



開発者ガイド

Amazon Managed Streaming for Apache Kafka



Amazon Managed Streaming for Apache Kafka: 開発者ガイド

Copyright © 2024 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 が所有していない他のすべての商標は、それぞれの所有者の所有物であり、Amazon と関連会社、接続、または後援されている場合とされていない場合があります。

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
開始	5
ステップ 1: クラスターを作成する	5
ステップ 2: IAM ロールを作成する	6
ステップ 3: クライアントマシンを作成する	8
ステップ 4: トピックを作成する	9
ステップ 5: データを生成および消費する	12
ステップ 6: メトリクスを表示する	13
ステップ 7: リソースを削除する	13
仕組み	15
クラスターの作成	15
ブローカーサイズ	16
を使用したクラスターの作成 AWS Management Console	17
を使用したクラスターの作成 AWS CLI	19
を使用したカスタム Amazon MSK 設定でのクラスターの作成 AWS CLI	21
API を使用したクラスターの作成	22
クラスターの削除	22
を使用したクラスターの削除 AWS Management Console	22
を使用したクラスターの削除 AWS CLI	22
API を使用したクラスターの削除	22
ブートストラップブローカーの取得	23
を使用したブートストラップブローカーの取得 AWS Management Console	23
を使用したブートストラップブローカーの取得 AWS CLI	23
API を使用したブートストラップブローカーの取得	24
クラスターの一覧表示	24
を使用したクラスターの一覧表示 AWS Management Console	24
を使用したクラスターの一覧表示 AWS CLI	24
API を使用したクラスターの一覧表示	24
メタデータ管理	25
ZooKeeper モード	25

KRaft モード	27
ストレージ管理	28
階層型ストレージ	29
ブローカーストレージのスケールアップ	38
プロビジョニングストレージのスループット	42
ブローカーサイズの更新	46
を使用したブローカーサイズの更新 AWS Management Console	47
を使用したブローカーサイズの更新 AWS CLI	47
API を使用したブローカーサイズの更新	49
クラスターの設定の更新	49
を使用したクラスターの設定の更新 AWS CLI	49
API を使用したクラスターの設定の更新	52
クラスターの拡張	52
を使用したクラスターの拡張 AWS Management Console	52
を使用したクラスターの拡張 AWS CLI	53
API を使用したクラスターの拡張	54
ブローカーを削除する	54
ブローカーパーティションを削除する	55
コンソールでブローカーを削除する	58
CLI を使用してブローカーを削除する	58
API を使用してブローカーを削除する	59
セキュリティの更新	59
を使用したクラスターのセキュリティ設定の更新 AWS Management Console	60
を使用したクラスターのセキュリティ設定の更新 AWS CLI	61
API を使用したクラスターのセキュリティ設定の更新	62
クラスターのブローカーの再起動	63
を使用したブローカーの再起動 AWS Management Console	63
を使用したブローカーの再起動 AWS CLI	63
API を使用したブローカーの再起動	63
パッチ適用	65
クラスターのタグ付け	65
タグの基本	66
タグ付けを使用したコストの追跡	66
タグの制限	67
Amazon MSK API を使用したリソースのタグ付け	67
構成	68

カスタム 設定	68
動的設定	78
トピックレベルの設定	78
状態	78
デフォルト設定	79
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	90
設定オペレーション	91
設定を作成する	92
MSK 設定を更新するには	93
MSK 設定を削除するには	93
MSK 設定を説明するには	94
MSK 設定リビジョンの説明	94
現在のリージョンのアカウントにあるすべての MSK 設定を一覧表示するには	96
MSK サーバーレス	98
入門チュートリアル	99
ステップ 1: クラスターを作成する	100
ステップ 2: IAM ロールを作成する	101
ステップ 3: クライアントマシンを作成する	103
ステップ 4: トピックを作成する	105
ステップ 5: データを生成および消費する	106
ステップ 6: リソースを削除する	106
構成	107
モニタリング	108
MSK Connect	111
MSK Connect とは何ですか？	111
開始	111
ステップ 1: 必要なリソースを設定する	112
ステップ 2: カスタム プラグインを作成する	116
ステップ 3: クライアントマシンと Apache Kafka トピックを作成する	117
ステップ 4: コネクタを作成する	119
ステップ 5: データを送信する	120
Connector	120
容量	121
コネクタの作成	122
プラグイン	124
ワーカー	124

