラスターの ACL も同時に削除する必要があります。Kafka ACL リソース、パターン、および操作の詳細については、「https://kafka.apache.org/documentation/#security_authz_cli」を参照してください。

MSK Replicator は、IAM アクセスコントロールでは使用されない Kafka ACL のみをレプリケートします。クライアントで IAM アクセスコントロールを使用して MSK クラスターへの読み込み/書き込みを行っている場合は、シームレスなフェイルオーバーを実現するために、ターゲットクラスターでも関連する IAM ポリシーを設定する必要があります。これは、プレフィックス付きトピック名レプリケーション設定と同一トピック名レプリケーション設定の両方にも当てはまります。

コンシューマーグループのオフセット同期の一環として、MSK Replicator は、ストリームの先端 (トピックパーティションの末尾) に近い場所から読み込みを行っているソースクラスターのコンシューマーを最適化します。ソースクラスターのコンシューマーグループが遅延している場合、ターゲットのコンシューマーグループの遅延はソースと比較して大きくなる可能性があります。つまり、ターゲットクラスターへのフェイルオーバー後、コンシューマーは重複したメッセージをさらに再処理することになります。この遅延を小さくするには、ソースクラスターのコンシューマーがストリームの先端 (トピックパーティションの末尾) にキャッチアップするように処理を行い、消費を開始する必要があります。コンシューマーによる処理がキャッチアップすると、MSK Replicator は自動的に遅延を小さくします。

トピック名の設定

MSK Replicator には、プレフィックス付きトピック名レプリケーション (デフォルト) または同一トピック名レプリケーションの 2 つのトピック名設定モードが用意されています。

プレフィックス付きトピック名レプリケーション

デフォルトでは、MSK Replicator は、自動生成されたプレフィックスをソースクラスターのトピック名に追加した新しいトピック (`<sourceKafkaClusterAlias>.topic` など) をターゲットクラスターに作成します。これは、レプリケートされたトピックをターゲットクラスターの他のトピックと区別し、クラスター間での循環レプリケーションを回避するためです。

例えば、MSK Replicator は、ソースクラスターから「topic」という名前のトピックのデータを、<sourceKafkaClusterAlias>.topic という名前のターゲットクラスターの新しいトピックにレプリケートします。ターゲットクラスターのトピック名に追加されるプレフィックスは、`DescribeReplicator` API を使用して [sourceKafkaClusterAlias] フィールドで確認するか、MSK コンソールの [レプリケーター] 詳細ページで確認することができます。ターゲットクラスターのプレフィックスは、<sourceKafkaClusterAlias> です。

コンシューマーがスタンバイクラスターから確実に処理を再開できるようにするには、ワイルドカード演算子 `.*` を使用してトピックからデータを読み取るようにコンシューマーを設定する必要があります。たとえば、コンシューマーは両方の AWS リージョンで `*topic1`を使用して を消費する必要があります。この例には `footopic1` などのトピックも含まれるため、必要に応じてワイルドカード演算子を調整してください。

アクティブ/アクティブクラスター設定など、レプリケーターデータをターゲットクラスター内の別のトピックに保持する場合は、プレフィックスを追加できる MSK レプリケーターを使用する必要があります。

同一トピック名レプリケーション

デフォルト設定の代わりに、Amazon MSK Replicator では、トピックのレプリケーションを同一トピック名レプリケーション (コンソールで同じトピック名を保持) に設定したレプリケーターを作成できます。ターゲット MSK クラスターがある AWS リージョンに新しいレプリケーターを作成できます。同じ名前のレプリケートされたトピックを使用すると、レプリケートされたトピックから読み込むようにクライアントを再設定する必要がなくなります。

同一トピック名レプリケーション (コンソールで同じトピック名を保持) には、次の利点があります。

- レプリケーション処理中に同一のトピック名を保持しながら、無限のレプリケーションループのリスクを自動的に回避できます。
- レプリケートされたトピックから読み込むようにクライアントを再設定する必要がなくなるため、マルチクラスターストリーミングアーキテクチャの設定と運用が簡単になります。
- アクティブ/パッシブクラスターアーキテクチャの場合、同一トピック名レプリケーション機能によってフェイルオーバープロセスも効率化されるため、トピック名の変更やクライアントの再設定を必要とすることなく、アプリケーションをスタンバイクラスターにシームレスにフェイルオーバーできるようになります。
- 複数の MSK クラスターからのデータを 1 つのクラスターに簡単に統合して、データの集約や集中分析を行うことができます。そのためには、ソースクラスターおよび同じターゲットクラスターごとに別個のレプリケーターを作成する必要があります。
- ターゲットクラスター内の同一の名前のトピックにデータをレプリケートすることで、ある MSK クラスターから別の MSK クラスターへのデータ移行を効率化できます。

Amazon MSK Replicator は、Kafka ヘッダーを使用して、データが元のトピックにレプリケートされることを自動的に回避し、レプリケーション中に無限サイクルが発生するリスクを排除します。ヘッダーは、各 Kafka メッセージ内のキー、値、およびタイムスタンプに含めることができるキーと値のペアです。MSK Replicator は、レプリケートされる各レコードのヘッダーにソースクラスターとトピックの識別子を埋め込みます。MSK Replicator は、ヘッダー情報を使用して無限のレプリケーションループを回避します。クライアントがレプリケートされたデータを期待どおりに読み込めることを確認する必要があります。

チュートリアル: Amazon MSK Replicator のソースクラスターとターゲットクラスターを設定する

このチュートリアルでは、ソースクラスターとターゲットクラスターを同じ AWS リージョンまたは異なる AWS リージョンに設定する方法を示します。次に、それらのクラスターを使用して Amazon MSK Replicator を作成します。

Amazon MSK ソースクラスターを準備する

MSK レプリケーター用の MSK ソースクラスターを既に作成している場合は、このセクションで説明する要件を満たしていることを確認してください。それ以外の場合は、次のステップを実行して MSK プロビジョンドソースクラスターまたはサーバーレスソースクラスターを作成してください。

クロスリージョンと同一リージョンの MSK レプリケーターのソースクラスターを作成するプロセスは類似しています。相違点については次の手順で説明します。

1. ソースリージョンで、IAM アクセスコントロールを有効にした MSK プロビジョンドクラスターまたはサーバーレスクラスターを作成します。ソースクラスターには、少なくとも 3 つのブローカーが必要です。
2. クロスリージョン MSK レプリケーターでは、ソースがプロビジョンドクラスターの場合は、IAM アクセスコントロールスキーム用にマルチ VPC プライベート接続を有効にして設定します。マルチ VPC を有効にした場合、認証されていない認証タイプはサポートされないので注意してください。他の認証スキーム (mTLS や SASL/SCRAM) では、マルチ VPC プライベート接続を有効にする必要はありません。MSK クラスターに接続する他のクライアントに、mTLS または SASL/SCRAM 認証スキームを同時に使用することは可能です。マルチ VPC プライベート接続は、コンソールのクラスター詳細の [ネットワーク設定] で、または `UpdateConnectivity` API を使用して設定できます。「クラスター所有者がマルチ VPC を有効にする」を参照してください。ソースクラスターが MSK サーバーレスクラスターの場合、マルチ VPC プライベート接続を有効にする必要はありません。

   同一リージョン MSK レプリケーターの場合、MSK ソースクラスターにマルチ VPC プライベート接続は必要なく、認証されていない認証タイプを使用する他のクライアントからでもそのソースクラスターにアクセスできます。
3. クロスリージョン MSK レプリケーターの場合、ソースクラスターにリソースベースのアクセス許可ポリシーをアタッチする必要があります。これにより、MSK がこのクラスターに接続してデータをレプリケートできます。これを行うには、以下の CLI または AWS コンソールの手順を使用します。「Amazon MSK のリソースベースのポリシー」も参照してください。同一リージョン MSK レプリケーターの場合は、このステップを実行する必要はありません。

Console: create resource policy

次の JSON でソースクラスターポリシーを更新します。プレースホルダーは、ソースクラスターの ARN に置き換えてください。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
```

```
        "Effect": "Allow",
        "Principal": {
            "Service": [
                "kafka.amazonaws.com"
            ]
        },
        "Action": [
            "kafka:CreateVpcConnection",
            "kafka:GetBootstrapBrokers",
            "kafka:DescribeClusterV2"
        ],
        "Resource": "arn:aws:kafka:us-
east-1:123456789012:cluster/myCluster/abcd1234-5678-90ab-cdef-1234567890ab-1"
    }
  ]
}
```

クラスター詳細ページの [アクション] メニューにある [クラスターポリシーの編集] オプションを使用します。

CLI: create resource policy

注: AWS コンソールを使用してソースクラスターを作成し、新しい IAM ロールを作成するオプションを選択した場合、 は必要な信頼ポリシーをロールに AWS アタッチします。MSK で既存の IAM ロールを使用する場合、または独自にロールを作成する場合は、MSK レプリケーターがロールを引き受けることができるように、次の信頼ポリシーをそのロールにアタッチします。ロールの信頼関係を変更する方法については、「ロールの修正」を参照してください。

1. このコマンドを使用して、MSK クラスターポリシーの現在のバージョンを取得します。プレースホルダーは、実際のクラスター ARN に置き換えてください。

```
aws kafka get-cluster-policy –cluster-arn <Cluster ARN>
{
"CurrentVersion": "K1PA6795UKM GR7",
"Policy": "..."
}
```

2. MSK レプリケーターがソースクラスターにアクセスすることを許可するリソースベースのポリシーを作成します。次の構文をテンプレートとして使用します。プレースホルダーは、実際のソースクラスター ARN に置き換えてください。

```
aws kafka put-cluster-policy --cluster-arn "<sourceClusterARN>" --policy '{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Principal": {
"Service": [
"kafka.amazonaws.com"
]
},
"Action": [
"kafka:CreateVpcConnection",
"kafka:GetBootstrapBrokers",
"kafka:DescribeClusterV2"
],
"Resource": "<sourceClusterARN>"
}
]
```

Amazon MSK ターゲットクラスターを準備する

IAM アクセスコントロールを有効にして、MSK ターゲットクラスター (プロビジョンドまたはサーバーレス) を作成します。ターゲットクラスターでは、マルチ VPC プライベート接続を有効にする必要はありません。ターゲットクラスターは、ソースクラスターと同じ AWS リージョンまたは別のリージョンにあることができます。ソースクラスターとターゲットクラスターの両方が同じ AWS アカウントにある必要があります。ターゲットクラスターには、少なくとも 3 つのブローカーが必要です。

チュートリアル: Amazon MSK Replicator を作成する

ソースクラスターとターゲットクラスターを設定したら、それらのクラスターを使用して Amazon MSK Replicator を作成できます。Amazon MSK Replicator を作成する前に、MSK レプリケーターの作成に必要な IAM アクセス許可を持っていることを確認してください。

トピック

- Amazon MSK Replicator を作成する際の考慮事項
  - MSK レプリケーターの作成に必要な IAM アクセス許可
  - MSK Replicator でサポートされるクラスタータイプとバージョン

- サポートされている MSK サーバーレスクラスター設定
  - クラスター設定の変更
- ターゲットクラスターリージョンの AWS コンソールを使用したレプリケータの作成
  - ソースクラスターの選択
  - ターゲットクラスターの選択
  - レプリケータの設定とアクセス許可を設定します。

Amazon MSK Replicator を作成する際の考慮事項

以下のセクションでは、MSK レプリケーター機能を使用するための前提条件、サポートされている設定、およびベストプラクティスの概要を説明します。ここでは、必要なアクセス許可、クラスターの互換性、サーバーレス固有の要件、および作成後のレプリケーターの管理に関するガイダンスについて説明します。

MSK レプリケーターの作成に必要な IAM アクセス許可

MSK レプリケーターの作成に必要な IAM ポリシーの例を次に示します。アクション `kafka:TagResource` は、MSK レプリケーターの作成時にタグを指定する場合にのみ必要です。レプリケーターの IAM ポリシーは、クライアントに対応する IAM ロールにアタッチする必要があります。認可ポリシーの作成については、「認可ポリシーの作成」を参照してください。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MSKReplicatorIAMPassRole",
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/MSKReplicationRole",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "kafka.amazonaws.com"
        }
      }
    },
    {
```

```
      "Sid": "MSKReplicatorServiceLinkedRole",
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::123456789012:role/aws-service-role/
kafka.amazonaws.com/AWSServiceRoleForKafka*"
    },
    {
      "Sid": "MSKReplicatorEC2Actions",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs",
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:123456789012:subnet/subnet-0abcd1234ef56789",
        "arn:aws:ec2:us-east-1:123456789012:security-group/sg-0123abcd4567ef89",
        "arn:aws:ec2:us-east-1:123456789012:network-
interface/eni-0a1b2c3d4e5f67890",
        "arn:aws:ec2:us-east-1:123456789012:vpc/vpc-0a1b2c3d4e5f67890"
      ]
    },
    {
      "Sid": "MSKReplicatorActions",
      "Effect": "Allow",
      "Action": [
        "kafka:CreateReplicator",
        "kafka:TagResource"
      ],
      "Resource": [
        "arn:aws:kafka:us-
east-1:123456789012:cluster/myCluster/abcd1234-56ef-78gh-90ij-klmnopqrstuv",
        "arn:aws:kafka:us-
east-1:123456789012:replicator/myReplicator/wxyz9876-54vu-32ts-10rq-ponmlkjihgfe"
      ]
    }
  ]
}
```

レプリケーターを記述する IAM ポリシーの例を次に示します。kafka:DescribeReplicator アクションと kafka:ListTagsForResource アクションは、いずれか一方が必要ですが、両方は必要ありません。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "kafka:DescribeReplicator",
                "kafka:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

MSK Replicator でサポートされるクラスタータイプとバージョン

以下は、サポートされるインスタンスタイプ、Kafka バージョン、ネットワーク設定の要件です。

- MSK レプリケーターは、MSK プロビジョンドクラスターと MSK サーバーレスクラスターの両方を、ソースクラスターとターゲットクラスターとしての任意の組み合わせでサポートします。現時点では、他のタイプの Kafka クラスターは MSK レプリケーターでサポートされていません。
- MSK サーバーレスクラスターには IAM アクセスコントロールが必要で、Apache Kafka ACL レプリケーションはサポートされておらず、トピック固有の設定のレプリケーションは制限付きでサポートされています。「MSK サーバーレスとは」を参照してください。
- MSK レプリケーターは、ソースクラスターとターゲットクラスターが同じか異なるかに関係なく、Apache Kafka 2.7.0 以降を実行しているクラスターでのみサポートされます AWS リージョン。
- MSK レプリケーターは、m5.large 以上のインスタンスタイプを使用するクラスターをサポートしています。t3.small クラスターはサポートされていません。
- MSK レプリケーターを MSK プロビジョンドクラスターで使用する場合、ソースクラスターとターゲットクラスターの両方に少なくとも 3 つのブローカーが必要です。2 つのアベイラビリ

ティーゾーンのクラスター間でデータをレプリケートできますが、それらのクラスターには少なくとも 4 つのブローカーが必要です。

- ソース MSK クラスターとターゲット MSK クラスターの両方が同じ AWS アカウントにある必要があります。異なるアカウントのクラスター間でのレプリケーションはサポートされません。
- ソース MSK クラスターとターゲット MSK クラスターが異なる AWS リージョン (クロスリージョン) にある場合、MSK レプリケーターでは、IAM アクセスコントロールメソッドでマルチ VPC プライベート接続を有効にする必要があります。

  マルチ VPC は、MSK レプリケーションのソースクラスター上の他の認証方法には必要ありません AWS リージョン。

  同じ 内のクラスター間でデータをレプリケートする場合も、マルチ VPC は必要ありません AWS リージョン。「the section called “単一リージョンのマルチ VPC プライベート接続”」を参照してください。
- 同一トピック名レプリケーション (コンソールで同じトピック名を保持) には、Kafka バージョン 2.8.1 以降を実行する MSK クラスターが必要です。
- 同一トピック名レプリケーション (コンソールで同じトピック名を保持) 設定では、循環レプリケーションのリスクを回避するために、MSK Replicator が作成するヘッダー (`__mskmr`) を変更しないでください。

サポートされている MSK サーバーレスクラスター設定

- MSK サーバーレスは、トピックの作成中に MSK サーバーレスターゲットクラスターに対する次のトピック設定のレプリケートをサポートします: `cleanup.policy`、`compression.type`、`max.message.bytes`、`retention.bytes`、`retention.`
- MSK サーバーレスは、トピック設定の同期中に次のトピック設定のみをサポートします: `compression.type`、`max.message.bytes`、`retention.bytes`、`retention.ms`。
- レプリケーターは、ターゲット MSK サーバーレスクラスター上の 83 個の圧縮パーティションを使用します。ターゲット MSK サーバーレスクラスターに十分な数の圧縮パーティションがあることを確認してください。「MSK サーバーレス クォータ」を参照してください。

クラスター設定の変更

- MSK レプリケーターの作成後は、階層型ストレージの有効/無効を変更しないことが推奨されます。ターゲットクラスターが階層化されていない場合、ソースクラスターが階層化されているかどうかに関係なく、MSK は階層型ストレージ設定をコピーしません。レプリケーターの作成後に

ターゲットクラスターの階層型ストレージを有効にする場合は、レプリケーターを再作成する必要があります。非階層型クラスターから階層型クラスターにデータをコピーする場合は、トピック設定をコピーしないでください。「既存のトピックでの階層型ストレージの有効化と無効化」を参照してください。