デフォルトのワーカー構成	125
サポートされているワーカー設定プロパティ	125
カスタム設定を作成する	127
コネクタオフセットを管理する	128
設定プロバイダー	131
ステップ 1: カスタムプラグインを作成して S3 にアップロードする	132
ステップ 2: プロバイダーを設定する	134
ステップ 3: カスタムワーカー設定を作成する	138
ステップ 4: コネクタを作成する	139
考慮事項	140
IAM ロールとポリシー	140
サービス実行のロール	141
ポリシーの例	143
サービス間の混乱した代理の防止	145
AWS マネージドポリシー	147
サービスリンクロールの使用	150
インターネットアクセスを有効にする	152
Amazon MSK Connect 用の NAT ゲートウェイのセットアップ	153
プライベート DNS ホスト名	155
設定	156
DNS 属性	156
障害処理	156
ログ記録	157
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	158
モニタリング	159
例	161
Amazon S3 シンクコネクタ	161
Debezium ソースコネクタ	163
ベストプラクティス	173
コネクタからの接続	174
移行ガイド	174
Amazon MSK Connect の利点	174
移行中	175
トラブルシューティング	179
MSK レプリケーター	181
Amazon MSK Replicator とは	181

Amazon MSK Replicator の仕組み	182
Amazon MSK Replicator を作成するための要件と考慮事項	184
MSK レプリケーターの作成に必要なアクセス許可	184
サポートされるクラスタータイプとバージョン	185
MSK サーバーレスクラスター設定	186
クラスター設定の変更	186
入門チュートリアル	187
ステップ 1: Amazon MSK ソースクラスターを準備する	187
ステップ 2: Amazon MSK ターゲットクラスターを準備する	190
ステップ 3: Amazon MSK Replicator を作成する	190
MSK レプリケーター設定の編集	198
MSK レプリケーターの削除	200
レプリケーションのモニタリング	200
MSK レプリケーターのメトリクス	200
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	209
.....	209
.....	209
アクティブ/パッシブ Kafka クラスター設定の作成とレプリケートされたトピックの命名規則	210
セカンダリ AWS リージョンにフェイルオーバーするタイミング	210
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の実行	210
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の実行	211
プライマリ AWS リージョンへのフェイルバックの実行	212
MSK レプリケーターを使用したアクティブ/アクティブ設定の作成	214
MSK レプリケーターのトラブルシューティング	215
MSK レプリケーターの状態が CREATING から FAILED に変わります。	215
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	216
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	216
ターゲットクラスターのメッセージオフセットがソースクラスターと異なる	217
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同期していないか、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ません	217
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	218
MSK レプリケーターの使用に関するベストプラクティス	220
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	220

クラスターの保持期間の設定	221
クラスターの状態	222
セキュリティ	224
データ保護	225
暗号化	226
暗号化を開始する方法	227
Amazon MSK API の認証と認可	230
Amazon MSK と IAM の連携の仕組み	230
アイデンティティベースポリシーの例	235
サービスリンクロール	239
AWS マネージドポリシー	242
トラブルシューティング	250
Apache Kafka API の認証と認可	251
IAM アクセスコントロール	251
相互 TLS 認証	269
SASL/SCRAM 認証	274
Apache Kafka ACL	280
セキュリティグループの変更	281
Apache へのアクセスの制御 ZooKeeper	282
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	283
Apache での TLS セキュリティの使用 ZooKeeper	284
ログ記録	285
ブローカーログ	285
CloudTrail イベント	288
コンプライアンス検証	293
耐障害性	293
インフラストラクチャセキュリティ	294
MSK クラスターへの接続	295
パブリックアクセス	295
内からのアクセス AWS	299
Amazon VPC ピアリング	299
AWS Direct Connect	299
AWS Transit Gateway	300
VPN 接続	300
REST プロキシ	300
複数リージョンのマルチ VPC 接続	300