- MSK レプリケーターの作成後にクラスター設定の設定内容を変更しないでください。クラスター設定の設定内容は MSK レプリケーターの作成中に検証されます。MSK レプリケーターでの問題を回避するために、MSK レプリケーターの作成後に次の設定を変更しないでください。
  - MSK クラスターを t3 インスタンスタイプに変更する。
  - サービス実行ロールのアクセス許可を変更する。
  - MSK マルチ VPC プライベート接続を無効にする。
  - アタッチされたクラスターのリソースベースポリシーを変更する。
  - クラスターセキュリティグループルールを変更する。

ターゲットクラスターリージョンの AWS コンソールを使用したレプリケータの作成

次のセクションでは、レプリケーターを作成するためのステップごとのコンソールワークフローについて説明します。

レプリケーターの詳細

1. ターゲット MSK クラスターがある AWS リージョンで、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. [レプリケーター] を選択すると、アカウント内のレプリケーターのリストが表示されます。
3. [レプリケータの作成] を選択します。
4. [レプリケータの詳細] ペインで、新しいレプリケーターに一意の名前を付けます。

ソースクラスターの選択

ソースクラスターには、ターゲット MSK クラスターにコピーするデータが含まれています。

1. [ソースクラスター] ペインで、ソースクラスターが配置されている AWS リージョンを選択します。

クラスターのリージョンを調べるには、[MSK クラスター] に移動し、[クラスター] の詳細で ARN を確認します。リージョン名は ARN 文字列に埋め込まれています。次の例の ARN では、ap-southeast-2 がクラスターリージョンです。

```
arn:aws:kafka:ap-southeast-2:123456789012:cluster/cluster-11/
eec93c7f-4e8b-4baf-89fb-95de01ee639c-s1
```

2. ソースクラスターの ARN を入力するか、参照してソースクラスターを選択します。
3. ソースクラスターのサブネットを選択します。

   コンソールには、ソースクラスターのリージョンで使用可能なサブネットが表示され、そこから選択できます。少なくとも 2 つのサブネットを選択する必要があります。同一リージョン MSK レプリケーターの場合、ソースクラスターにアクセスするために選択するサブネットと、ターゲットクラスターにアクセスするためのサブネットが、同じアベイラビリティーゾーンにある必要があります。
4. MSK レプリケーターがソースクラスターにアクセスするためのセキュリティグループを選択します。

   - クロスリージョンレプリケーション (CRR) の場合、ソースクラスターのセキュリティグループを指定する必要はありません。
   - 同一リージョンレプリケーション (SRR) の場合は、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) にアクセスし、レプリケーター用に指定するセキュリティグループに、ソースクラスターのセキュリティグループへのトラフィックを許可するアウトバウンドルールがあることを確認します。同様に、ソースクラスターのセキュリティグループに、ソース用に指定したレプリケーターセキュリティグループからのトラフィックを許可するインバウンドルールがあることを確認します。

     ソースクラスターのセキュリティグループにインバウンドルールを追加するには

     1. AWS コンソールで、クラスター名を選択してソースクラスターの詳細に移動します。
     2. [プロパティ] タブを選択し、[ネットワーク設定] ペインまで下にスクロールして、適用するセキュリティグループの名前を選択します。
     3. インバウンドルールに移動し、[インバウンドルールの編集] を選択します。
     4. [Add rule] (ルールを追加) を選択します。
     5. 新しいルールの [タイプ] 列で、[カスタム TCP] を選択します。

6. [ポート範囲] 列に「9098」と入力します。MSK Replicator は IAM アクセスコントロールを使用して、9098 番ポートを使用するクラスターに接続します。
7. [ソース] 列に、レプリケーター作成時にソースクラスター用に指定するセキュリティグループの名前 (これは、MSK ソースクラスターのセキュリティグループと同じになる場合があります) を入力し、[ルールの保存] を選択します。

ソース用に指定したレプリケーターのセキュリティグループにアウトバウンドルールを追加するには

1. Amazon EC2 の AWS コンソールで、ソースのレプリケーターの作成時に指定するセキュリティグループに移動します。
2. アウトバウンドルールに移動し、[アウトバウンドルールの編集] を選択します。
3. [Add rule] (ルールを追加) を選択します。
4. 新しいルールの [タイプ] 列で、[カスタム TCP] を選択します。
5. [ポート範囲] 列に「9098」と入力します。MSK Replicator は IAM アクセスコントロールを使用して、9098 番ポートを使用するクラスターに接続します。
6. [ソース] 列に、MSK ソースクラスターのセキュリティグループの名前を入力し、[ルールの保存] を選択します。

> **Note**
>
> あるいは、セキュリティグループを使用したトラフィックの制限を希望しない場合は、すべてのトラフィックを許可するインバウンドルールとアウトバウンドルールを追加できます。
> 1. [Add rule] (ルールを追加) を選択します。
> 2. [タイプ] 列で [すべてのトラフィック] を選択します。
> 3. [ソース] 列に `0.0.0.0/0` と入力し、[ルールの保存] を選択します。

ターゲットクラスターの選択

ターゲットクラスターは、ソースデータのコピー先となる、MSK プロビジョンドクラスターまたはサーバーレスクラスターです。

> **Note**
>
> MSK レプリケーターは、ターゲットクラスターに新しいトピックを作成し、自動生成されたプレフィックスをトピック名に追加します。例えば、MSK レプリケーターは、ソースクラスターの「topic」内のデータをターゲットクラスターの <sourceKafkaClusterAlias>.topic という名前の新しいトピックにレプリケートします。これは、ソースクラスターからレプリケートされたデータを含むトピックを、ターゲットクラスターの他のトピックと区別し、データがクラスター間で循環的にレプリケートされないようにするためです。ターゲットクラスターでトピック名に追加されるプレフィックスは、DescribeReplicator API を使用して sourceKafkaClusterAlias フィールドで確認するか、または MSK コンソールの [レプリケータの詳細] ページで確認することができます。ターゲットクラスターのプレフィックスは <sourceKafkaClusterAlias> です。

1. ターゲットクラスターペインで、ターゲットクラスターがある AWS リージョンを選択します。
2. ターゲットクラスターの ARN を入力するか、参照してターゲットクラスターを選択します。
3. ターゲットクラスターのサブネットを選択します。

   コンソールには、ターゲットクラスターのリージョンで使用可能なサブネットが表示され、そこから選択できます。少なくとも 2 つのサブネットを選択します。
4. MSK レプリケーターがターゲットクラスターにアクセスするためのセキュリティグループを選択します。

   ターゲットクラスターのリージョンで使用可能なセキュリティグループが表示され、そこから選択できます。選択したセキュリティグループは各接続に関連付けられます。セキュリティグループの使用の詳細については、「Amazon VPC ユーザーガイド」の「セキュリティグループを使用して AWS リソースへのトラフィックを制御する」を参照してください。

   - クロスリージョンレプリケーション (CRR) と同一リージョンレプリケーション (SRR) のどちらの場合も、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) にアクセスし、レプリケーター用に指定するセキュリティグループに、ターゲットクラスターのセキュリティグループへのトラフィックを許可するアウトバウンドルールがあることを確認します。また、ターゲットクラスターのセキュリティグループに、ターゲット用に指定されたレプリケーターセキュリティグループからのトラフィックを受け入れるインバウンドルールがあることを確認します。

ターゲットクラスターのセキュリティグループにインバウンドルールを追加するには

1. AWS コンソールで、クラスター名を選択してターゲットクラスターの詳細に移動します。
2. [プロパティ] タブを選択し、[ネットワーク設定] ペインまで下にスクロールして、適用するセキュリティグループの名前を選択します。
3. インバウンドルールに移動し、[インバウンドルールの編集] を選択します。
4. [Add rule] (ルールを追加) を選択します。
5. 新しいルールの [タイプ] 列で、[カスタム TCP] を選択します。
6. [ポート範囲] 列に「9098」と入力します。MSK Replicator は IAM アクセスコントロールを使用して、9098 番ポートを使用するクラスターに接続します。
7. [ソース] 列に、レプリケーター作成時にターゲットクラスター用に指定するセキュリティグループの名前 (これは、MSK ターゲットクラスターのセキュリティグループと同じになる場合があります) を入力し、[ルールの保存] を選択します。

ターゲット用に指定したレプリケーターのセキュリティグループにアウトバウンドルールを追加するには

1. AWS コンソールで、ターゲットのレプリケーターの作成時に指定するセキュリティグループに移動します。
2. [プロパティ] タブを選択し、[ネットワーク設定] ペインまで下にスクロールして、適用するセキュリティグループの名前を選択します。
3. アウトバウンドルールに移動し、[アウトバウンドルールの編集] を選択します。
4. [Add rule] (ルールを追加) を選択します。
5. 新しいルールの [タイプ] 列で、[カスタム TCP] を選択します。
6. [ポート範囲] 列に「9098」と入力します。MSK Replicator は IAM アクセスコントロールを使用して、9098 番ポートを使用するクラスターに接続します。
7. [ソース] 列で、MSK ターゲットクラスターのセキュリティグループの名前を入力し、[ルールの保存] を選択します。

> **Note**
>
> あるいは、セキュリティグループを使用したトラフィックの制限を希望しない場合は、すべてのトラフィックを許可するインバウンドルールとアウトバウンドルールを追加できます。
> 1. [Add rule] (ルールを追加) を選択します。

2. [タイプ] 列で [すべてのトラフィック] を選択します。
3. [ソース] 列に `0.0.0.0/0` と入力し、[ルールの保存] を選択します。

レプリケータの設定とアクセス許可を設定します。

1. [レプリケータ設定] ペインで、レプリケートするトピックを、正規表現を使用して許可リストと拒否リストに指定します。デフォルトでは、すべてのトピックがレプリケートされます。

   > **Note**
   >
   > MSK Replicator は、ソート順に最大 750 のトピックのみをレプリケートします。より多くのトピックをレプリケートする必要がある場合は、別のレプリケーターを作成することをお勧めします。レプリケーターあたり 750 を超えるトピックのサポートが必要な場合は、 AWS コンソールサポートセンターに移動し、サポートケースを作成します。レプリケートされるトピックの数は、「TopicCount」メトリクスを使用してモニタリングできます。「Amazon MSK Standard ブローカークォータ」を参照してください。

2. デフォルトでは、MSK Replicator は、選択したトピックの最も新しいオフセットからレプリケーションを開始します。あるいは、トピックの既存のデータをレプリケートする場合は、選択したトピックの最も古いオフセットからレプリケーションを開始できます。レプリケーターが作成されると、この設定を変更することはできません。この設定は、`CreateReplicator` リクエスト API と `DescribeReplicator` レスポンス API の `startingPosition` フィールドに対応します。

3. トピック名の設定を選択します。

   - `PREFIXED` トピック名レプリケーション (コンソールでトピック名にプレフィックスを追加): デフォルト設定。MSK Replicator は、ソースクラスターの「topic1」をターゲットクラスター内の新しいトピックに `<sourceKafkaClusterAlias>.topic1` という名前でレプリケートします。
   - 同一トピック名レプリケーション (コンソールで同じトピック名を保持): ソースクラスターのトピックは、ターゲットクラスターに同一のトピック名でレプリケートされます。

   この設定は、`CreateReplicator` リクエスト API と `DescribeReplicator` レスポンス API の `TopicNameConfiguration` フィールドに対応します。「Amazon MSK Replicator の仕組み」を参照してください。

> **Note**
>
> デフォルトでは、MSK Replicator は、自動生成されたプレフィックスをトピック名に追加した新しいトピックをターゲットクラスターに作成します。これは、ソースクラスターからレプリケートされたデータを含むトピックを、ターゲットクラスターの他のトピックと区別し、データがクラスター間で循環的にレプリケートされないようにするためです。あるいは、同一トピック名レプリケーション (コンソールで同じトピック名を保持) を使用して MSK レプリケーターを作成し、レプリケーション中にトピック名が保持されるようにすることもできます。この設定により、セットアップ中にクライアントアプリケーションを再設定する必要性が減り、マルチクラスターストリーミングアーキテクチャの操作が簡単になります。

4. デフォルトでは、MSK レプリケーターは、トピック設定、アクセスコントロールリスト (ACL)、コンシューマーグループのオフセットを含むすべてのメタデータをコピーし、シームレスなフェイルオーバーを実現します。フェイルオーバー用にレプリケーターを作成しない場合は、オプションで、[追加設定] セクションで使用可能なこれらの設定の 1 つ以上を無効にすることができます。

> **Note**
>
> MSK レプリケーターは書き込み ACL をレプリケートしません。これは、プロデューサーがターゲットクラスター内のレプリケートされたトピックに直接書き込みを行えないようにするためです。プロデューサーは、フェイルオーバー後にターゲットクラスターのローカルトピックに書き込む必要があります。詳細については、「セカンダリ AWS リージョンへの計画的なフェイルオーバーを実行する」を参照してください。

5. [コンシューマーグループのレプリケーション] ペインで、レプリケートするコンシューマーグループを、正規表現を使用して許可リストと拒否リストに指定します。デフォルトでは、すべてのコンシューマーグループがレプリケートされます。
6. [圧縮] ペインでは、オプションで、ターゲットクラスターに書き込まれたデータを圧縮することを選択できます。圧縮を使用する場合は、ソースクラスターのデータと同じ圧縮方式を使用することが推奨されます。
7. [アクセス許可] ペインで、次のいずれかを実行します。

a. [必要なポリシーで IAM ロールを作成または更新] を選択します。MSK コンソールは、ソースおよびターゲット MSK クラスターに対する読み取りと書き込みに必要なサービス実行ロールに、必要なアクセス許可と信頼ポリシーを自動的にアタッチします。

b. [Amazon MSK が引き受けることができる IAM ロールから選択] を選択して、独自の IAM ロールを指定します。独自の IAM ポリシーを記述するのではなく、AWSMSKReplicatorExecutionRole マネージド IAM ポリシーをサービス実行ロールにアタッチすることをお勧めします。

- 信頼ポリシーの一部として以下の JSON を使用し、ロールに AWSMSKReplicatorExecutionRole をアタッチして、レプリケーターがソースおよびターゲット MSK クラスターへの読み込みと書き込みに使用する IAM ロールを作成します。す。信頼ポリシー内のプレースホルダー <yourAccountID> は、実際のアカウント ID に置き換えてください。

  JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "kafka.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<yourAccountID>"
                }
            }
        }
    ]
}
```

8. [レプリケータータグ] ペインでは、オプションで、MSK レプリケーターリソースにタグを割り当てることができます。詳細については、「Amazon MSK クラスターをタグ付ける」を参照してください。クロスリージョン MSK レプリケーターの場合、タグは、レプリケーターの作成時に自動的にリモートリージョンに同期されます。レプリケーターの作成後にタグを変更しても、

その変更はリモートリージョンに自動的に同期されないため、ローカルレプリケーターとリモートレプリケーターのリファレンスを手動で同期する必要があります。

9. [作成] を選択します。

`kafka-cluster:WriteData` アクセス許可を制限する場合は、「Amazon MSK の IAM アクセスコントロールの仕組み」の「承認ポリシーの作成」セクションを参照してください。ソースクラスターとターゲットクラスターの両方に `kafka-cluster:WriteDataIdempotently` アクセス許可を追加する必要があります。

MSK レプリケーターが正常に作成され、RUNNING ステータスに移行するまでに約 30 分かかります。

削除した MSK レプリケーターを置き換えるために新しい MSK レプリケーターを作成すると、新しいレプリケーターは最新のオフセットからレプリケーションを開始します。

MSK レプリケーターが FAILED ステータスに移行した場合は、トラブルシューティングのセクション「Troubleshooting MSK Replicator」を参照してください。

MSK レプリケータ設定の編集

MSK レプリケーターの作成後は、ソースクラスター、ターゲットクラスター、レプリケーターの開始位置、またはトピック名レプリケーション設定を変更することはできません。同一トピック名レプリケーション設定を使用するには、新しいレプリケーターを作成する必要があります。ただし、レプリケートするトピックやコンシューマーグループなど、その他のレプリケーター設定については編集が可能です。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。
2. 左側のナビゲーションペインで、[レプリケーター] を選択してアカウント内のレプリケーターのリストを表示し、編集する MSK レプリケーターを選択します。
3. [プロパティ] タブを選択します。
4. [レプリケータ設定] セクションで、[レプリケーターを編集] を選択します。
5. これらの任意の設定を変更することで、MSK レプリケータ設定を編集できます。

   - 許可リストと拒否リストで、レプリケートするトピックを正規表現を使用して指定します。デフォルトでは、MSK レプリケーターは、トピック設定、アクセスコントロールリスト (ACL)、コンシューマーグループのオフセットを含むすべてのメタデータをコピーし、シーム

レスなフェイルオーバーを実現します。フェイルオーバー用にレプリケーターを作成しない場合は、オプションで、[追加設定] セクションで使用可能なこれらの設定の 1 つ以上を無効にすることができます。

> **Note**
>
> MSK レプリケーターは書き込み ACL をレプリケートしません。これは、プロデューサーがターゲットクラスター内のレプリケートされたトピックに直接書き込みを行えないようにするためです。プロデューサーは、フェイルオーバー後にターゲットクラスターのローカルトピックに書き込む必要があります。詳細については、「セカンダリ AWS リージョンへの計画的なフェイルオーバーを実行する」を参照してください。

- [コンシューマーグループのレプリケーション] では、レプリケートするコンシューマーグループを、正規表現を使用して許可リストと拒否リストに指定できます。デフォルトでは、すべてのコンシューマーグループがレプリケートされます。許可リストと拒否リストが空の場合、コンシューマーグループのレプリケーションは無効です。
- [ターゲット圧縮タイプ] では、ターゲットクラスターに書き込まれるデータを圧縮するかどうかを選択できます。圧縮を使用する場合は、ソースクラスターのデータと同じ圧縮方式を使用することが推奨されます。

6. 変更内容を保存します。

   MSK レプリケーターが正常に作成され、実行中状態に移行するまでに約 30 分かかります。MSK レプリケーターが FAILED ステータスに移行した場合は、トラブルシューティングのセクション「???」を参照してください。

MSK レプリケーターの削除

作成に失敗した場合 (FAILED ステータス)、MSK レプリケーターは削除する必要がある場合があります。MSK レプリケーターに割り当てられたソースクラスターとターゲットクラスターは、レプリケーターの作成後に変更することはできません。既存の MSK レプリケーターを削除して、新しいレプリケーターを作成できます。新しい MSK レプリケーターを作成して、削除したレプリケーターと置き換える場合、新しいレプリケーターは最新のオフセットからレプリケーションを開始します。

1. ソースクラスターがある AWS リージョンで、 にサインインし AWS Management Console、https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。

2. ナビゲーションペインで、[レプリケーター] を選択します。

3. MSK レプリケーターのリストから、削除するレプリケーターを選択し、[削除] を選択します。

レプリケーションのモニタリング

ターゲットクラスターリージョンで https://console.aws.amazon.com/cloudwatch/ を使用すると、`ReplicationLatency`、`MessageLag`、`ReplicatorThroughput` のメトリクスを各 Amazon MSK Replicator のトピックレベルと集計レベルで表示できます。メトリクスは、「AWS/Kafka」名前空間の [ReplicatorName] の下に表示されます。また、`ReplicatorFailure`、`AuthError`、`ThrottleTime` メトリクスを参照して、問題がないかを確認することもできます。

MSK コンソールには、各 MSK レプリケーターの CloudWatch メトリクスのサブセットが表示されます。コンソールの [レプリケーター] リストから、レプリケーターの名前を選択し、[モニタリング] タブを選択します。

MSK レプリケーターのメトリクス

次のメトリクスは、MSK レプリケーターのパフォーマンスまたは接続のメトリクスを示しています。

AuthError メトリクスには、トピックレベルの認証エラーは含まれません。MSK レプリケーターのトピックレベルの認証エラーをモニタリングするには、レプリケーターの ReplicationLatency メトリクスとソースクラスターのトピックレベルのメトリクス MessagesInPerSec をモニタリングします。トピックの ReplicationLatency が 0 に低下してもトピックでまだデータが生成されている場合は、レプリケーターにそのトピックに関する認証の問題があることを示しています。レプリケーターのサービス実行 IAM ロールに、トピックにアクセスするための十分なアクセス許可があることを確認してください。

メトリクスのタイプ	メトリクス	説明	ディメンション	単位	未加工メトリクスの細分性	未加工メトリクスの集計統計	
パフォーマンス	ReplicationLatency	ソースクラスターからターゲットクラスターにレコードをレプリケートするのにかかる時間。ソースでレコードが作成されてからターゲットにレプリケートされるまでの時間。ReplicationLatency が増加した場合は、クラスターにレプリケーションをサポートするのに十分なパーティションがあるかどうかを確認してください。パーティション数が少なすぎて高スループットに対応できない場合、レプリケーションのレイテンシーが高くな	ReplicatorName	ミリ秒	パーティション	最大値	
			ReplicatorName、ic	ミリ秒	パーティション	最大値	

メトリクスのタイプ	メトリクス	説明	ディメンション	単位	未加工メトリクスの細分性	未加工メトリクスの集計統計	
		る可能性があります。					

メトリクスのタイプ	メトリクス	説明	ディメンション	単位	未加工メトリクスの細分性	未加工メトリクスの集計統計	
パフォーマンス	MessageLag	MSK レプリケーターとソースクラスター間の同期をモニタリングします。MessageLag は、ソースクラスターに生成されたメッセージとレプリケーターによって消費されたメッセージ間の遅延を示します。ソースクラスターとターゲットクラスター間の遅延ではありません。ソースクラスターが使用不可または中断された場合でも、レプリケーターはターゲットクラスターへの消費したメッセージの書き込みを完了させます。停止後、Messag	ReplicatorName	カウント	パーティション	合計	
			ReplicatorName、ic	カウント	パーティション	合計	

メトリクスのタイプ	メトリクス	説明	ディメンション	単位	未加工メトリクスの細分性	未加工メトリクスの集計統計	
		eLag は増加して、レプリケーターがソースクラスターに対して遅れているメッセージ数を示します。これは、メッセージ数が 0 になる (レプリケーターがソースクラスターにキャッチアップしたことを示す) までモニタリングできます。					

メトリクスのタイプ	メトリクス	説明	ディメンション	単位	未加工メトリクスの細分性	未加工メトリクスの集計統計	
パフォーマンス	ReplicatorBytesInPerSec	1 秒あたりのレプリケーターによって処理された平均バイト数。MSK Replicator によって処理されるデータは、MSK Replicator が受け取るすべてのデータで構成されます。これには、ターゲットクラスターにレプリケートされたデータと、データが元の同じトピックにコピーされないように MSK Replicator によってフィルタリングされたデータ (レプリケーターが同一トピック名設定で設定されている場合のみ) が含まれます。レプリケー	ReplicatorName	BytesPerSecond	ReplicatorName	合計	

メトリクスのタイプ	メトリクス	説明	ディメンション	単位	未加工メトリクスの細分性	未加工メトリクスの集計統計	
		ターが「プレフィクス付き」トピック名設定で設定されている場合、MSK Replicator によってデータがフィルタリングされないため、ReplicatorBytesInPerSec と ReplicatorThroughput メトリクスの両方が同じ値になります。					

メトリクスのタイプ	メトリクス	説明	ディメンション	単位	未加工メトリクスの細分性	未加工メトリクスの集計統計	
パフォーマンス	ReplicatorThroughput	1 秒あたりのレプリケートされた平均バイト数。トピックの ReplicatorThroughput が低下した場合は、KafkaClusterPingSuccessCount と AuthError メトリクスをチェックして、レプリケーターがクラスターと通信できることを確認してから、クラスターメトリクスをチェックしてクラスターがダウンしていないことを確認します。	ReplicatorName	BytesPerSecond	パーティション	合計	
			ReplicatorName、Topic	BytesPerSecond	パーティション	合計	

メトリクスのタイプ	メトリクス	説明	ディメンション	単位	未加工メトリクスの細分性	未加工メトリクスの集計統計	
デバッグ	AuthError	1 秒あたりの認証に失敗した接続の数。このメトリクスが 0 より大きい場合は、レプリケーターのサービス実行ロールポリシーが有効かどうかを確認し、クラスターのアクセス許可に対して拒否アクセス許可が設定されていないことを確認できます。clusterAlias ディメンションに基づいて、ソースクラスターまたはターゲットクラスターで認証エラーが発生しているかどうかを識別できます。	ReplicatorName、clusterAlias	カウント	ワーカー	合計	

メトリクスのタイプ	メトリクス	説明	ディメンション	単位	未加工メトリクスの細分性	未加工メトリクスの集計統計	
デバッグ	ThrottleTime	クラスター上でブローカーによってリクエストがスロットリングされた平均時間 (ミリ秒単位)。MSK レプリケーターがクラスターを圧迫しないように、スロットリングを設定します。このメトリクスが 0 で、replicationLatency が高くなく、replicatorThroughput が期待どおりであれば、スロットリングは期待どおりに機能しています。このメトリクスが 0 より大きい場合は、それに応じてスロットリングを調整できます。	ReplicatorName、sterAlias	ミリ秒	ワーカー	最大値	

メトリクスのタイプ	メトリクス	説明	ディメンション	単位	未加工メトリクスの細分性	未加工メトリクスの集計統計	
デバッグ	ReplicatorFailure	レプリケーターで発生している障害の数。	ReplicatorName	カウント		合計	
デバッグ	KafkaClusterPingSuccessCount	Kafka クラスターへのレプリケーター接続の正常性を示します。この値が 1 の場合、接続は正常です。値が 0 またはデータポイントがない場合、接続は異常です。値が 0 の場合は、Kafka クラスターのネットワークまたは IAM アクセス許可設定を確認してください。ClusterAlias ディメンションに基づいて、このメトリクスがソースクラスター用かターゲットクラスター用かを識別できます。	ReplicatorName、ClusterAlias	カウント		合計	

レプリケーションを使用してリージョン間の Kafka ストリーミングアプリケーションの耐障害性を高める

MSK レプリケーターを使用して、アクティブ/アクティブまたはアクティブ/パッシブのクラストポロジを設定し、 AWS リージョン間で Apache Kafka アプリケーションの耐障害性を高めることができます。アクティブ/アクティブ設定では、両方の MSK クラスターが読み取りと書き込みをアクティブに処理します。アクティブ/パッシブ設定では、一度に 1 つの MSK クラスターのみがストリーミングデータをアクティブに処理し、もう一方のクラスターはスタンバイ状態になります。

マルチリージョン Apache Kafka アプリケーションを構築する際の考慮事項

コンシューマーは、ダウンストリームに影響を与えることなく、重複したメッセージを再処理できる必要があります。MSK レプリケーターは、少なくとも 1 回はデータをレプリケートするため、スタンバイクラスターで重複が発生する可能性があります。セカンダリ AWS リージョンに切り替えると、コンシューマーが同じデータを複数回処理することがあります。MSK レプリケーターは、パフォーマンスを向上させるために、コンシューマーオフセットよりもデータのコピーを優先します。フェイルオーバー後に、コンシューマーが以前のオフセットから読み取りを開始し、それによって重複処理が発生する可能性があります。

プロデューサーとコンシューマーは、最小限のデータ損失を許容する必要もあります。MSK レプリケーターはデータを非同期的にレプリケートするため、プライマリ AWS リージョンで障害が発生し始めると、すべてのデータがセカンダリリージョンにレプリケートされる保証はありません。レプリケーションのレイテンシーに基づいて、セカンダリリージョンにコピーされなかった最大データ量を判別できます。

アクティブ/アクティブとアクティブ/パッシブのクラストポロジの使用

アクティブ/アクティブのクラストポロジを使用すると、復旧時間がほぼゼロになり、ストリーミングアプリケーションを複数の AWS リージョンで同時に動作させることができます。一方のリージョンのクラスターに障害が発生しても、もう一方のリージョンのクラスターに接続されているアプリケーションがデータの処理を続行します。

アクティブ/パッシブ設定は、一度に 1 つの AWS リージョンでのみ実行できるアプリケーションや、データ処理順序をより細かく制御する必要があるアプリケーションに適しています。アクティブ/パッシブ設定では、アクティブ/アクティブ設定よりも復旧時間が長くなります。これは、フェイルオーバー後にデータのストリーミングを再開するには、プロデューサーとコンシューマーを含めて、アクティブ/パッシブ設定全体をセカンダリリージョンで開始する必要があるためです。

推奨されるトピック命名設定を使用してアクティブ/パッシブ Kafka クラスター設定を作成する

アクティブ/パッシブ設定の場合、プロデューサー、MSK クラスター、コンシューマー (同じコンシューマーグループ名) の同様の設定を 2 つの異なる AWS リージョンで運用することをお勧めします。信頼性の高いデータレプリケーションを実現するには、2 つの MSK クラスターの読み取り容量と書き込み容量が同じであることが重要です。プライマリクラスターからスタンバイクラスターにデータを継続的にコピーするには、MSK レプリケーターを作成する必要があります。また、同じ AWS リージョンのクラスターのトピックにデータを書き込むようにプロデューサーを設定する必要があります。

アクティブ/パッシブ設定では、同一トピック名レプリケーション (コンソールで同じトピック名を保持) を使用して新しいレプリケーターを作成し、プライマリリージョンの MSK クラスターからセカンダリリージョンのクラスターへのデータのレプリケートを開始します。2 つの AWS リージョンで重複するプロデューサーとコンシューマーのセットを操作することをお勧めします。各リージョンは、ブートストラップ文字列を使用して独自のリージョンのクラスターに接続されます。これにより、ブートストラップ文字列の変更が不要になるため、フェイルオーバープロセスが簡素化されます。コンシューマーが中断した場所の近くから読み込みできるようにするには、ソースクラスターとターゲットクラスターのコンシューマーが同じコンシューマーグループ ID を持つ必要があります。

MSK レプリケーターに同一トピック名レプリケーション (コンソールで同じトピック名を保持) を使用した場合、トピックは対応するソーストピックと同じ名前でレプリケートされます。

ターゲットクラスターのクライアントに対してクラスターレベルの設定とアクセス許可を設定することをお勧めします。アクセスコントロールリストをコピーするオプションを選択した場合、MSK Replicator によってトピックレベルの設定とリテラル読み込み ACL が自動的にコピーされるため、ユーザーによる設定は必要ありません。「メタデータレプリケーション」を参照してください。

セカンダリ AWS リージョンへのフェイルオーバー

Amazon CloudWatch を使用して、セカンダリ AWS リージョンのレプリケーションレイテンシーをモニタリングすることをお勧めします。プライマリ AWS リージョンのサービスイベント中に、レプリケーションのレイテンシーが突然増加する可能性があります。レイテンシーが増加し続ける場合は、 AWS Service Health Dashboard を使用して、プライマリ AWS リージョンのサービスイベントを確認します。イベントがある場合は、セカンダリ AWS リージョンにフェイルオーバーできます。

セカンダリ AWS リージョンへの計画的なフェイルオーバーを実行する

計画的なフェイルオーバーを実行して、ソース MSK クラスターがあるプライマリ AWS リージョンの予期しないイベントに対してアプリケーションの回復性をテストできます。計画的なフェイルオーバーによってデータが失われることはありません。

同一トピック名レプリケーション設定を使用している場合は、次の手順に従います。

1. ソースクラスターに接続しているすべてのプロデューサーとコンシューマーをシャットダウンします。
2. 同一トピック名レプリケーション (コンソールで同じトピック名を保持) を使用して新しい MSK レプリケーターを作成し、セカンダリリージョンの MSK クラスターからプライマリリージョンの MSK クラスターにデータをレプリケートします。これは、セカンダリリージョンに書き込むデータをプライマリリージョンにコピーバックするために必要です。これにより、予期しないイベントが終了した後にプライマリリージョンにフェイルバックできるようになります。
3. セカンダリ AWS リージョンでターゲットクラスターに接続されたプロデューサーとコンシューマーを起動します。

プレフィックス付きトピック名設定を使用している場合は、次の手順に従ってフェイルオーバーします。

1. ソースクラスターに接続しているすべてのプロデューサーとコンシューマーをシャットダウンします。
2. 新しい MSK レプリケーターを作成して、セカンダリリージョンの MSK クラスターからプライマリリージョンの MSK クラスターにデータをレプリケートします。これは、セカンダリリージョンに書き込むデータをプライマリリージョンにコピーバックするために必要です。これにより、予期しないイベントが終了した後にプライマリリージョンにフェイルバックできるようになります。
3. セカンダリ AWS リージョンのターゲットクラスターでプロデューサーを起動します。
4. アプリケーションのメッセージ順序要件に応じて、次のいずれかのタブのステップを実行します。

   No message ordering

   アプリケーションでメッセージの順序付けが必要ない場合は、ワイルドカード演算子 (など) を使用して、ローカルトピック (トピックなど) とレプリケートされたトピック (な

ど<sourceKafkaClusterAlias>.topic) の両方から読み取るセカンダリ AWS リージョンでコンシューマーを起動します.*topic。

Message ordering

アプリケーションでメッセージの順序付けが必要な場合は、ターゲットクラスター上のレプリケートされたトピック (<sourceKafkaClusterAlias>.topic など) のコンシューマーのみを起動し、ローカルトピック (topic など) のコンシューマーは起動しません。

5. ターゲット MSK クラスター上のレプリケートされたトピックのすべてのコンシューマーがすべてのデータの処理を終了して、コンシューマーラグが 0 になり、処理されたレコード数も 0 になるのを待ちます。次に、ターゲットクラスター上のレプリケートされたトピックのコンシューマーを停止します。この時点で、ソース MSK クラスターからターゲット MSK クラスターにレプリケートされたすべてのレコードが消費されました。
6. ターゲット MSK クラスター上のローカルトピック (topic など) のコンシューマーを起動します。

セカンダリ AWS リージョンへの計画外のフェイルオーバーを実行する

ソース MSK クラスターを持つプライマリ AWS リージョンにサービスイベントがあり、ターゲット MSK クラスターを持つセカンダリリージョンにトラフィックを一時的にリダイレクトする場合は、予期しないフェイルオーバーを実行できます。MSK Replicator はデータを非同期的にレプリケートするため、計画外のフェイルオーバーが発生すると、データが失われる可能性があります。??? のメトリクスを使用して、メッセージの遅延を追跡できます。

同一トピック名レプリケーション設定 (コンソールで同じトピック名を保持) を使用している場合は、次の手順に従います。

1. プライマリリージョンのソース MSK クラスターに接続しているすべてのプロデューサーとコンシューマーをシャットダウンしてみます。この操作は、そのリージョンの障害が原因で成功しない可能性があります。
2. セカンダリ AWS リージョンのターゲット MSK クラスターに接続するプロデューサーとコンシューマーを起動して、フェイルオーバーを完了します。MSK Replicator は、読み取り ACL やコンシューマーグループのオフセットなどのメタデータもレプリケートするため、プロデューサーとコンシューマーはフェイルオーバー前に中断した場所からシームレスに処理を再開します。

PREFIX トピック名設定を使用している場合は、次の手順に従ってフェイルオーバーします。

1. プライマリリージョンのソース MSK クラスターに接続しているすべてのプロデューサーとコンシューマーをシャットダウンしてみます。この操作は、そのリージョンの障害が原因で成功しない可能性があります。
2. セカンダリ AWS リージョンのターゲット MSK クラスターに接続するプロデューサーとコンシューマーを起動して、フェイルオーバーを完了します。MSK Replicator は、読み取り ACL やコンシューマーグループのオフセットなどのメタデータもレプリケートするため、プロデューサーとコンシューマーはフェイルオーバー前に中断した場所からシームレスに処理を再開します。
3. アプリケーションのメッセージ順序要件に応じて、次のいずれかのタブのステップを実行します。

   No message ordering

   アプリケーションでメッセージの順序付けが必要ない場合は、ワイルドカード演算子 (など`topic`) を使用して、ローカルトピック (など) とレプリケートされたトピック (など`<sourceKafkaClusterAlias>.topic`) の両方から読み取るターゲット AWS リージョンでコンシューマーを起動します`.*topic`。

   Message ordering

   1. ターゲットクラスター上のレプリケートされたトピック (`<sourceKafkaClusterAlias>.topic` など) のコンシューマーのみを起動し、ローカルトピック (`topic` など) のコンシューマーは起動しません。
   2. ターゲット MSK クラスター上のレプリケートされたトピックのすべてのコンシューマーがすべてのデータの処理を終了して、オフセットラグが 0 になり、処理されたレコード数も 0 になるのを待ちます。次に、ターゲットクラスター上のレプリケートされたトピックのコンシューマーを停止します。この時点で、ソース MSK クラスターからターゲット MSK クラスターにレプリケートされたすべてのレコードが消費されました。
   3. ターゲット MSK クラスター上のローカルトピック (`topic` など) のコンシューマーを起動します。
4. プライマリリージョンでサービスイベントが終了したら、新しい MSK レプリケーターを作成し、レプリケーターの開始位置を earliest に設定して、セカンダリリージョンの MSK クラスターからプライマリリージョンの MSK クラスターにデータをレプリケートします。これは、セカンダリリージョンに書き込むデータをプライマリリージョンにコピーバックするために必要です。これにより、サービスイベントが終了した後にプライマリリージョンにフェイルバックできるようになります。レプリケーターの開始位置を earliest に設定しない場合、プライマリリー

ジョンのサービスイベント中にセカンダリリージョンのクラスターに生成したデータは、プライマリリージョンのクラスターにコピーされません。

プライマリ AWS リージョンへのフェイルバックの実行

その AWS リージョンのサービスイベントが終了した後、プライマリリージョンにフェイルバックできます。