単一リージョンのマルチ VPC プライベート接続	300
EC2-Classic ネットワークは廃止されました	300
単一リージョンのマルチ VPC プライベート接続	301
ポート情報	315
移行	317
Apache Kafka クラスターを Amazon MSK に移行する	317
1 つの Amazon MSK クラスターから別のクラスターに移行する	318
MirrorMaker 1.0 のベストプラクティス	319
MirrorMaker 2.* の利点	320
クラスターのモニタリング	322
でモニタリングするための Amazon MSK メトリクス CloudWatch	322
DEFAULT レベルモニタリング	323
PER_BROKER レベルモニタリング	334
PER_TOPIC_PER_BROKER レベルモニタリング	343
PER_TOPIC_PER_PARTITION レベルモニタリング	345
を使用した Amazon MSK メトリクスの表示 CloudWatch	346
コンシューマーラグモニタリング	347
Prometheus によるオープンモニタリング	347
オープンモニタリングを有効にして Amazon MSK クラスターを作成する	348
既存の Amazon MSK クラスターのオープンモニタリングの有効化	348
Amazon EC2 インスタンスでの Prometheus ホストの設定	349
Prometheus メトリクス	352
Amazon Managed Service for Prometheus への Prometheus メトリクスの保存	353
Amazon MSK ストレージ容量アラート	353
Amazon MSK ストレージ容量アラートをモニタリングする	354
Cruise Control	355
Cruise Control	357
クォータ	358
Amazon MSK クォータ	358
MSK レプリケータークォータ	359
サーバーレスクラスターのクォータ	359
MSK Connect クォータ	361
リソース	362
MSK の統合	363
Athena	363
Redshift	363

Firehose	363
EventBridge パイプへのアクセス	364
Apache Kafka バージョン	366
サポート対象の Apache Kafka バージョン	366
Apache Kafka バージョン 3.7.x (本番環境に対応した階層型ストレージ搭載)	368
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)	368
Amazon MSK バージョン 3.5.1	369
Amazon MSK バージョン 3.4.0	369
Amazon MSK バージョン 3.3.2	369
Amazon MSK バージョン 3.3.1	370
Amazon MSK バージョン 3.1.1	370
Amazon MSK 階層型ストレージバージョン 2.8.2.tiered	370
Apache Kafka バージョン 2.5.1	370
Amazon MSK のバグ修正バージョン 2.4.1.1	371
Apache Kafka バージョン 2.4.1 (代わりに 2.4.1.1 を使用)	372
Amazon MSK バージョンのサポート	372
Amazon MSK バージョンサポートポリシー	373
Apache Kafka バージョンの更新	373
バージョンアップグレードのベストプラクティス	377
トラブルシューティング	378
ボリュームの置換により、レプリケーションの過負荷によるディスクの飽和が発生する	379
コンシューマーグループが PreparingRebalance 状態から変化しない	380
静的メンバーシッププロトコル	380
識別と再起動	381
Amazon CloudWatch Logs へのブローカーログの配信エラー	381
デフォルトのセキュリティグループがない	381
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	382
クラスターの状態が [作成中] から [失敗] に変わる	382
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	382
AWS CLI Amazon MSK を認識しない	382
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	383
ディスク容量が不足している	383
メモリが不足している	383
プロデューサーが取得する NotLeaderForPartitionException	383
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	383

クラスターには <code>__amazon_msk_canary</code> と <code>__amazon_msk_canary_state</code> というトピックがあります	384
パーティションレプリケーションが失敗する	384
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	384
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	385
同じ VPC 内の Amazon EC2 クライアントおよび MSK クラスター	386
異なる VPC 内の Amazon EC2 クライアントと MSK クラスター	386
オンプレミスクライアント	386
AWS Direct Connect	387
認証の失敗: 接続が多すぎる	387
MSK サーバーレス: クラスターの作成に失敗する	387
ベストプラクティス	388
クラスターの適切なサイズ設定: ブローカーあたりのパーティション数	388
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	389
m5.4xl、m7g.4xl 以上のインスタンスのクラスタースループットを最適化する	389
最新の Kafka AdminClient を使用してトピック ID の不一致の問題を回避する	391
高可用性クラスターの構築	391
CPU 使用率をモニタリングする	392
ディスク容量のモニタリング	393
データ保持パラメータの調整	394
不正シャットダウン後のログ復旧の高速化	394
Apache Kafka メモリのモニタリング	395
MSK 以外のブローカーを追加しない	395
転送中の暗号化を有効にする	395
パーティションの再割り当て	395
ドキュメント履歴	397
AWS 用語集	408
.....	cdix