同一トピック名レプリケーション設定を使用している場合は、次の手順に従います。

1. セカンダリクラスターをソース、プライマリクラスターをターゲットとし、開始位置を earliest に設定して、同一トピック名レプリケーション (コンソールで同じトピック名を保持) を使用して、新しい MSK レプリケーターを作成します。

   これにより、プライマリリージョンへのフェイルオーバー後にセカンダリクラスターに書き込まれたすべてのデータをコピーするプロセスが開始されます。
2. Amazon CloudWatch で、新しいレプリケーターの `MessageLag` メトリクスを、すべてのデータがセカンダリからプライマリにレプリケートされたことを示す `0` に達するまでモニタリングします。
3. すべてのデータがレプリケートされたら、セカンダリクラスターに接続しているすべてのプロデューサーを停止し、プライマリクラスターに接続しているプロデューサーを起動します。
4. セカンダリクラスターに接続しているコンシューマーの `MaxOffsetLag` メトリクスが `0` になるまで待機し、すべてのデータを処理されたことを確認します。「コンシューマーラグをモニタリングする」を参照してください。
5. すべてのデータが処理されたら、セカンダリリージョンのコンシューマーを停止し、プライマリクラスターに接続しているコンシューマーを起動してフェイルバックを完了します。
6. セカンダリクラスターからプライマリにデータをレプリケートする最初のステップで作成したレプリケーターを削除します。
7. プライマリクラスターからセカンダリクラスターにデータをコピーしている既存のレプリケーターのステータスが「RUNNING」で、Amazon CloudWatch の `ReplicatorThroughput` メトリクスが `0` であることを確認します。

   フェイルバックの開始位置を Earliest として新しいレプリケーターを作成すると、そのレプリケーターはセカンダリクラスターのトピック内のすべてのデータの読み込みを開始します。データ保持設定によっては、トピックにソースクラスターから取得されたデータが含まれる場合があります。MSK Replicator はこれらのメッセージを自動的にフィルタリングします

が、セカンダリクラスター内のすべてのデータに対してデータ処理および転送料金が発生します。ReplicatorBytesInPerSec を使用して、レプリケーターによって処理されたデータの合計を追跡できます。「MSK レプリケーターのメトリクス」を参照してください。

プレフィックス付きトピック名設定を使用している場合は、次のステップに従います。

フェイルバックステップは、セカンダリリージョンのクラスターからプライマリリージョンのクラスターへのレプリケーションがキャッチアップし、Amazon CloudWatch の MessageLag メトリクスが 0 に近づいてから開始する必要があります。計画的なフェイルバックによってデータが失われることはありません。

1. セカンダリリージョンの MSK クラスターに接続しているすべてのプロデューサーとコンシューマーをシャットダウンします。
2. アクティブ/パッシブトポロジの場合は、セカンダリリージョンのクラスターからプライマリリージョンにデータをレプリケートしているレプリケーターを削除します。アクティブ/アクティブトポロジの場合は、レプリケーターを削除する必要はありません。
3. プライマリリージョンの MSK クラスターに接続しているプロデューサーを起動します。
4. アプリケーションのメッセージ順序要件に応じて、次のいずれかのタブのステップを実行します。

   No message ordering

   アプリケーションでメッセージの順序付けが必要ない場合は、ワイルドカード演算子 (など`topic`) を使用して、ローカルトピック (など) とレプリケートされたトピック (など`<sourceKafkaClusterAlias>.topic`) の両方から読み取るプライマリ AWS リージョンでコンシューマーを起動します`.*topic`。ローカルトピック (topic など) のコンシューマーは、フェイルオーバー前に消費した最後のオフセットから再開します。フェイルオーバー前に未処理のデータがあった場合は、この時点で処理されます。計画的なフェイルオーバーの場合は、そのようなレコードはないはずです。

   Message ordering

   1. プライマリリージョンのレプリケートされたトピック (`<sourceKafkaClusterAlias>.topic` など) のコンシューマーのみを起動し、ローカルトピック (`topic` など) のコンシューマーは起動しません。
   2. プライマリリージョンのクラスター上のレプリケートされたトピックのすべてのコンシューマーがすべてのデータの処理を終了して、オフセットラグが 0 になり、処理されたレコード数も 0 になるのを待ちます。次に、プライマリリージョンのクラスター上のレプ

リケートされたトピックのコンシューマーを停止します。この時点で、フェイルオーバー後にセカンダリリージョンで生成されたすべてのレコードが、プライマリリージョンで消費されました。

3. プライマリリージョンのクラスター上のローカルトピック (topic など) のコンシューマーを起動します。

5. ReplicatorThroughput とレイテンシーメトリクスを使用して、プライマリリージョンのクラスターからセカンダリリージョンのクラスターへの既存のレプリケーターが RUNNING 状態にあり、期待どおりに動作していることを確認します。

MSK Replicator を使用したアクティブ/アクティブ設定の作成

両方の MSK クラスターが読み込みと書き込みをアクティブに処理するアクティブ/アクティブ設定を作成する場合は、プレフィックス付きトピック名レプリケーション (コンソールでトピック名にプレフィックスを追加) が設定された MSK レプリケーターを使用することをお勧めします。ただし、レプリケートされたトピックを読み込むようにコンシューマーを再設定する必要があります。

次のステップを実行して、ソース MSK クラスター A とターゲット MSK クラスター B の間にアクティブ/アクティブトポロジを設定します。

1. MSK クラスター A をソース、MSK クラスター B をターゲットとして MSK レプリケーターを作成します。
2. 上記の MSK レプリケーターが正常に作成されたら、クラスター B をソース、クラスター A をターゲットとしてレプリケーターを作成します。
3. 2 組のプロデューサーを作成し、それぞれがプロデューサーと同じリージョン内のクラスターのローカルトピック (「topic」など) に同時にデータを書き込みます。
4. コンシューマーのセットを 2 つ作成します。各コンシューマーは、コンシューマーと同じ AWS リージョンの MSK クラスターからワイルドカードサブスクリプション (「.*topic」など) を使用してデータを読み込みます。これにより、コンシューマーはローカルトピック (topic など) からリージョン内でローカルに生成されたデータを自動的に読み取るだけでなく、他のリージョンからレプリケートされたデータ (トピック内でプレフィックス <sourceKafkaClusterAlias>.topic が付いているトピック) も自動的に読み取ります。これら 2 組のコンシューマーは、異なるコンシューマーグループ ID を持つ必要があります。これにより、MSK レプリケーターがコンシューマーグループのオフセットを他のクラスターにコピーしたときに、それらのオフセットが上書きされなくなります。

クライアントの再設定を避けたい場合は、プレフィックス付きトピック名レプリケーション (コンソールでトピック名にプレフィックスを追加) の代わりに、同一トピック名レプリケーション (コンソールで同じトピック名を保持) を使用して MSK レプリケーターを作成し、アクティブ/アクティブ設定を作成してください。ただし、レプリケーターごとに追加のデータ処理および転送料金が発生します。これは、各レプリケーターが通常の 2 倍のデータ量 (レプリケーションのために 1 回、無限ループを防ぐために 1 回) を処理する必要があるためです。`ReplicatorBytesInPerSec` メトリクスを使用して、各レプリケーターによって処理されるデータの合計量を追跡できます。「レプリケーションのモニタリング」を参照してください。このメトリクスには、ターゲットクラスターにレプリケートされたデータと、データが元のトピックと同じトピックにコピーされないように MSK Replicator によってフィルタリングされたデータが含まれます。

> **Note**
>
> 同一トピック名レプリケーション (コンソールで同じトピック名を保持) を使用してアクティブ/アクティブトポロジを設定している場合は、トピックを削除した後に少なくとも 30 秒待ってから、同じ名前のトピックを再作成します。この待機期間は、重複したメッセージがソースクラスターにレプリケートされるのを防ぐのに役立ちます。コンシューマーは、ダウンストリームに影響を与えることなく、重複したメッセージを再処理できる必要があります。「マルチリージョン Apache Kafka アプリケーションを構築する際の考慮事項」を参照してください。

MSK Replicator を使用して、1 つの Amazon MSK クラスターから別のクラスターに移行する

クラスターの移行には同一トピック名レプリケーションを使用できますが、コンシューマーはダウンストリームに影響を与えることなく重複したメッセージを処理できる必要があります。これは、MSK Replicator が少なくとも 1 回のレプリケーションを提供するため、まれに重複したメッセージが発生する可能性があるためです。コンシューマーがこの要件を満たしている場合は、以下の手順に従ってください。

1. レプリケーターの開始位置を Earliest に設定し、同一トピック名レプリケーション (コンソールで同じトピック名を保持) を使用して、古いクラスターから新しいクラスターにデータをレプリケートするレプリケーターを作成します。
2. 新しいクラスターに対してクラスターレベルの設定とアクセス許可を設定します。MSK Replicator によってトピックレベルの設定と「リテラル」読み取り ACL が自動的にコピーされるため、ユーザーによる設定は必要ありません。

3. Amazon CloudWatch で、MessageLag メトリクスを、すべてのデータがレプリケートされたことを示す 0 に達するまでモニタリングします。
4. すべてのデータがレプリケートされたら、プロデューサーが古いクラスターにデータを書き込まないようにします。
5. これらのプロデューサーを再設定して新しいクラスターに接続し、起動します。
6. 古いクラスターからデータを読み込んでいるコンシューマーの MaxOffsetLag メトリクスを、既存のすべてのデータが処理されたことを示す 0 になるまでモニタリングします。
7. 古いクラスターに接続しているコンシューマーを停止します。
8. コンシューマーを再設定して新しいクラスターに接続し、起動します。

セルフマネージド型の MirrorMaker2 から MSK Replicator に移行する

MirrorMaker (MM2) から MSK Replicator に移行するには、次の手順に従います。

1. ソース Amazon MSK クラスターに書き込みを行っているプロデューサーを停止します。
2. MM2 がソースクラスターのトピック上のすべてのメッセージをレプリケートできるようにします。ソース MSK クラスターで MM2 コンシューマーのコンシューマーラグをモニタリングし、すべてのデータがレプリケートされたタイミングを判断します。
3. 開始位置を Latest に設定し、トピック名設定を IDENTICAL (コンソールで同じトピック名のレプリケーション) に設定して、新しいレプリケーターを作成します。
4. レプリケーターが RUNNING 状態になったら、プロデューサーによるソースクラスターへの書き込みを再開できます。

MSK Replicator のトラブルシューティング

次の情報は、MSK レプリケーターに関する問題のトラブルシューティングに役立ちます。他の Amazon MSK 機能に関する問題の解決情報については、「Amazon MSK クラスターをトラブルシューティングする」を参照してください。AWS re:Post に問題を投稿することもできます。

MSK レプリケーターの状態が CREATING から FAILED に変わります。

MSK レプリケーターの作成に失敗する一般的な原因には、次のようなものがあります。

1. ターゲットクラスターのセクションでレプリケーターの作成用に指定したセキュリティグループに、ターゲットクラスターのセキュリティグループへのトラフィックを許可するアウトバウンドルールがあることを確認します。また、ターゲットクラスターのセキュリティグループに、ターゲットクラスターのセクションでレプリケーターの作成用に指定するセキュリティグループからのトラフィックを受け入れるインバウンドルールがあることを確認します。「ターゲットクラスターの選択」を参照してください。
2. クロスリージョンレプリケーション用にレプリケーターを作成する場合は、ソースクラスターで IAM アクセスコントロール認証方法に対してマルチ VPC 接続が有効になっていることを確認します。「単一リージョンの Amazon MSK マルチ VPC プライベート接続」を参照してください。また、MSK レプリケーターがソースクラスターに接続できるように、ソースクラスターでクラスターポリシーが設定されていることも確認します。「Amazon MSK ソースクラスターを準備する」を参照してください。
3. MSK レプリケーターの作成時に指定した IAM ロールに、ソースクラスターとターゲットクラスターに対する読み取りと書き込みに必要なアクセス許可があることを確認します。また、IAM ロールに、トピックに書き込むアクセス許可があることも確認します。「レプリケータの設定とアクセス許可を設定します。」を参照してください。
4. ネットワーク ACL が MSK レプリケーターとソースクラスターおよびターゲットクラスターの間の接続をブロックしていないことを確認します。
5. MSK レプリケーターが接続を試みたときに、ソースクラスターまたはターゲットクラスターが完全には利用できない可能性があります。これは、負荷、ディスク使用量、または CPU 使用率が高すぎるために、レプリケーターがブローカーに接続できなくなったことが原因である可能性があります。ブローカーの問題を修正し、レプリケーターの作成を再試行してください。

上記の検証を実行したら、MSK レプリケーターを再度作成します。

MSK レプリケーターが CREATING 状態でスタックしているように見える

MSK レプリケーターの作成には、最大 30 分かかる場合があります。30 分間待ってから、レプリケーターの状態を再度確認してください。

MSK レプリケーターがデータをレプリケートしていない、または一部のデータしかレプリケートしていない

データレプリケーションの問題をトラブルシューティングするには、次のステップを実行します。

1. MSK Replicator が提供する Amazon CloudWatch の AuthError メトリクスを使用して、レプリケーターで認証エラーが発生していないことを確認します。このメトリクスが 0 より大きい場合

は、レプリケーター用に指定した IAM ロールのポリシーが有効かどうかと、クラスターのアクセス許可に対して拒否アクセス許可が設定されていないかを確認します。clusterAlias ディメンションに基づいて、ソースクラスターまたはターゲットクラスターで認証エラーが発生しているかどうかを識別できます。

2. ソースクラスターとターゲットクラスターで問題が発生していないことを確認します。レプリケーターがソースクラスターまたはターゲットクラスターに接続できない可能性があります。これは、接続数が多すぎる、ディスクの容量が満杯である、または CPU 使用率が高いことが原因で発生する可能性があります。
3. Amazon CloudWatch の KafkaClusterPingSuccessCount メトリクスを使用して、ソースクラスターとターゲットクラスターに MSK Replicator からアクセスできることを確認します。clusterAlias ディメンションに基づいて、ソースクラスターまたはターゲットクラスターで認証エラーが発生しているかどうかを識別できます。このメトリクスが 0 またはデータポイントがない場合、接続は異常です。MSK レプリケーターがクラスターへの接続に使用しているネットワークと IAM ロールのアクセス許可を確認する必要があります。
4. Amazon CloudWatch の ReplicatorFailure メトリクスを使用して、トピックレベルのアクセス許可の欠落が原因でレプリケーターに障害が発生していないことを確認します。このメトリクスが 0 より大きい場合は、トピックレベルのアクセス許可に対して指定した IAM ロールを確認してください。
5. レプリケーターの作成時に許可リストに指定した正規表現が、レプリケートするトピックの名前と一致することを確認します。また、拒否リストの正規表現が原因でトピックがレプリケートから除外されていないことも確認します。
6. レプリケーターがターゲットクラスターの新しいトピックまたはトピックパーティションを検出して作成するまでに最大 30 秒かかる場合があります。レプリケーターの開始位置が latest (デフォルト) の場合、ターゲットクラスターでトピックが作成される前にソーストピックに生成されたメッセージはレプリケートされません。あるいは、ターゲットクラスター上のトピックの既存のメッセージをレプリケートする場合は、ソースクラスターのトピックパーティション内の最も古いオフセットからレプリケーションを開始できます。「レプリケータの設定とアクセス許可を設定します。」を参照してください。

メッセージのオフセットがターゲットクラスターとソースクラスターとで異なる

データのレプリケートの一環として、MSK Replicator はソースクラスターからのメッセージを消費し、ターゲットクラスターに生成します。これにより、ソースクラスターとターゲットクラスターでメッセージのオフセットが異なる場合があります。ただし、レプリケーターの作成中にコンシュー

マーグループのオフセットの同期をオンにした場合、MSK Replicator はメタデータをコピーしながらオフセットを自動的に変換するため、ターゲットクラスターにフェイルオーバーした後、コンシューマーはソースクラスターで中断した場所の近くから処理を再開できます。

MSK Replicator がコンシューマーグループのオフセットを同期していない、またはコンシューマーグループがターゲットクラスターに存在しない

メタデータレプリケーションの問題をトラブルシューティングするには、以下の手順を実行します。

1. データレプリケーションが期待どおりに機能していることを確認します。そうでない場合は、「MSK レプリケーターがデータをレプリケートしていない、または一部のデータしかレプリケートしていない」を参照してください。
2. レプリケーターの作成時に許可リストに指定した正規表現が、レプリケートするコンシューマーグループの名前と一致することを確認します。また、拒否リストの正規表現が原因でコンシューマーグループがレプリケーションから除外されていないことも確認します。
3. MSK Replicator がターゲットクラスターにトピックを作成したことを確認します。レプリケーターがターゲットクラスター上の新しいトピックまたはトピックパーティションを検出して作成するまでに最大 30 秒かかる場合があります。レプリケーターの開始位置が latest (デフォルト) の場合、ターゲットクラスターでトピックが作成される前にソーストピックに生成されたメッセージはレプリケートされません。ソースクラスターのコンシューマーグループが MSK Replicator によってレプリケートされていないメッセージのみを消費した場合、コンシューマーグループはターゲットクラスターにレプリケートされません。トピックがターゲットクラスターに正常に作成されると、MSK Replicator はソースクラスターに新しく書き込まれたメッセージをターゲットクラスターにレプリケートし始めます。コンシューマーグループがソースからこれらのメッセージの読み込みを開始すると、MSK Replicator はコンシューマーグループをターゲットクラスターに自動的にレプリケートします。あるいは、ターゲットクラスター上のトピックの既存のメッセージをレプリケートする場合は、ソースクラスターのトピックパーティション内の最も古いオフセットからレプリケーションを開始できます。「レプリケータの設定とアクセス許可を設定します。」を参照してください。

> **Note**
>
> MSK Replicator は、トピックパーティションの末尾に近い場所から読み込みを行っているソースクラスター上のコンシューマーのコンシューマーグループのオフセット同期を最適化します。ソースクラスターのコンシューマーグループが遅延している場合、ターゲットのコンシューマーグループの遅延はソースと比較して大きくなる可能性があります。つまり、

ターゲットクラスターへのフェイルオーバー後、コンシューマーは重複したメッセージをさらに再処理しすることになります。この遅延を小さくするには、ソースクラスターのコンシューマーがストリームの先端 (トピックパーティションの末尾) にキャッチアップするように処理を行い、消費を開始する必要があります。コンシューマーによる処理がキャッチアップすると、MSK Replicator は自動的に遅延を小さくします。

レプリケーションのレイテンシーが高い、または増加し続けている

レプリケーションレイテンシーが高くなる一般的な原因は次のとおりです。

1. ソースとターゲットの MSK クラスターに適切な数のパーティションがあることを確認します。パーティションが少なすぎたり多すぎたりすると、パフォーマンスに影響する可能性があります。パーティション数の選択に関するガイダンスについては、「MSK レプリケーターの使用に関するベストプラクティス」を参照してください。次の表は、MSK レプリケーターで必要なスループットを実現するために推奨される最小パーティション数を示しています。

   スループットと推奨される最小パーティション数

スループット (MB/秒)	必要な最小パーティション数
50	167
100	334
250	833
500	1666
1,000	3333

2. ソースとターゲットの MSK クラスターに、レプリケーショントラフィックをサポートするのに十分な読み取りおよび書き込み容量があることを確認します。MSK レプリケーターは、ソースクラスター (エグレス) のコンシューマーとして、またターゲットクラスター (イングレス) のプロデューサーとして機能します。そのため、クラスター上の他のトラフィックに加えてレプリケーショントラフィックもサポートできるように、クラスター容量をプロビジョニングする必要があります。MSK クラスターのサイズ設定のガイダンスについては、「???」を参照してください。
3. レプリケーションのレイテンシーは、レプリケート元とレプリケート先の AWS リージョンのペアが異なる MSK クラスターでは、クラスター間の地理的距離によって異なる場合があります。例

えば、欧州 (アイルランド) リージョンと欧州 (ロンドン) リージョンのクラスター間でレプリケーションを行う場合、欧州 (アイルランド) リージョンとアジアパシフィック (シドニー) リージョンのクラスター間のレプリケーションと比較して、レプリケーションレイテンシーは一般的に低くなります。

4. ソースまたはターゲットのクラスターで過度に厳しいクォータが設定されているためにレプリケーターがスロットルされていないことを確認してください。MSK Replicator が提供する Amazon CloudWatch の ThrottleTime メトリクスを使用して、ソース/ターゲットクラスター上のブローカーによってリクエストがスロットルされた平均時間 (ミリ秒単位) を確認できます。このメトリクスが 0 より大きい場合は、レプリケーターがキャッチアップできるように、Kafka のクォータを調整してスロットリングを減らす必要があります。レプリケーターの Kafka クォータの管理については、「Kafka クォータを使用した MSK レプリケータースループットの管理」を参照してください。
5. AWS リージョンが劣化すると、ReplicationLatency と MessageLag が増加する可能性があります。AWS Service Health Dashboard を使用して、プライマリ MSK クラスターが配置されているリージョンで MSK サービスイベントがないかを確認します。サービスイベントが発生した場合は、一時的にアプリケーションの読み取りと書き込みを別のリージョンにリダイレクトできます。

ReplicatorFailure メトリクスを使用した MSK レプリケーターの障害のトラブルシューティング

ReplicatorFailure メトリクスは、MSK レプリケーターのレプリケーション問題をモニタリングおよび検出するのに役立ちます。このメトリクスの 0 以外の値は、通常、レプリケーションの失敗の問題を示します。これは、次の要因が原因である可能性があります。

- メッセージサイズの制限
- タイムスタンプ範囲違反
- バッチサイズの問題を記録する

ReplicatorFailure メトリクスがゼロ以外の値を報告した場合は、以下の手順に従って問題をトラブルシューティングします。

> **Note**
>
> このメトリクスの詳細については、「MSK レプリケーターのメトリクス」を参照してください。

1. ターゲット MSK クラスターに接続でき、Apache Kafka CLI ツールが設定されているクライアントを設定します。クライアントと Kafka CLI ツールの設定については、「」を参照してくださいAmazon MSK プロビジョンドクラスターに接続する。
2. https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ で Amazon MSK コンソールを開きます。

   次に、以下の操作を実行します。

   a. MSK レプリケーターとターゲット MSK クラスターの ARNs を取得します。

   b. ターゲット MSK クラスターのブローカーエンドポイントを取得します。これらのエンドポイントは、次のステップで使用します。
3. 次のコマンドを実行して、前のステップで取得した MSK レプリケーター ARN とブローカーエンドポイントをエクスポートします。

   次の例で使用されている <*ReplicatorARN*> 、 <*BootstrapServerString*> 、 <*ConsumerConfigFile*> のプレースホルダー値を実際の値に置き換えてください。

   ```
   export TARGET_CLUSTER_SERVER_STRING=<BootstrapServerString>
   ```

   ```
   export REPLICATOR_ARN=<ReplicatorARN>
   ```

   ```
   export CONSUMER_CONFIG_FILE=<ConsumerConfigFile>
   ```
4. <*path-to-your-kafka-installation*>/bin ディレクトリで、次の操作を行います。

   a. 次のスクリプトを保存し、 という名前を付けます**query-replicator-failure-message.sh**。

   ```
   #!/bin/bash