Amazon MSK デベロッパーガイドへようこそ

Amazon MSK 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 Amazon MSK クラスターを作成する。
- [Amazon MSK: 仕組み](#) で Amazon MSK の機能を詳しく見る。
- [MSK サーバーレス](#) を使用してクラスター容量の管理やスケールリングなしで Apache Kafka を実行する。
- [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する。
- を使用して [MSK レプリケーター](#)、異なるリージョンまたは同じ AWS リージョンの Amazon MSK クラスター間でデータを確実にレプリケートします (複数可)。

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 クラスターの作成時に、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。
- 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サーバーです。

- KRaft コントローラー — Apache Kafka コミュニティは、Apache Kafka クラスターのメタデータ管理 ZooKeeper のために Apache を置き換えるために KRaft を開発しました。KRaft モードでは、クラスターメタデータは、ZooKeeper ノード間ではなく 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、追加のセットアップや管理は必要ありません。

 Note

MSK の Apache Kafka バージョン 3.7.x から、モードの代わりに KRaft ZooKeeper モードを使用するクラスターを作成できます。

- 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一ブローカー 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 を含む のすべてのサービスに AWS 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。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用して [ルートユーザーアクセスが必要なタスク](#) を実行してください。

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。を最新バージョン AWS CLI 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](#) <https://docs.aws.amazon.com/sdk-for-go/api/service/kafka/>、[Node.js](#)、[PHP](#)、[Python](#)、および [Ruby](#) 用の Amazon Web Services SDKs には、Amazon MSK サポートとサンプルが含まれています。 <https://aws.amazon.com/sdk-for-net/> <https://aws.amazon.com/sdk-for-python/>

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を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: Amazon MSK クラスターを作成する](#)
- [ステップ 2: 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: 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: Amazon MSK クラスターを作成する

「[Amazon MSK の使用を開始する](#)」のこのステップでは、Amazon MSK クラスターを作成します。

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** です。
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 [プロビジョンド] 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、[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。
- [クラスターを作成] を選択します。
- [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、「[クラスターの状態](#)」を参照してください。

次のステップ

[ステップ 2: IAM ロールを作成する](#)

ステップ 2: 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

- IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。
- ナビゲーションペインで [Policies] (ポリシー) を選択します。
- [ポリシーの作成] を選択します。
- 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。

region を、クラスターを作成した AWS リージョンのコードに置き換えます。[*Account-ID*] (アカウント ID) をお客様のアカウント ID に置き換えます。*MSKTutorialCluster* をクラスターの名前に置き換えます。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "kafka-cluster:Connect",
      "kafka-cluster:AlterCluster",
      "kafka-cluster:DescribeCluster"
    ],
    "Resource": [
      "arn:aws:kafka:region:Account-ID:cluster/MSKTutorialCluster/*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "kafka-cluster:*Topic*",
      "kafka-cluster:WriteData",
      "kafka-cluster:ReadData"
    ],
    "Resource": [
      "arn:aws:kafka:region:Account-ID:topic/MSKTutorialCluster/*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "kafka-cluster:AlterGroup",
      "kafka-cluster:DescribeGroup"
    ],
    "Resource": [
      "arn:aws:kafka:region:Account-ID:group/MSKTutorialCluster/*"
    ]
  }
]
```

セキュアポリシーを書き込む手順については、「[the section called “IAM アクセスコントロール”](#)」を参照してください。

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

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. [次へ: タグ] を選択します。
6. [次へ: レビュー] を選択します。
7. ロール名には、msk-tutorial-role など、わかりやすい名前を入力します。
8. Create role (ロールの作成) 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 (<https://console.aws.amazon.com/ec2/>) を開きます。
2. [Launch Instances] (インスタンスの起動) を選択します。
3. クライアントマシンの [名前] (**MSKTutorialClient** 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 **MSKKeyPair** 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. [詳細情報] セクションを展開し、「[ステップ 2: IAM ロールを作成する](#)」で作成した IAM ロールを選択します。
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、[セキュリティグループ] を選択します。「[the section called “ステップ 1: 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] を選択します。
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) を選択します。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: トピックを作成する](#)

ステップ 4: 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欠落している可能性があります。

MSK クラスターのバージョンを確認する方法

1. <https://eu-west-2.console.aws.amazon.com/msk/> に移動します
2. MSK クラスターを選択します。
3. 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。
4. 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トピックを作成するには

1. Amazon EC2 コンソール (<https://console.aws.amazon.com/ec2/>) を開きます。
2. ナビゲーションペインで、[インスタンス] を選択します。次に、「[ステップ 3: クライアントマシンを作成する](#)」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。
3. [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。
4. 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
```

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/{YOUR MSK VERSION}/kafka_2.13-{YOUR MSK VERSION}.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_2.13-{YOUR MSK VERSION}.tgz
```

7. kafka_2.13-{YOUR MSK VERSION}/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/v1.1.1/aws-msk-iam-auth-1.1.1-all.jar
```

8. kafka_2.13-{YOUR MSK VERSION}/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
```

9. <https://console.aws.amazon.com/msk/> で Amazon MSK コンソールを開きます。
10. 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。
11. View client information (クライアント情報の表示) を選択します。
12. 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が提供されます。次のステップに必要なブローカーエンドポイントは 1 つだけです。

13. 次のコマンドを実行し、**BootstrapServer###**を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 つに置き換えます。

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --create --bootstrap-server  
BootstrapServerString --command-config client.properties --replication-factor 3 --  
partitions 1 --topic MSKTutorialTopic
```

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.

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 の使用をスタートする](#)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。*BootstrapServerString* を、「[トピックの作成](#)」で取得したプレーンテキストの接続文字列に置き換えます。この接続文字列を取得する方法については、「[Amazon MSK クラスターで使用するブートストラップブローカーの取得](#)」を参照してください。

```
<path-to-your-kafka-installation>/bin/kafka-console-producer.sh --  
broker-list BootstrapServerString --producer.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。
4. 次のコマンドで、*BootstrapServerString* を前に保存したプレーンテキストの接続文字列に置き換えます。次に、コンソールコンシューマーを作成するために、クライアントマシンへの 2 番目の接続で次のコマンドを実行します。

```
<path-to-your-kafka-installation>/bin/kafka-console-consumer.sh --bootstrap-  
server BootstrapServerString --consumer.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、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。

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 の使用を開始するのこのステップでは、Amazon 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 CloudWatch

1. <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) 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、MSK TutorialCluster 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 (<https://console.aws.amazon.com/ec2/>) を開きます。

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. [削除] を選択します。

Amazon MSK: 仕組み

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

- [Amazon MSK クラスターの作成](#)
- [Amazon MSK クラスターの削除](#)
- [Amazon MSK クラスター用のブートストラップブローカーの取得](#)
- [Amazon MSK クラスターの一覧表示](#)
- [メタデータ管理](#)
- [ストレージ管理](#)
- [ブローカーサイズの更新](#)
- [Amazon MSK クラスター設定の更新](#)
- [Amazon MSK クラスターの拡張](#)
- [Amazon MSK クラスターからブローカーを削除する](#)
- [クラスターのセキュリティ設定の更新](#)
- [Amazon MSK クラスターのブローカーの再起動](#)
- [パッチ適用やその他のメンテナンス中のブローカーの再起動の影響](#)
- [Amazon MSK クラスターのタグ付け](#)

Amazon MSK クラスターの作成

Important

クラスターの作成後は Amazon MSK クラスターの VPC を変更できません。

Amazon MSK クラスターを作成する前に、Amazon Virtual Private Cloud VPC を準備し、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) リージョン内のそれぞれ異なる2つのアベイラビリティゾーンに、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。

ブローカーサイズ

Amazon MSK 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。Amazon MSK では、次のブローカーサイズがサポートされています。

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

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ます。M7g ブローカーは、同等の M5 インスタンスよりも少ない電力を消費します。

M7g Graviton ブローカーは、CDG (パリ)、CGK (ジャカルタ)、CPT (ケープタウン)、DXB (ドバイ)、HKG (香港)、KIX (大阪)、LHR (ロンドン)、MEL (メルボルン)、MXP (ミラノ)、OSU (米国東部)、PDT (米国西部)、TLV (テルアビブ)、YYC (カルガリー)、ZRH (チューリッヒ) の各リージョンでは使用できません。