   # Script: Query MSK Replicator Failure Message
   ```

```
# Description: This script queries exceptions from AWS MSK Replicator status
 topics
# It takes a replicator ARN and bootstrap server as input and searches for
 replicator exceptions
# in the replicator's status topic, formatting and displaying them in a
 readable manner
#
# Required Arguments:
#   --replicator-arn: The ARN of the AWS MSK Replicator
#   --bootstrap-server: The Kafka bootstrap server to connect to
#   --consumer.config: Consumer config properties file
# Usage Example:
#   ./query-replicator-failure-message.sh ./query-replicator-failure-message.sh
 --replicator-arn <replicator-arn> --bootstrap-server <bootstrap-server> --
consumer.config <consumer.config>

print_usage() {
  echo "USAGE: $0 ./query-replicator-failure-message.sh --replicator-arn
 <replicator-arn> --bootstrap-server <bootstrap-server> --consumer.config
 <consumer.config>"
  echo "--replicator-arn <String: MSK Replicator ARN>      REQUIRED: The ARN of
 AWS MSK Replicator."
  echo "--bootstrap-server <String: server to connect to>  REQUIRED: The Kafka
 server to connect to."
  echo "--consumer.config <String: config file>            REQUIRED: Consumer
 config properties file."
  exit 1
}

# Initialize variables
replicator_arn=""
bootstrap_server=""
consumer_config=""

# Parse arguments
while [[ $# -gt 0 ]]; do
  case "$1" in
    --replicator-arn)
      if [ -z "$2" ]; then
        echo "Error: --replicator-arn requires an argument."
        print_usage
      fi
      replicator_arn="$2"; shift 2 ;;
    --bootstrap-server)
```

```
      if [ -z "$2" ]; then
        echo "Error: --bootstrap-server requires an argument."
        print_usage
      fi
      bootstrap_server="$2"; shift 2 ;;
    --consumer.config)
      if [ -z "$2" ]; then
        echo "Error: --consumer.config requires an argument."
        print_usage
      fi
      consumer_config="$2"; shift 2 ;;
    *) echo "Unknown option: $1"; print_usage ;;
  esac
done

# Check for required arguments
if [ -z "$replicator_arn" ] || [ -z "$bootstrap_server" ] || [ -z
 "$consumer_config" ]; then
  echo "Error: --replicator-arn, --bootstrap-server, and --consumer.config are
 required."
  print_usage
fi

# Extract replicator name and suffix from ARN
replicator_arn_suffix=$(echo "$replicator_arn" | awk -F'/' '{print $NF}')
replicator_name=$(echo "$replicator_arn" | awk -F'/' '{print $(NF-1)}')
echo "Replicator name: $replicator_name"

# List topics and find the status topic
topics=$(./kafka-topics.sh --command-config client.properties --list --
bootstrap-server "$bootstrap_server")
status_topic_name="__amazon_msk_replicator_status_${replicator_name}_
${replicator_arn_suffix}"

# Check if the status topic exists
if echo "$topics" | grep -Fq "$status_topic_name"; then
  echo "Found replicator status topic: '$status_topic_name'"
  ./kafka-console-consumer.sh --bootstrap-server "$bootstrap_server" --
consumer.config "$consumer_config" --topic "$status_topic_name" --from-
beginning | stdbuf -oL grep "Exception" | stdbuf -oL sed -n 's/.*Exception:\(.*
\) Topic: \([^,]*\), Partition: \([^\]*\).*/ReplicatorException:\1 Topic: \2,
 Partition: \3/p'
else
  echo "No topic matching the pattern '$status_topic_name' found."
```

```
fi
```

b. このスクリプトを実行して、MSK レプリケーターの失敗メッセージをクエリします。

```
<path-to-your-kafka-installation>/bin/query-replicator-failure-message.sh --
replicator-arn $REPLICATOR_ARN --bootstrap-server $TARGET_CLUSTER_SERVER_STRING
 --consumer.config $CONSUMER_CONFIG_FILE
```

このスクリプトは、例外メッセージと影響を受けるトピックパーティションを含むすべてのエラーを出力します。この例外情報を使用して、「」で説明されているように障害を軽減できますMSK レプリケーターの一般的な障害とその解決策。トピックには過去の失敗メッセージがすべて含まれているため、最後のメッセージを使用して調査を開始します。失敗メッセージの例を次に示します。

```
ReplicatorException: The request included a message larger than the max message
 size the server will accept. Topic: test, Partition: 1
```

MSK レプリケーターの一般的な障害とその解決策

次のリストでは、発生する可能性のある MSK レプリケーターの障害とその軽減方法について説明します。

max.request.size より大きいメッセージサイズ

原因

この障害は、個々のメッセージサイズが 10 MB を超えたために MSK レプリケーターがデータのレプリケートに失敗した場合に発生します。デフォルトでは、MSK レプリケーターは最大 10 MB のサイズのメッセージをレプリケートします。

この失敗メッセージタイプの例を次に示します。

```
ReplicatorException: The message is 20635370 bytes when serialized which is larger
 than 10485760, which is the value of the max.request.size configuration. Topic:
 test, Partition: 1
```

ソリューション

トピック内の個々のメッセージサイズを減らします。できない場合は、以下の手順に従って制限の引き上げをリクエストしてください。

サーバーが受け入れるメッセージの最大サイズより大きいメッセージサイズ

原因

このエラーは、メッセージサイズがターゲットクラスターの最大メッセージサイズを超えた場合に発生します。

この失敗メッセージタイプの例を次に示します。

```
ReplicatorException: The request included a message larger than the max message size
 the server will accept. Topic: test, Partition: 1
```

ソリューション

ターゲットクラスターまたは対応するターゲットクラストピックmax.message.bytesの設定を増やします。ターゲットクラスターmax.message.bytesの設定を、最大非圧縮メッセージサイズに合わせて設定します。これを行う方法については、「max.message.bytes」を参照してください。

タイムスタンプが範囲外です

原因

このエラーは、個々のメッセージのタイムスタンプがターゲットクラスターの許容範囲外であるために発生します。

この失敗メッセージタイプの例を次に示します。

```
ReplicatorException: Timestamp 1730137653724 of message with offset 0 is out of
 range. The timestamp should be within [1730137892239, 1731347492239] Topic: test,
 Partition: 1
```

ソリューション

ターゲットクラスターmessage.timestamp.before.max.msの設定を更新して、古いタイムスタンプのメッセージを許可します。これを行う方法については、message.timestamp.before.max.ms を参照してください。

レコードバッチが大きすぎる

原因

このエラーは、レコードバッチサイズがターゲットクラスターの トピックに設定されたセグメントサイズを超えているために発生します。MSK レプリケーターは、最大バッチサイズ 1 MB をサポートします。

この失敗メッセージタイプの例を次に示します。

```
ReplicatorException: The request included message batch larger than the configured
 segment size on the server. Topic: test, Partition: 1
```

ソリューション

レプリケーターがエラーなしで続行するには、ターゲットクラスターの segment.bytes 設定がバッチサイズ (1 MB) 以上である必要があります。ターゲットクラスターの segment.bytes を少なくとも 1048576 (1 MB) に更新します。これを行う方法については、「segment.bytes」を参照してください。

> **Note**
>
> これらのソリューションを適用した後に ReplicatorFailure メトリクスがゼロ以外の値を引き続き出力する場合は、メトリクスがゼロを出力するまでトラブルシューティングプロセスを繰り返します。

MSK レプリケーターの使用に関するベストプラクティス

このセクションでは、Amazon MSK Replicator を使用する際の一般的なベストプラクティスと実装戦略について説明します。

トピック

- Kafka クォータを使用した MSK レプリケータースループットの管理
- クラスターの保持期間の設定

Kafka クォータを使用した MSK レプリケータースループットの管理

MSK レプリケーターはソースクラスターのコンシューマーとして機能するため、レプリケーションによってソースクラスター上の他のコンシューマーがスロットルされる可能性があります。スロットリング量は、ソースクラスターの読み取り容量と、レプリケートするデータのスループットによって

異なります。必要な容量を計算する際には、ソースクラスターとターゲットクラスターに同じ容量をプロビジョニングし、レプリケーションのスループットを考慮に入れることが推奨されます。

また、ソースクラスターとターゲットクラスターのレプリケーターに Kafka クォータを設定して、MSK レプリケーターが使用できる容量を制御することもできます。ネットワーク帯域幅クォータが推奨されます。ネットワーク帯域幅クォータは、クォータを共有する 1 つ以上のクライアントに対して、バイトレートのしきい値 (1 秒あたりのバイト数) を定義します。このクォータはブローカーごとに定義されます。

クォータを適用するには、次のステップを実行します。

1. ソースクラスターのブートストラップサーバー文字列を取得します。「Amazon MSK クラスターのブートストラップブローカーを取得する」を参照してください。
2. MSK レプリケーターが使用するサービス実行ロール (SER) を取得します。これは `CreateReplicator` リクエストに使用した SER です。既存のレプリケーターからの DescribeReplicator レスポンスから SER をプルすることもできます。
3. Kafka CLI ツールを使用して、ソースクラスターに対して次のコマンドを実行します。

```
./kafka-configs.sh --bootstrap-server <source-cluster-bootstrap-server> --alter --
add-config 'consumer_byte_
rate=<quota_in_bytes_per_second>' --entity-type users --entity-name
 arn:aws:sts::<customer-account-id>:assumed-role/<ser-role-name>/<customer-account-
id> --command-config <client-properties-for-iam-auth></programlisting>
```

4. 上記のコマンドを実行した後、`ReplicatorThroughput` メトリクスが設定したクォータを超えていないことを確認します。

複数の MSK レプリケーター間でサービス実行ロールを再利用する場合、それらはすべてこのクォータの対象となることに注意してください。レプリケーターごとに個別のクォータを維持する必要がある場合は、個別のサービス実行ロールを使用します。

MSK IAM 認証とクォータの使用に関する詳細については、「Multi-tenancy Apache Kafka clusters in Amazon MSK with IAM access control and Kafka Quotas – Part 1」を参照してください。

> **Warning**
>
> consumer_byte_rate を極端に低く設定すると、MSK レプリケーターが予期しない動作をする可能性があります。

クラスターの保持期間の設定

MSK プロビジョンドクラスターとサーバーレスクラスターのログの保持期間を設定できます。推奨される保持期間は 7 日間です。「クラスター設定の変更」または「サポートされている MSK サーバーレスクラスター設定」を参照してください。

MSK の統合

このセクションでは、Amazon MSK と統合される AWS 機能について説明します。

トピック

- Amazon MSK 用の Amazon Athena コネクタ
- Amazon MSK の Amazon Redshift ストリーミングデータインジェスト
- Amazon MSK の Firehose 統合
- Amazon MSK コンソールから Amazon EventBridge Pipes にアクセスする
- MSK Express ブローカーと MSK Serverless での Kafka Streams の使用
- リアルタイムベクトル埋め込みブループリント

Amazon MSK 用の Amazon Athena コネクタ

Amazon MSK 用の Amazon Athena コネクタを使用すると、Amazon Athena で Apache Kafka トピックに対する SQL クエリを実行できるようになります。このコネクタを使用すると、Apache Kafka トピックをテーブルとして、メッセージを Athena の行として表示できます。

詳細については、「Amazon Athena ユーザーガイド」の「Amazon Athena MSK コネクタ」を参照してください。

Amazon MSK の Amazon Redshift ストリーミングデータインジェスト

Amazon Redshift は、Amazon MSK からのストリーミング取り込みをサポートしています。Amazon Redshift ストリーミング取り込み機能を使用すると、Amazon MSK から Amazon Redshift マテリアライズドビューに低レイテンシーかつ高速でストリーミングデータを取り込むことができます。Amazon S3 にデータをステージングする必要がないため、Amazon Redshift はより低いレイテンシー、より低いストレージコストでストリーミングデータを取り込むことができます。SQL ステートメントを使用して Amazon Redshift クラスターに Amazon Redshift ストリーミング取り込みを設定し、Amazon MSK トピックを認証して接続することができます。

詳細については、「Amazon Redshift データベース開発者ガイド」の「ストリーミング取り込み」を参照してください。

Amazon MSK の Firehose 統合

Amazon MSK は Firehose と統合され、Apache Kafka クラスターから Amazon S3 データレイクにストリームを配信するためのサーバーレス、ノーコードのソリューションを提供します。Firehose は、Amazon MSK Kafka トピックからデータを読み取り、Parquet への変換などの変換を実行し、データを集約して Amazon S3 に書き込むストリーミング抽出、変換、ロード (ETL) サービスです。コンソールから数回クリックするだけで、Kafka トピックから読み取って S3 の場所に配信するように Firehose ストリームを設定できます。コードを記述する必要がなく、コネクタアプリケーションも、プロビジョニングするリソースもありません。Firehose は Kafka トピックに発行されたデータ量に基づいて自動的にスケールされ、Kafka から取り込まれたバイト数に対してのみ料金が発生します。

この機能の詳細については、以下を参照してください。

- 「Amazon Data Firehose デベロッパーガイド」の Amazon MSK を使用した Kinesis Data Firehose への書き込み (Amazon Kinesis Data Firehose) に関するページ
- ブログ: Amazon MSK Introduces Managed Data Delivery from Apache Kafka to Your Data Lake
- ラボ: Firehose を使用した Amazon S3 への配信

Amazon MSK コンソールから Amazon EventBridge Pipes にアクセスする

Amazon EventBridge Pipes はソースをターゲットに接続します。パイプは、サポートされているソースとターゲット間のポイントツーポイント統合を目的としており、高度な変換とエンリッチメントをサポートしています。EventBridge Pipes は、Amazon MSK クラスターを Step Functions、Amazon SQS、API Gateway などの AWS サービスや、Salesforce などのサードパーティーの Software as a Service (SaaS) アプリケーションに接続するための非常にスケーラブルな方法を提供します。

パイプをセットアップするには、ソースを選択し、オプションのフィルタリングを追加し、オプションのエンリッチメントを定義し、イベントデータのターゲットを選択します。

Amazon MSK クラスターの詳細ページでは、そのクラスターをソースとして使用しているパイプを表示できます。そこから、次の事柄も実行できます。

- EventBridge コンソールを起動し、パイプの詳細を表示します。
- EventBridge コンソールを起動し、クラスターをソースとする新しいパイプを作成します。

Amazon MSK クラスターをパイプソースとして設定する方法の詳細については、「Amazon EventBridge ユーザーガイド」の「Amazon Managed Streaming for Apache Kafka トピックをソースとして使用する場合」を参照してください。EventBridge Pipes 全般の詳細については、「Amazon EventBridge Pipes」を参照してください。

特定の Amazon MSK クラスターの EventBridge パイプにアクセスするには

1. Amazon MSK コンソールを開き、[クラスター] を選択します。
2. クラスターを選択します。
3. クラスター詳細ページで、[統合] タブを選択します。

   [統合] タブには、選択したクラスターをソースとして使用するように現在設定されているパイプのリストが表示され、以下の情報が含まれます。

   - パイプ名
   - 現在の状態
   - パイプのターゲット
   - パイプが最後に変更された時間
4. 必要に応じて Amazon MSK クラスターのパイプを管理します。

   パイプの詳細にアクセスするには

   - パイプを選択します。

     これにより、EventBridge コンソールの [パイプの詳細] ページが起動します。

   新しいパイプを作成するには

   - [Amazon MSK クラスターを Pipe に接続] を選択します。

     これにより、EventBridge コンソールの [パイプの作成] ページが起動し、Amazon MSK クラスターがパイプのソースとして指定された状態になります。詳細については、「Amazon EventBridge ユーザーガイド」の「Amazon EventBridge パイプの作成」を参照してください。

   - [クラスター] ページからクラスターのパイプを作成することもできます。クラスターを選択し、[アクション] メニューから [EventBridge パイプの作成] を選択します。

MSK Express ブローカーと MSK Serverless での Kafka Streams の使用

Kafka Streams は、ステートレス変換とステートフル変換をサポートしています。カウント、集計、結合などのステートフル変換では、内部 Kafka トピックに状態を保存する演算子を使用します。さらに、groupBy や再パーティションなどのステートレス変換の中には、結果を内部 Kafka トピックに格納するものがあります。デフォルトでは、Kafka Streams は対応する演算子に基づいてこれらの内部トピックに名前を付けます。これらのトピックが存在しない場合、Kafka Streams は内部 Kafka トピックを作成します。内部トピックを作成するために、Kafka Streams は segment.bytes 設定をハードコードし、50 MB に設定します。Express ブローカーと MSK Serverless でプロビジョニングされた MSK は、トピック作成時の segment.size など、一部のトピック設定を保護します。したがって、ステートフル変換を使用する Kafka Streams アプリケーションは、MSK Express ブローカーまたは MSK Serverless を使用して内部トピックを作成できません。

このような Kafka Streams アプリケーションを MSK Express ブローカーまたは MSK Serverless で実行するには、内部トピックを自分で作成する必要があります。これを行うには、まずトピックを必要とする Kafka Streams 演算子を特定して名前を付けます。次に、対応する内部 Kafka トピックを作成します。

> **Note**
>
> - Kafka Streams、特に内部トピックに依存する演算子に手動で名前を付けるのがベストプラクティスです。命名演算子の詳細については、Kafka Streams ドキュメントの「Kafka Streams DSL アプリケーションの命名演算子」を参照してください。
> - ステートフル変換の内部トピック名は、`application.id`Kafka Streams アプリケーションの とステートフル演算子 の名前によって異なります`application.id-statefuloperator_name`。

トピック

- MSK Express ブローカーまたは MSK Serverless を使用した Kafka Streams アプリケーションの作成

MSK Express ブローカーまたは MSK Serverless を使用した Kafka Streams アプリケーションの作成

Kafka Streams アプリケーションが にapplication.id設定されている場合msk-streams-processing、MSK Express ブローカーまたは MSK Serverless を使用して Kafka Streams アプリケーションを作成できます。これを行うには、 count()演算子を使用します。この演算子には、 という名前の内部トピックが必要です。例えば、msk-streams-processing-count-store と指定します。

Kafka Streams アプリケーションを作成するには、次の手順を実行します。

トピック

- 演算子の特定と名前付け
- 内部トピックを作成する
- (オプション) トピック名を確認する
- 命名演算子の例

演算子の特定と名前付け

1. Kafka Streams ドキュメントの「ステートフル変換」を使用してステートフルプロセッサを特定します。

   ステートフルプロセッサの例にはcount、、aggregate、 などがありますjoin。
2. 再パーティション化のトピックを作成するプロセッサを特定します。

   次の例には、 状態を必要とする count()オペレーションが含まれています。

   ```
   var stream =
       paragraphStream
           .groupByKey()
           .count()
           .toStream();
   ```
3. トピックに名前を付けるには、ステートフルプロセッサごとに名前を追加します。プロセッサタイプに基づいて、命名は別の命名クラスによって行われます。例えば、 count()オペレーションは集計オペレーションです。したがって、 Materialized クラスが必要です。

ステートフルオペレーションの命名クラスの詳細については、Kafka Streams ドキュメントの「結論」を参照してください。

次の例では、 Materialized クラスcount-storeを使用してcount()演算子の名前を に設定します。

```
var stream =
    paragraphStream
        .groupByKey()
        .count(Materialized.<String, Long, KeyValueStore<Bytes, byte[]>>as("count-
store")  // descriptive name for the store
            .withKeySerde(Serdes.String())
            .withValueSerde(Serdes.Long()))
        .toStream();
```

内部トピックを作成する

Kafka Streams プレフィックスを内部トピックの名前application.idに。 application.idはユーザー定義です。例えば、application.id-internal_topic_name と指定します。内部トピックは通常の Kafka トピックであり、Kafka API AdminClientの Apache Kafka トピックを作成するまたは で利用可能な情報を使用してトピックを作成できます。

ユースケースに応じて、Kafka Streams のデフォルトのクリーンアップポリシーと保持ポリシーを使用するか、値をカスタマイズできます。これらは cleanup.policyおよび で定義しますretention.ms。

次の例では、 AdminClient API を使用してトピックを作成し、 application.idを に設定します**msk-streams-processing**。

```
try (AdminClient client = AdminClient.create(configs.kafkaProps())) {
    Collection<NewTopic> topics = new HashSet<>();
    topics.add(new NewTopic("msk-streams-processing-count-store", 3, (short) 3));
    client.createTopics(topics);
}
```

トピックがクラスターに作成されると、Kafka Streams アプリケーションは msk-streams-processing-count-store トピックを count()オペレーションに使用できます。

(オプション) トピック名を確認する

トポグラフィ記述子を使用して、ストリームのトポロジを記述し、内部トピックの名前を表示できます。次の例は、トポロジ記述子を実行する方法を示しています。

```
final StreamsBuilder builder = new StreamsBuilder();
Topology topology = builder.build();
System.out.println(topology.describe());
```

次の出力は、前の例のストリームのトポロジを示しています。

```
Topology Description:
Topologies:
   Sub-topology: 0
    Source: KSTREAM-SOURCE-0000000000 (topics: [input_topic])
      --> KSTREAM-AGGREGATE-0000000001
    Processor: KSTREAM-AGGREGATE-0000000001 (stores: [count-store])
      --> KTABLE-TOSTREAM-0000000002
      <-- KSTREAM-SOURCE-0000000000
    Processor: KTABLE-TOSTREAM-0000000002 (stores: [])
      --> KSTREAM-SINK-0000000003
      <-- KSTREAM-AGGREGATE-0000000001
    Sink: KSTREAM-SINK-0000000003 (topic: output_topic)
      <-- KTABLE-TOSTREAM-0000000002
```

トポロジー記述子の使用方法については、Kafka Streams ドキュメントの「Kafka Streams DSL アプリケーションの命名演算子」を参照してください。

命名演算子の例

このセクションでは、命名演算子の例をいくつか示します。

groupByKey() の命名演算子の例

```
groupByKey() -> groupByKey(Grouped.as("kafka-stream-groupby"))
```

法線カウントの命名演算子の例 ()

```
normal count() -> .count(Materialized.<String, Long, KeyValueStore<Bytes,
 byte[]>>as("kafka-streams-window")  // descriptive name for the store
    .withKeySerde(Serdes.String())
```

```
    .withValueSerde(Serdes.Long()))
```

ウィンドウカウントの命名演算子の例 ()

```
windowed count() -> .count(Materialized.<String, Long, WindowStore<Bytes,
 byte[]>>as("kafka-streams-window")  // descriptive name for the store
    .withKeySerde(Serdes.String())
    .withValueSerde(Serdes.Long()))
```

windowed suppressed() の命名演算子の例