MSK は、次の Kafka バージョンのいずれかを実行するクラスターで M7g ブローカーをサポートします。

- 2.8.2.階層化
- 3.3.2
- 3.4.0
- 3.5.1
- 階層型ストレージを使用した 3.6.0
- 3.7.x

• 3.7.x.kraft

M7g および M5 ブローカーは T3 ブローカーよりもベースラインスループットパフォーマンスが高く、本番ワークロードに推奨されます。M7g および M5 ブローカーは、ブローカーごとに T3 ブローカーよりも多くのパーティションを持つこともできます。本番稼働用のワークロードが多い場合やパーティション数が多い場合は、M7g または M5 ブローカーを使用します。M7g および M5 インスタンスサイズの詳細については、[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 ブローカーが本番稼働用ワークロードまたは重要なワークロードに十分かどうかを判断する proof-of-concept テストを実行することをお勧めします。T3 ブローカーサイズの詳細については、[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の詳細については、「」を参照してください [ベストプラクティス](#)。

を使用したクラスターの作成 AWS Management Console

このプロセスでは、カスタム作成オプ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作成する一般的なタスクについて説明します。MSK コンソールで他のオプションを選択して、サーバーレスクラスターを作成できます。

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. クラスター作成方法 で、カスタム作成 を選択します。
4. 一意で 64 文字以下のクラスター名を指定します。
5. クラスタータイプ で、プロビジョニングされた を選択します。これにより、ブローカーの数、ブローカーサイズ、クラスターストレージ容量を指定できます。
6. ブローカーで実行する Apache Kafka 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 MSK 機能の比較を確認するには、バージョン互換性の表示 を選択します。
7. 選択した Apache Kafka バージョンによっては、クラスターのメタデータモード : [ZooKeeper または KRaft](#) を選択できます。
8. 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ブローカーサイズを選択します。??? を参照してください。
9. ブローカーが分散されているゾーンの数を選択します。

10. 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 つのブローカーで、最大値は ZooKeeper ベースのクラスターの場合はクラスターごとに 30 のブローカー、[KRaft ベースのクラスターの場合はクラスターごとに 60 のブローカー](#)です。
11. クラスターに含める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。
12. 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できます。このオプションを有効にするには、x86 にブローカーサイズ (インスタンスサイズ) kafka.m5.4xlarge 以上、Graviton ベースのインスタンスに kafka.m7g.2xlarge 以上を選択します。[???](#) を参照してください。
13. EBS ストレージのみ、または階層型ストレージと EBS ストレージのいずれかのクラスターストレージモードオプションを選択します。
14. カスタムクラスター設定を作成して使用する場合 (またはクラスター設定が既に保存されている場合)、設定を選択します。それ以外の場合は、Amazon MSK のデフォルトのクラスター設定を使用してクラスターを作成できます。Amazon MSK の設定については、「[構成](#)」を参照してください。
15. [次へ] を選択します。
16. ネットワーク設定で、クラスターに使用する VPC を選択します。
17. 以前に選択した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。サブネットは異なるアベイラビリティーゾーンになければなりません。
18. 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 (クライアントマシンのセキュリティグループなど) を選択できます。共有されているセキュリティグループを指定する場合は、それら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。[Amazon MSK クラスターへの接続](#)。
19. [次へ] を選択します。
20. クラスターのアクセスコントロール方法と暗号化設定を選択して、クライアントとブローカー間で転送されるデータを暗号化します。詳細については、「[the section called “転送中の暗号化”](#)」を参照してください。
21. 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 “保管中の暗号化”](#)」を参照してください。
22. [Next (次へ)] を選択します。
23. 必要なモニタリングとタグを選択します。これにより、取得するメトリックのセットが決まります。詳細については、「[クラスターのモニタリング](#)」を参照してください。[Amazon](#)

[CloudWatch](#)、[Prometheus](#)、[ブローカーログ配信](#)、または[クラスタータグ](#)を選択し、次へを選択します。

24. クラスターの設定を確認します。前のコンソール画面に戻るには前の、特定のクラスター設定を変更するには編集を選択して、設定を戻して変更できます。設定が正しい場合は、クラスターの作成を選択します。
25. [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、「[クラスターの状態](#)」を参照してください。

を使用したクラスターの作成 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

米国西部 (北カリフォルニア) リージョンを使用している場合は、2つのサブネットを正確に指定します。Amazon MSK が使用可能な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。指定するサブネットは、個別のアベイラビリティゾーンに存在する必要があります。クラスターを作成すると、Amazon MSK は、指定したサブネット上にブローカーノードを均等に分散します。

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"` 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"
}
```

ℹ 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、「[クラスターの状態](#)」を参照してください。

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

を使用したカスタム Amazon MSK 設定でのクラスターの作成 AWS CLI

カスタム Amazon MSK 設定とその作成方法については、「[構成](#)」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。次に例を示します。