```
windowed suppressed() ->
Suppressed<Windowed> suppressed = Suppressed
        .untilWindowCloses(Suppressed.BufferConfig.unbounded())
        .withName("kafka-suppressed");
    .suppress(suppressed)
```

リアルタイムベクトル埋め込みブループリント

Amazon MSK (Managed Streaming for Apache Kafka) は、Amazon Managed Service for Apache Flink の設計図をサポートし、Amazon Bedrock を使用してベクトル埋め込みを生成し、up-to-dateコンテキストデータを活用したリアルタイムの AI アプリケーションを構築するプロセスを合理化します。MSF ブループリントは、Amazon MSK ストリーミングパイプラインの最新データを生成 AI モデルに組み込むプロセスを簡素化するため、リアルタイムデータストリーム、ベクトルデータベース、大規模言語モデルを統合するカスタムコードを記述する必要がなくなります。

Bedrock の埋め込みモデルを使用してベクトル埋め込みを継続的に生成し、Amazon MSK データストリームの OpenSearch Service でそれらの埋め込みのインデックスを作成するように MSF ブループリントを設定できます。これにより、リアルタイムデータのコンテキストと Bedrock の強力な大規模言語モデルを組み合わせて、カスタムコードを記述することなく、正確でup-to-date AI レスポンスを生成できます。また、オープンソースライブラリである LangChain からのデータチャンキング技術の組み込みサポートを使用して、モデル取り込みのための高品質の入力をサポートするデータ取り出しの効率を向上させることもできます。このブループリントは、MSK、選択した埋め込みモデル、OpenSearch ベクトルストア間のデータ統合と処理を管理し、基盤となる統合を管理するのではなく、AI アプリケーションの構築に集中できるようにします。

リアルタイムベクトル埋め込みブループリントは、次の AWS リージョンで使用できます。

- バージニア北部 - us-east-1

- オハイオ - us-east-2
- オレゴン - us-west-2
- ムンバイ - ap-south-1
- ソウル - ap-northeast-2
- シンガポール - ap-southeast-1
- シドニー - ap-southeast-2
- 東京 - ap-northeast-1
- カナダ中部 - ca-central-1
- フランクフルト - eu-central-1
- アイルランド - eu-west-1
- ロンドン - eu-west-2
- パリ - eu-west-3
- サンパウロ - sa-east-1

トピック

- ログ記録とオブザーバビリティ
- リアルタイムベクトル埋め込みブループリントを有効にする前の注意事項
- ストリーミングデータベクトル化の設計図をデプロイする

ログ記録とオブザーバビリティ

リアルタイムベクトル埋め込みブループリントのすべてのログとメトリクスは、CloudWatch Logs を使用して有効にできます。

通常の MSF アプリケーションと Amazon Bedrock で使用できるすべてのメトリクスは、アプリケーションと Bedrock メトリクスをモニタリングできます。

埋め込みの生成のパフォーマンスをモニタリングするための追加のメトリクスが 2 つあります。これらのメトリクスは、CloudWatch の EmbeddingGeneration オペレーション名の一部です。

- BedrockTitanEmbeddingTokenCount: Bedrock への 1 回のリクエストに存在するトークンの数をモニタリングします。
- BedrockEmbeddingGenerationLatencyMs: Bedrock から埋め込みを生成するレスポンスを送受信するのにかかった時間をミリ秒単位でレポートします。

OpenSearch Service では、次のメトリクスを使用できます。

- OpenSearch Serverless コレクションメトリクス: Amazon OpenSearch Service デベロッパーガイドの「Amazon CloudWatch による OpenSearch Serverless のモニタリング」を参照してください。 OpenSearch
- OpenSearch でプロビジョニングされたメトリクス: Amazon OpenSearch Service デベロッパーガイドの「Amazon CloudWatch による OpenSearch クラスターメトリクスのモニタリング」を参照してください。 OpenSearch

リアルタイムベクトル埋め込みブループリントを有効にする前の注意事項

Managed Service for Apache Flink アプリケーションは、入力ストリーム内の非構造化テキストまたは JSON データのみをサポートします。

2 つの入力処理モードがサポートされています。

- 入力データが非構造化テキストの場合、テキストメッセージ全体が埋め込まれます。ベクトル DB には、元のテキストと生成された埋め込みが含まれます。
- 入力データが JSON 形式の場合、アプリケーションでは、埋め込みプロセスに使用する JSON オブジェクト値内で 1 つ以上のキーを設定および指定できます。キーが複数ある場合、すべてのキーは一緒にベクトル化され、ベクトル DB でインデックスが作成されます。ベクトル DB には、元のメッセージと生成された埋め込みが含まれます。

埋め込み生成: アプリケーションは、Bedrock が提供するすべてのテキスト埋め込みモデルをサポートしています。

ベクトル DB ストアに保持する: アプリケーションは、埋め込みデータを保持するための送信先として、お客様のアカウントの既存の OpenSearch クラスター (プロビジョニング済みまたはサーバーレス) を使用します。Opensearch Serverless を使用してベクトルインデックスを作成する場合は、常にベクトルフィールド名 を使用しますembedded_data。

MSF ブループリントと同様に、リアルタイムベクトル埋め込みブループリントに関連付けられたコードを実行するインフラストラクチャを管理することが期待されます。

MSF ブループリントと同様に、MSF アプリケーションが作成されると、コンソールまたは CLI を使用して AWS アカウントでのみ起動する必要があります。 AWS は MSF アプリケーションを起動しません。アプリケーションを実行するには、StartApplication API を (CLI またはコンソールを介して) 呼び出す必要があります。

データのクロスアカウント移動: アプリケーションでは、異なる AWS アカウントに存在する入力ストリームとベクトル送信先の間でデータを移動することはできません。

ストリーミングデータベクトル化の設計図をデプロイする

このトピックでは、ストリーミングデータベクトル化ブループリントをデプロイする方法について説明します。

ストリーミングデータベクトル化の設計図をデプロイする

1. 次のリソースが正しく設定されていることを確認します。
   - データを含む 1 つ以上のトピックを含むプロビジョンド MSK クラスターまたはサーバーレス MSK クラスター。
2. Bedrock セットアップ: 目的の Bedrock モデルへのアクセス。現在サポートされている Bedrock モデルは次のとおりです。
   - Amazon Titan Embeddings G1 - Text
   - Amazon Titan Text Embeddings V2
   - Amazon Titan Multimodal Embeddings G1
   - Cohere Embed English
   - Cohere Embed Multilingual
3. AWS OpenSearch コレクション：
   - プロビジョニングされた OpenSearch Service コレクションまたは Serverless OpenSearch Service コレクションを使用できます。
   - OpenSearch Service コレクションには少なくとも 1 つのインデックスが必要です。
   - OpenSearch Serverless コレクションを使用する場合は、必ずベクトル検索コレクションを作成してください。ベクトルインデックスの設定方法の詳細については、「ナレッジベースの独自のベクトルストアの前提条件」を参照してください。ベクトル化の詳細については、「Amazon OpenSearch Service のベクトルデータベース機能の説明」を参照してください。

     > **Note**
     >
     > ベクトルインデックスを作成するときは、ベクトルフィールド名 を使用する必要がありますembedded_data。

- OpenSearch Provisioned コレクションを使用する場合は、ブループリントによって作成された MSF アプリケーションロール (Opensearch アクセスポリシーを含む) を、マスターユーザーとして OpenSearch コレクションに追加する必要があります。また、OpenSearch のアクセスポリシーが「許可」アクションに設定されていることを確認します。これは、きめ細かなアクセスコントロールを有効にするために必要です。
- オプションで、OpenSearch ダッシュボードへのアクセスを有効にして結果を表示できます。「」を参照して、きめ細かなアクセスコントロールを有効にします。

4. aws:CreateStack アクセス許可を許可するロールを使用してログインします。
5. MSF コンソールダッシュボードに移動し、ストリーミングアプリケーションの作成を選択します。
6. 「メソッドを選択してストリーム処理アプリケーションを設定する」で「設計図を使用する」を選択します。
7. ブループリントドロップダウンメニューからリアルタイム AI アプリケーションのブループリントを選択します。
8. 必要な設定を指定します。「ページ設定を作成する」を参照してください。
9. デプロイブループリントを選択して CloudFormation デプロイを開始します。
10. CloudFormation のデプロイが完了したら、デプロイされた Flink アプリケーションに移動します。アプリケーションのランタイムプロパティを確認します。
11. ランタイムプロパティをアプリケーションに変更/追加することを選択できます。これらのプロパティを設定する詳細については、「ランタイムプロパティ設定」を参照してください。

> **Note**
>
> メモ:
> OpenSearch プロビジョニングを使用している場合は、きめ細かなアクセスコントロールが有効になっていることを確認してください。
> プロビジョニングされたクラスターがプライベートの場合は、OpenSearch Provisioned VPC エンドポイント URL `https://` に を追加し、このエンドポイントを指すsink.os.endpointように を変更します。
> プロビジョニングされたクラスターがパブリックの場合は、MSF アプリケーションがインターネットにアクセスできることを確認します。詳細については、「>>>>>> express-brokers-publication-merge type"documentation" url"managed-flink/latest/java/vpc-internet.html" >VPC に接続された Managed Service for Apache Flink アプリケーションのインターネットおよびサービスアクセス」を参照してください。

12. すべての設定に満足したら、 を選択しますRun。アプリケーションの実行が開始されます。
13. MSK クラスターでメッセージをポンプします。
14. Opensearch クラスターに移動し、OpenSearch ダッシュボードに移動します。
15. ダッシュボードで、左側のメニューで検出を選択します。永続ドキュメントとそのベクトル埋め込みが表示されます。
16. インデックスに保存されているベクトルの使用方法については、「ベクトル検索コレクションの使用」を参照してください。

ページ設定を作成する

このトピックでは、リアルタイム AI アプリケーションブループリントの設定を指定するときに参照するページ作成設定について説明します。

アプリケーション名

MSF の既存のフィールドで、アプリケーションに任意の名前を付けます。

MSK クラスター

ドロップダウンリストから、セットアップ中に作成した MSK クラスターを選択します。

トピック

セットアップで作成したトピックの名前を追加します (複数可) 。

入力ストリームのデータ型

MSK ストリームに文字列入力を指定する場合は、文字列を選択します。

MSK ストリームの入力が JSON の場合は、JSON を選択します。埋め込み JSON キーで、埋め込みを生成するために Bedrock に送信する値を持つ入力 JSON のフィールドの名前を書き込みます。

Bedrock 埋め込みモデル

リストから 1 つ選択します。選択したモデルのモデルアクセス権があることを確認してください。そうしないと、スタックが失敗する可能性があります。「Amazon Bedrock 基盤モデルへのアクセスの追加または削除」を参照してください。

OpenSearch クラスター

ドロップダウンから作成したクラスターを選択します。

OpenSearch ベクトルインデックス名

上記のステップで作成したベクトルインデックスを選択します。

Amazon MSK クォータ

AWS アカウント には Amazon MSK のデフォルトのクォータがあります。特に明記されていない限り、アカウントごとの各クォータは 内でリージョン固有です AWS アカウント。

トピック

- Amazon MSK のクォータ引き上げのリクエスト
- Amazon MSK Standard ブローカークォータ
- Amazon MSK Express ブローカークォータ
- MSK レプリケータークォータ
- MSK サーバーレス クォータ
- MSK Connect クォータ

Amazon MSK のクォータ引き上げのリクエスト

Service Quotas コンソール、またはサポートケースを使用して AWS CLI、リージョンごとにクォータの引き上げをリクエストできます。Service Quotas コンソールで調整可能なクォータを使用できない場合は、 AWS Support Center Console を使用してサービスクォータの引き上げケースを作成します。

サポートは、クォータ引き上げリクエストを承認、拒否、または部分的に承認できます。引き上げはすぐには許可されず、有効になるまでに数日かかる場合があります。

増加をリクエストするには、Service Quotas コンソールを使用してください。

1. Service Quotas のコンソールを開きます。https://console.aws.amazon.com/servicequotas/
2. 画面上部のナビゲーションバーで、リージョンを選択します。
3. 左側のナビゲーションペインで、[AWS のサービス] を選択します。
4. 「サービスの検索」ボックスで「」と入力し**msk**、「Amazon Managed Streaming for Apache Kafka (MSK)」を選択します。
5. サービスクォータで、引き上げをリクエストするクォータ名を選択します。例えば、**Number of brokers per account**。
6. [アカウントレベルでの引き上げをリクエスト] をクリックします。
7. クォータ値を増やすには、新しいクォータ値を入力します。

8. [リクエスト] を選択します。
9. (オプション) コンソールで保留中または最近解決されたリクエストを表示するには、左側のナビゲーションペインで Dashboard を選択します。保留中のリクエストの場合は、リクエストのステータスを選択してリクエストの受信をオープンします。リクエストの初期ステータスは [Pending] (保留中) です。ステータスがクォータリクエストに変更されると、 サポートにケース番号が表示されます。リクエストのチケットを開くには、ケース番号を選択します。

AWS CLI または SDKs を使用してクォータの引き上げをリクエストする方法などの詳細については、Service Quotas ユーザーガイド」の「クォータの引き上げのリクエスト」を参照してください。

Amazon MSK Standard ブローカークォータ

次の表に、標準ブローカーのクォータを示します。

ディメンション	クォータ	メモ
アカウントあたりのブローカー数	90	クォータの引き上げをリクエストするには、Service Quotas コンソールに移動します。
クラスターあたりのブローカー数	ZooKeeper ベースのクラスターの場合は 30、KRaft ベースのクラスターの場合は 60	クォータの引き上げをリクエストするには、Service Quotas コンソールに移動します。
ブローカーあたりの最小ストレージ	1 GiB	
ブローカーあたりの最大ストレージ	16384 GiB	
ブローカーあたりの最大 TCP 接続数 (IAM アクセスコントロール)	3000	この制限を引き上げるには、Kafka `AlterConfig` API `listener.name.client_iam.max.connect`

ディメンション	クォータ	メモ
		ions または kafka-configs.sh ツールを使用して または listener.name.client_iam_public.max.connections 設定プロパティを調整できます。いずれかのプロパティを高い値に増やすと、使用できなくなる可能性があることに注意してください。
ブローカーあたりの最大 TCP 接続レート (IAM)	100/秒 (M5 および M7g インスタンスサイズ) 4/秒 (t3 インスタンスサイズ)	接続に失敗した場合の再試行を処理するには、クライアント側で reconnect.backoff.ms 設定パラメータを設定できます。たとえば、クライアントが 1 秒後に接続を再試行する場合は、reconnect.backoff.ms を に設定します1000。詳細については、Apache Kafka のドキュメントの reconnect.backoff.msを参照してください。
ブローカーあたりの最大 TCP 接続数 (非 IAM)	該当なし	MSK は、IAM 以外の認証の接続制限を適用しません。CPU やメモリの使用状況などの他のメトリクスをモニタリングして、過剰な接続が原因でクラスターが過負荷にならないようにする必要があります。

ディメンション	クォータ	メモ
アカウントあたりの設定	100	クォータの引き上げをリクエストするには、Service Quotas コンソールに移動します。 MSK クラスターの構成または Apache Kafka バージョンを更新するには、最初に、ブローカーごとのパーティションの数が クラスターの適切なサイズ設定: 標準ブローカーあたりのパーティション数 で説明されている制限を下回っていることを確認します。
アカウントあたりの設定リビジョン	50	

Amazon MSK Express ブローカークォータ

次の表に、Express ブローカーのクォータを示します。

ディメンション	クォータ	メモ
アカウントあたりのブローカー数	90	クォータの引き上げをリクエストするには、Service Quotas コンソールに移動します。
クラスターあたりのブローカー数	30	クォータの引き上げをリクエストするには、Service Quotas コンソールに移動します。
最大ストレージ	無制限	

ディメンション	クォータ	メモ
ブローカーあたりの最大 TCP 接続数 (IAM アクセスコントロール)	3000	接続制限を引き上げるには、Kafka AlterConfig API または kafka-configs.sh ツールを使用して、次のいずれかの設定プロパティを調整します。 • `listener.name.client_iam.max.connections` • `listener.name.client_iam_public.max.connections` これらのプロパティを高い値に設定すると、クラスターが使用できなくなる可能性があります。
ブローカーあたりの最大 TCP 接続レート (IAM)	100/秒	接続に失敗した場合の再試行を処理するには、クライアント側で `reconnect.backoff.ms` 設定パラメータを設定できます。たとえば、クライアントが 1 秒後に接続を再試行する場合は、`reconnect.backoff.ms` を `1000` に設定します。詳細については、Apache Kafka のドキュメントの reconnect.backoff.ms を参照してください。

ディメンション	クォータ	メモ
ブローカーあたりの最大 TCP 接続数 (非 IAM)	該当なし	MSK は、IAM 以外の認証の接続制限を適用しません。ただし、CPU やメモリの使用状況などの他のメトリクスをモニタリングして、過剰な接続が原因でクラスターが過負荷にならないようにする必要があります。
アカウントあたりの設定	100	クォータの引き上げをリクエストするには、Service Quotas コンソールに移動します。MSK クラスターの構成または Apache Kafka バージョンを更新するには、最初に、ブローカーごとのパーティションの数が クラスターの適切なサイズ設定: Express ブローカーあたりのパーティション数 で説明されている制限を下回っていることを確認します。
アカウントあたりの設定リビジョン	50	
ブローカーあたりの最大進入	推奨: 15.6 ~ 500.0 MBps	インスタンスサイズに基づきます。
ブローカーあたりの最大エグレス	推奨: 31.2 ~ 1000.0 MBps	インスタンスサイズに基づきます。

トピック

- ブローカーサイズ別の高速ブローカースループットスロットル制限
- Express ブローカーパーティションクォータ

ブローカーサイズ別の高速ブローカースループットスロットル制限

次の表に、さまざまなブローカーサイズの入出力に関連する推奨スループットスロットル制限と最大スループットスロットル制限を示します。この表では、推奨スループットは継続的なパフォーマンスとして表されます。これは、アプリケーションがパフォーマンスの低下を経験しないしきい値です。いずれかのディメンションでこれらの制限を超えて操作すると、スループットが高くなる可能性がありますが、パフォーマンスが低下する可能性もあります。最大クォータは、クラスターが読み取り/書き込みトラフィックをスロットリングするしきい値です。アプリケーションはこのしきい値を超えて動作することはできません。

インスタンスサイズ	進入の持続的なパフォーマンス (MBps)	イングレスの最大クォータ (MBps)	出力の持続的なパフォーマンス (MBps)	出力の最大クォータ (MBps)
express.m7g.large	15.6	23.4	31.2	58.5
express.m7g.xlarge	31.2	46.8	62.5	117
express.m7g.2xlarge	62.5	93.7	125	234.2
express.m7g.4xlarge	124.9	187.5	249.8	468.7
express.m7g.8xlarge	250	375	500	937.5
express.m7g.12xlarge	375	562.5	750	1406.2
express.m7g.16xlarge	500	750	1,000	1875

Express ブローカーパーティションクォータ

次の表は、各 Express ブローカーの推奨パーティション数 (リーダーレプリカとフォロワーレプリカを含む) を示しています。Express ブローカーごとに、以下の表に記載されているパーティションの最大数を超えることはできません。

Express ブローカーにパーティションを割り当てる際に考慮すべきベストプラクティスについては、「」を参照してくださいクラスターの適切なサイズ設定: Express ブローカーあたりのパーティション数。

ブローカーサイズ	推奨されるブローカーあたりのパーティション数 (リーダーとフォロワーのレプリカを含む)	ブローカーあたりのパーティションの最大数
express.m7g.large	1,000	1500
express.m7g.xlarge	1,000	2000
express.m7g.2xlarge	2500	4000
express.m7g.4xlarge	6000	8000
express.m7g.8xlarge	12000	16000
express.m7g.12xlarge	16000	24000
express.m7g.16xlarge	20000	32000

MSK レプリケーターク ォータ

- アカウントあたり最大 15 個の MSK レプリケーター。
- MSK Replicator は、ソート順に最大 750 のトピックのみをレプリケートします。より多くのトピックをレプリケートする必要がある場合は、別のレプリケーターを作成することをお勧めします。レプリケーターごとに 750 を超えるトピックのサポートが必要な場合は、Service Quotas コンソールに移動します。レプリケートされるトピックの数は、「TopicCount」メトリクスを使用してモニタリングできます。

- MSK レプリケーターごとに 1 秒あたり 1 GB の最大入力スループット。Service Quotas コンソールを使用して、クオータの引き上げをリクエストします。
- MSK Replicator レコードサイズ – 最大 10 MB のレコードサイズ (message.max.bytes)。Service Quotas コンソールを使用して、クオータの引き上げをリクエストします。

MSK サーバーレス クォータ

次の表で指定されているクォータは、特に明記されていない限り、クラスターごとです。

> **Note**
>
> サービスクォータの制限に関する問題が発生した場合は、ユースケースとリクエストした制限を記載したサポートケースを作成してください。

ディメンション	クォータ	クォータ違反の結果
最大入力スループット	200 MBps	レスポンスのスロットル時間による速度低下
最大出力スループット	400 MBps	レスポンスのスロットル時間による速度低下
最大保持期間	無制限	該当なし
クライアント接続の最大数	3000	接続終了
最大接続試行回数	100/秒	接続終了
最大メッセージサイズ (MB)	8 MiB	リクエストの失敗 (エラーコード: INVALID_REQUEST)
最大リクエストレート	1 秒あたり 15,000	レスポンスのスロットル時間による速度低下
トピック管理 API の最大リクエストレート	2/秒	レスポンスのスロットル時間による速度低下

ディメンション	クォータ	クォータ違反の結果
リクエストあたりの最大フェッチバイト数	55 MB	リクエストの失敗 (エラーコード: INVALID_REQUEST)
ユーザーグループの最大数	500	JoinGroup リクエストの失敗
パーティション (リーダー) の最大数	圧縮されていないトピックの場合は 2400、圧縮されたトピックの場合は 120。サービスクォータの調整をリクエストするには、ユースケースとリクエストした制限を記載したサポートケースを作成します。	リクエストの失敗 (エラーコード: INVALID_REQUEST)
パーティションの作成と削除の最大レート	5 分で 250	リクエストの失敗 (エラーコード: THROUGHPUT_QUOTA_EXCEEDED)
パーティションあたりの最大入力スループット	5 MBps	レスポンスのスロットル時間による速度低下
パーティションあたりの最大出力スループット	10 MBps	レスポンスのスロットル時間による速度低下
最大パーティションサイズ (圧縮されたトピック用)	250 GB	リクエストの失敗 (エラーコード: THROUGHPUT_QUOTA_EXCEEDED)
サーバーレスクラスターあたりのクライアント VPC の最大数	5	
アカウントあたりのサーバーレスクラスターの最大数	10. サービスクォータの調整をリクエストするには、ユースケースとリクエストした制限を記載したサポートケースを作成します。	

MSK Connect クォータ

- 最大 100 個のカスタムプラグイン。
- 最大 100 個のワーカー構成。
- 最大 60 人のコネクトワーカー。コネクタがオートスケーリングされた容量を持つように設定されている場合、コネクタが持つように設定されているワーカーの最大数は、MSK Connect がアカウントのクォータを計算するために使用する数です。
- コネクタごとに最大 10 人のワーカー。

MSK Connect のクォータの引き上げをリクエストするには、Service Quotas コンソールに移動します。

Amazon MSK デベロッパーガイドのドキュメント履歴

次の表に、Amazon MSK デベロッパーガイドの重要な変更点を示します。

ドキュメント最終更新日: 2024 年 6 月 25 日

変更	説明	日付
StorageExpress ブローカーの StorageUsedメトリクス	Amazon MSK に、レプリカを除くストレージの総消費量をクラスターレベルで可視化する Express ブローカー`StorageUsed` 用の新しい DEFAULT レベルメトリクスが追加されました。詳細については、「Express ブローカーのデフォルトレベルのモニタリング」を参照してください。	2025-07-24
Amazon MSK Express ブローカーのパーティション数が多い	Amazon MSK は Express ブローカーに対して高いパーティション数を開始しました。詳細については、「Express broker partition quota」を参照してください。	2025-07-21
新しい Amazon MSK Connect メトリクス	MSK Connect は、コネクタデータスループット率を測定する`SourceProducerByteRate` ための `SinkConsumerByteRate` と の 2 つの新しいメトリクスを追加しました。詳細については、「MSK Connect のモニタリング」を参照してください。	2025-06-30

変更	説明	日付
MSK プロビジョンド入門トピックの見直し	ユーザーエクスペリエンス、コンテンツフロー、読みやすさを向上させるために、MSK プロビジョンド入門トピックを完全に再構築しました。改訂されたトピックには、ストレージモード、認証方法、モニタリングレベルなど、すべてのコンソールオプションの詳細なドキュメントと明確な決定ガイダンスも含まれています。 詳細については、「Amazon MSK の使用を開始する」を参照してください。	2025-06-28
Apache Kafka バージョン 3.8.x での Express ブローカーのサポート	Amazon MSK は、Apache Kafka バージョン 3.8.x で Express ブローカーをサポートするようになりました。詳細については、「Amazon MSK Express ブローカー」を参照してください。	2025-06-05

変更	説明	日付
新しい Amazon MSK レプリケーターのトラブルシューティング情報	Amazon Managed Streaming for Apache Kafka デベロッパーガイドに、診断スクリプトと詳細なエラー軽減手順を含む包括的な MSK レプリケーターのトラブルシューティングドキュメントが追加されました。詳細については、「ReplicatorFailure メトリクスを使用した MSK レプリケーターの障害のトラブルシューティング」を参照してください。	2025-05-09
中断のない証明書の更新	Amazon MSK は、Express ブローカーの非中断証明書更新を開始して、必須の 13 か月間の証明書更新中のメンテナンスダウンタイムを排除しました。詳細については、「Amazon MSK 暗号化」を参照してください。	2025-05-05
Apache Kafka バージョン 4.0.x のサポート	Amazon MSK が Apache Kafka バージョン 4.0.x をサポートするようになりました。詳細については、「サポートされている Apache Kafka バージョン」を参照してください。	2025-05-02
中国リージョンでの Amazon MSK Connect の起動	MSK Connect が、中国 (北京) と中国 (寧夏) のすべての中国リージョンで利用可能になりました。	2025-04-10

変更	説明	日付
Apache Kafka バージョン 3.9.x のサポート	Amazon MSK が Apache Kafka バージョン 3.9.x をサポートするようになりました。詳細については、「サポートされている Apache Kafka バージョン」を参照してください。	2025-04-21
MSK Connect 用の Amazon EventBridge Kafka シンクコネクタ	Amazon Managed Streaming for Apache Kafka デベロッパーガイドに、MSK Connect で EventBridge Kafka シンクコネクタを使用する方法を説明する包括的なトピックが追加されました。詳細については、「MSK Connect 用の EventBridge Kafka シンクコネクタのセットアップ」を参照してください。	2025-03-28
Express ブローカーのクォータの更新	Amazon Managed Streaming for Apache Kafka デベロッパーガイドに、Express ブローカーの入出力のスループット制限に関する情報が追加されました。詳細については、「Amazon MSK Express ブローカークォータ」を参照してください。	2025-03-06

変更	説明	日付
Apache Kafka バージョン 3.8.x のサポート	Amazon MSK が Apache Kafka バージョン 3.8.x をサポートするようになりました。詳細については、「サポートされている Apache Kafka バージョン」を参照してください。	2025-02-20
Amazon MSK バージョン 3.4.0 のサポート終了日の改訂	Apache Kafka バージョン 3.4.0 のサポート終了日は 2025 年 8 月 4 日です。詳細については、「サポートされている Apache Kafka バージョン」を参照してください。	2025-02-18
既存の MSK Connect コネクタ設定を変更するための UpdateConnector API の起動	Amazon MSK に UpdateConnector API が追加され、既存の MSK Connect コネクタ設定を変更できるようになりました。これにより、新しいコネクタを作成する必要がなくなります。また、コネクタの更新オペレーションを追跡し、設定変更の監査証跡の履歴を維持するために、DescribeConnectorOperation API と ListConnectorOperations APIs のドキュメントを追加しました。	2025-01-12

変更	説明	日付
AWS PrivateLink インターフェイス VPC エンドポイントのドキュメント	Amazon Managed Streaming for Apache Kafka デベロッパーガイドに、 AWS PrivateLink インターフェイス VPC エンドポイントのドキュメントが追加されました。詳細については、「インターフェイス VPC エンドポイントで Amazon MSK APIs」を参照してください。	2024-12-18
MSK Connect 3.7.x バージョンの起動	MSK Connect がバージョン 3.7.x をサポートするようになりました。詳細については、「MSK Connect を理解する」を参照してください。	2024-12-18
Express ブローカー機能が追加されました。デベロッパーガイドのトピックが再編成されました。	MSK は、標準および新しい Express ブローカーをサポートしています。	2024-11-6
Graviton インプレースアップグレード機能が追加されました。	クラスターブローカーのサイズは、M5 または T3 から M7g に、または M7g から M5 に更新できます。	2024-6-25
3.4.0 のサポート終了日が発表されました。	Apache Kafka バージョン 3.4.0 のサポート終了日は 2025 年 6 月 17 日です。	2024-6-24

変更	説明	日付
ブローカーの削除機能が追加されました。	一連のブローカーを削除することで、可用性への影響、データ耐久性のリスク、データストリーミングアプリケーションの中断なしに、プロビジョニングされたクラスターのストレージとコンピューティング容量を減らすことができます。	2024-5-16
`WriteDataIdempotently` が AWSMSKReplicatorExecutionRole に追加されました	WriteDataIdempotently アクセス許可が AWSMSKReplicatorExecutionRole ポリシーに追加され、MSK クラスター間のデータレプリケーションがサポートされます。	2024-5-16
Graviton M7g ブローカーがブラジルとバーレーンでリリースされました。	Amazon MSK は、Graviton プロセッサ (Amazon Web Services によって構築されたカスタム Arm ベースのプロセッサ) を使用する M7g ブローカーの南米 (sa-east-1、サンパウロ) および中東 (me-south-1、バーレーン) AWS リージョンの可用性をサポートするようになりました。	2024-2-07

変更	説明	日付
Graviton M7g ブローカーを中国リージョンにリリース	Amazon MSK は、Graviton プロセッサ (Amazon Web Services によって構築されたカスタム Arm ベースのプロセッサ) AWS を使用した M7g ブローカーの中国リージョンの可用性をサポートするようになりました。	2024-01-11
Amazon MSK Kafka バージョンサポートポリシー	Amazon MSK がサポートする Kafka バージョンサポートポリシーの説明を追加しました。詳細については、「Apache Kafka バージョン」を参照してください。	2023-12-08
Amazon MSK Replicator をサポートする新しいサービス実行ロールポリシー。	Amazon MSK に、Amazon MSK Replicator をサポートするための新しい `AWSMSKReplicatorExecutionRole` ポリシーが追加されました。詳細については、「AWS マネージドポリシー AWSMSKReplicatorExecutionRole 」を参照してください。	2023-12-06
M7g Graviton のサポート	Amazon MSK は、Graviton プロセッサ (Amazon Web Services によって構築されたカスタム Arm ベースのプロセッサ) AWS を使用する M7g ブローカーをサポートするようになりました。	2023-11-27

変更	説明	日付
Amazon MSK Replicator	Amazon MSK Replicator は、Amazon MSK クラスター間でデータをレプリケートするために使用できる新機能です。Amazon MSK Replicator には、AmazonMSKFullAccess ポリシーの更新が含まれます。詳細については、「AWS マネージドポリシー AmazonMSKFullAccess」を参照してください。	2023-09-28
IAM ベストプラクティスの更新	IAM ベストプラクティスに沿ってガイドを更新しました。詳細については、「IAM のセキュリティのベストプラクティス」を参照してください。	2023-03-08

変更	説明	日付
マルチ VPC プライベート接続をサポートする、サービスにリンクされたロールの更新	Amazon MSK には、アカウント内のネットワークインターフェイスと VPC エンドポイントを管理するため、AWS ServiceRoleForKafka サービスにリンクされたロールの更新が含まれるようになりました。これにより、VPC 内のクライアントがクラスターブローカーにアクセスできるようになります。Amazon MSK は、`DescribeVpcEndpoints` 、`ModifyVpcEndpoint` 、および `DeleteVpcEndpoints` に対するアクセス許可を使用します。詳細については、「Amazon MSK のサービスリンクロール」を参照してください。	2023-03-08
Apache Kafka 2.7.2 のサポート	Amazon MSK は、Apache Kafka バージョン 2.7.2 をサポートするようになりました。詳細については、「サポート対象の Apache Kafka バージョン」を参照してください。	2021-12-21

変更	説明	日付
Apache Kafka 2.6.3 のサポート	Amazon MSK は、Apache Kafka バージョン 2.6.3 をサポートするようになりました。詳細については、「サポート対象の Apache Kafka バージョン」を参照してください。	2021-12-21
MSK サーバーレスプレリリース	MSK サーバーレスは、サーバーレスクラスターの作成に使用できる新機能です。詳細については、「MSK サーバーレス」を参照してください。	2021-11-29
Apache Kafka 2.8.1 のサポート	Amazon MSK は、Apache Kafka バージョン 2.8.1 をサポートするようになりました。詳細については、「サポート対象の Apache Kafka バージョン」を参照してください。	2021-09-30
MSK Connect	MSK Connect は、Apache Kafka コネクタを作成および管理するために使用できる新機能です。詳細については、「MSK Connect を理解する」を参照してください。	2021-09-16

変更	説明	日付
Apache Kafka 2.7.1 のサポート	Amazon MSK は、Apache Kafka バージョン 2.7.1 をサポートするようになりました。詳細については、「サポート対象の Apache Kafka バージョン」を参照してください。	2021-05-25
Apache Kafka 2.8.0 のサポート	Amazon MSK は、Apache Kafka バージョン 2.8.0 をサポートするようになりました。詳細については、「サポート対象の Apache Kafka バージョン」を参照してください。	2021-04-28
Apache Kafka 2.6.2 のサポート	Amazon MSK は、Apache Kafka バージョン 2.6.2 をサポートするようになりました。詳細については、「サポート対象の Apache Kafka バージョン」を参照してください。	2021-04-28
ブローカータイプの更新のサポート	これで、既存のクラスターのブローカータイプを変更できます。詳細については、「Amazon MSK クラスターのブローカーサイズを更新する」を参照してください。	2021-01-21

変更	説明	日付
Apache Kafka 2.6.1 のサポート。	Amazon MSK は、Apache Kafka バージョン 2.6.1 をサポートするようになりました。詳細については、「サポート対象の Apache Kafka バージョン」を参照してください。	2021-01-19
Apache Kafka 2.7.0 のサポート	Amazon MSK は、Apache Kafka バージョン 2.7.0 をサポートするようになりました。詳細については、「サポート対象の Apache Kafka バージョン」を参照してください。	2020-12-29
Apache Kafka バージョン 1.1.1 では新しいクラスターはありません	Apache Kafka バージョン 1.1.1 を使用して新しい Amazon MSK クラスターを作成することはできなくなりました。ただし、Apache Kafka バージョン 1.1.1 を実行している既存の MSK クラスターがある場合は、それらの既存のクラスターで現在サポートされているすべての機能を引き続き使用できます。詳細については、「Apache Kafka バージョン」を参照してください。	2020-11-24

変更	説明	日付
Consumer-Lag メトリクス	Amazon MSK は、コンシューマーラグをモニタリングするために使用できるメトリクスを提供できるようになりました。詳細については、「Amazon MSK プロビジョンドクラスターをモニタリングする」を参照してください。	2020-11-23
Cruise Control のサポート	Amazon MSK は、LinkedIn の Cruise Control をサポートするようになりました。詳細については、「LinkedIn の Cruise Control for Apache Kafka を Amazon MSK で使用する」を参照してください。	2020-11-17
Apache Kafka 2.6.0 のサポート	Amazon MSK は、Apache Kafka バージョン 2.6.0 をサポートするようになりました。詳細については、「サポート対象の Apache Kafka バージョン」を参照してください。	2020-10-21

変更	説明	日付
Apache Kafka 2.5.1 のサポート	Amazon MSK は、Apache Kafka バージョン 2.5.1 をサポートするようになりました。Apache Kafka バージョン 2.5.1 では、Amazon MSK はクライアントと ZooKeeper エンドポイント間の転送中の暗号化をサポートしています。詳細については、「サポート対象の Apache Kafka バージョン」を参照してください。	2020-09-30
アプリケーションの自動拡張	使用量の増加に応じてクラスターのストレージを自動的に拡張するように、Amazon Managed Streaming for Apache Kafka を設定できます。詳細については、「クラスターの自動スケーリング」を参照してください。	2020-09-30
ユーザーネームとパスワードのセキュリティのサポート	Amazon MSK は、ユーザーネームとパスワードを使用したクラスターへのログインをサポートするようになりました。Amazon MSK は認証情報を AWS Secrets Manager に保存します。詳細については、「SASL/SCRAM 認証」を参照してください。	2020-09-17

変更	説明	日付
Amazon MSK クラスターの Apache Kafka バージョンのアップグレードのサポート	これで、既存の MSK クラスターの Apache Kafka バージョンをアップグレードできます。	2020 年 5 月 28 日
T3.small ブローカーノードのサポート	Amazon MSK は、Amazon EC2 タイプ T3.small のブローカーを使用したクラスターの作成をサポートするようになりました。	2020 年 4 月 8 日
Apache Kafka 2.4.1 のサポート	Amazon MSK は、Apache Kafka バージョン 2.4.1 をサポートするようになりました。	2020-04-02
ストリーミングブローカーログのサポート	Amazon MSK は、ブローカーログを CloudWatch Logs、Amazon S3、および Amazon Data Firehose にストリーミングできるようになりました。さらに、Firehose は、OpenSearch Service などのサポートする配信先にこれらのログを配信できます。	2020-02-25
Apache Kafka 2.3.1 のサポート	Amazon MSK は、Apache Kafka バージョン 2.3.1 をサポートするようになりました。	2019-12-19
オープンモニタリング	Amazon MSK は、Prometheus によるオープンモニタリングをサポートするようになりました。	2019-12-04

変更	説明	日付
Apache Kafka 2.2.1 のサポート	Amazon MSK は、Apache Kafka バージョン 2.2.1 をサポートするようになりました。	2019-07-31
一般提供	新機能には、タグ付けのサポート、認証、TLS暗号化、設定、およびブローカーストレージを更新する機能が含まれます。	2019-05-30
Apache Kafka 2.1.0 のサポート	Amazon MSK は、Apache Kafka バージョン 2.1.0 をサポートするようになりました。	2019-02-05

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。