```
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"
}
```

API を使用したクラスターの作成

API を使用してクラスターを作成するには、「」を参照してください [CreateCluster](#)。

Amazon MSK クラスターの削除

Note

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が推奨されます。詳細については、「[Auto Scaling](#)」を参照してください。

を使用したクラスターの削除 AWS Management Console

1. <https://console.aws.amazon.com/msk/> 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。

を使用したクラスターの削除 AWS CLI

次のコマンドを実行し、 をクラスターの作成時に取得した Amazon リソースネーム (ARN) *ClusterArn* に置き換えます。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かります。詳細については、「[the section called “クラスターの一覧表示”](#)」を参照してください。

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

API を使用したクラスターの削除

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

Amazon MSK クラスター用のブートストラップブローカーの取得

を使用したブートストラップブローカーの取得 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. <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。

を使用したブートストラップブローカーの取得 AWS CLI

次のコマンドを実行し、 をクラスターの作成時に取得した Amazon リソースネーム (ARN) *ClusterArn* に置き換えます。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かります。詳細については、「[the section called “クラスターの一覧表示”](#)」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 を使用したブートストラップブローカーの取得

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、[GetBootstrap「ブローカー」](#) を参照してください。

Amazon MSK クラスターの一覧表示

を使用したクラスターの一覧表示 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。詳細を表示するには、クラスターの名前を選択します。

を使用したクラスターの一覧表示 AWS CLI

以下のコマンドを実行します。

```
aws kafka list-clusters
```

API を使用したクラスターの一覧表示

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

メタデータ管理

Amazon MSK は、Apache ZooKeeper または KRaft メタデータ管理モードをサポートしています。

Amazon MSK の Apache Kafka バージョン 3.7.x から、モードの代わりに KRaft ZooKeeper 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ために Kafka 内のコントローラーに依存しています。

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、命名、分散同期の提供、グループサービスの提供のための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、以下のステップを使用して Apache ZooKeeper 接続文字列を取得できます。ただし、を使用してクラスターBootstrapServerStringに接続し、Kafka 2.5 で `--zookeeper` フラグが廃止され、Kafka 3.0 から削除されたため、管理者オペレーションを実行することをお勧めします。

を使用した Apache ZooKeeper 接続文字列の取得 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。

を使用した Apache ZooKeeper 接続文字列の取得 AWS CLI

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 “クラスターの一覧表示”](#)」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、次のコマンドを実行し、 をクラスターの ARN *ClusterArn*に置き換えます。

```
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",
    "ZookeeperConnectionString": "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 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、「」を参照してください [DescribeCluster](#)。

KRaft モード

Amazon MSK は、Kafka バージョン 3.7.x で KRaft (Apache Kafka Raft) のサポートを導入しました。Apache Kafka コミュニティは、[Apache Kafka ZooKeeper](#) クラスターのメタデータ管理のために Apache を置き換えるために KRaft を開発しました。KRaft モードでは、クラスターメタデータは、ZooKeeper ノード間ではなく 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、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 モードについて注意すべき点を以下に示します。

- KRaft モードは、新しいクラスターでのみ使用できます。クラスターの作成後に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、Kafka バージョン 3.7.x を選択し、クラスター作成ウィンドウで KRaft チェックボックスを選択することで、Kraft ベースのクラスターを作成できます。
- MSK API [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、バージョン 3.7.x.kraft としてを使用する必要があります。をバージョン 3.7.x 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 および 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、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。](#)
- 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 バージョン 2.5 では非推奨となり、Kafka バージョン 3.0 以降では削除されます。そのため、クラスターへのすべての `--bootstrap-server` 接続には、最新の Apache Kafka クライアントバージョンと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 が Apache Kafka でもサポートされているすべての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。
- 使用するツールが、ZooKeeper 接続なしで Kafka Admin APIs 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 に接続するための更新された手順 [Amazon MSK での LinkedIn の Cruise Control for Apache Kafka の使用](#) については、「」を参照してください。Cruise Control には、[を使用せずに Cruise Control を実行する ZooKeeper](#) 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以外のメトリクスを収集するために、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のオープンモニタリングを設定する手順の更新 [Prometheus によるオープンモニタリング](#) については、「」を参照してください。
- モードクラスターで KRaft モード ZooKeeper クラスター [CloudWatch](#) をモニタリングする必要は他にありません。MSK は、クラスターで使用される KRaft コントローラーを管理します。
- `--bootstrap-server` 接続文字列を使用して、KRaft モードのクラスターで `--zookeeper` 接続文字列を使用して ACLs を管理し続けることができます。`--zookeeper` 接続文字列 を使用して ACLs を管理しないでください。[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 ZooKeeper ノードには保存されません。したがって、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 ZooKeeper](#)。

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。

トピック

- [階層型ストレージ](#)
- [ブローカストレージのスケールアップ](#)
- [プロビジョニングストレージのスループット](#)

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ルアップできるため、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。

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ルアップできます。Apache Kafka インフラストラクチャをスケールアップする方法を考える必要はありません。
- ブローカ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。

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: 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。

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。

- 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。
- 階層型ストレージはブローカーサイズ t3.small をサポートしていません。
- 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の cleanup.policy が 'DELETE' のみに設定されていることを確認してください。
- 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は、別の階層型ストレージ対応 Apache Kafka バージョンにのみ移行できます。階層型ストレージでサポートされているバージョンを引き続き使用しない場合は、新しい MSK クラスターを作成し、そのクラスターにデータを移行します。
- この kafka-log-dirs ツールは階層型ストレージのデータサイズをレポートできません。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。

ログセグメントを階層型ストレージにコピーする方法

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。

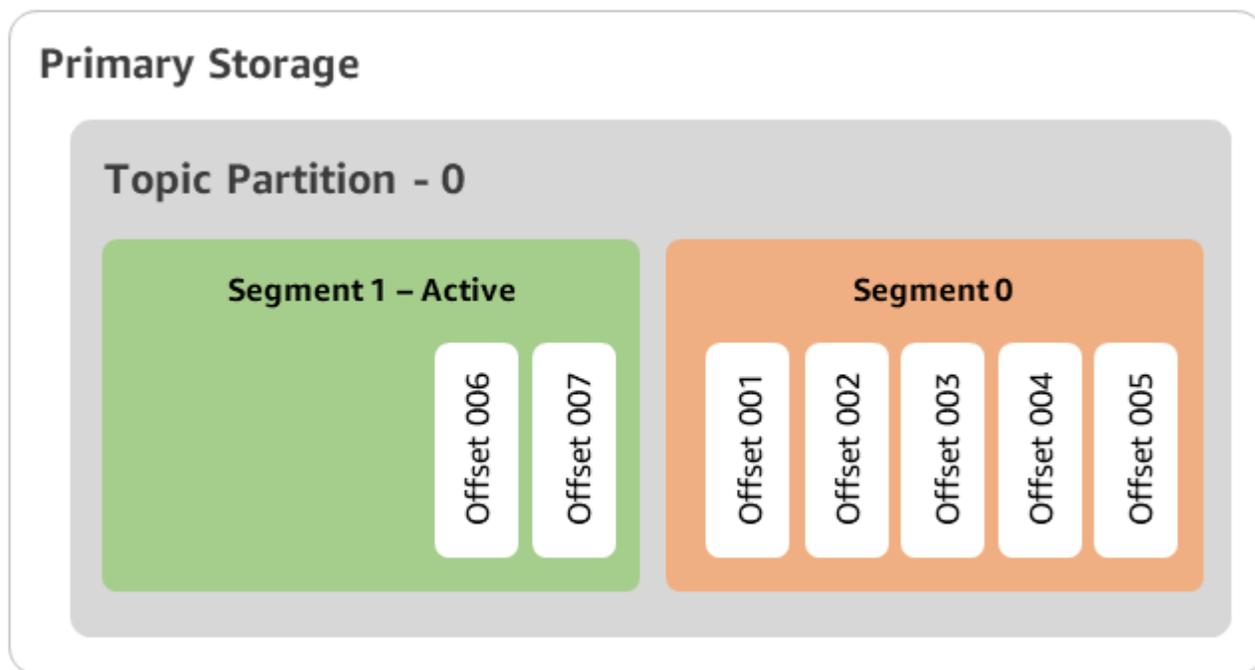
- Apache Kafka では、log.retention.ms (時間) と log.retention.bytes (サイズ) という 2 つの設定で保持を定義します。これらの設定によって、Apache Kafka がクラスターに保持するデータの合計期間とサイズが決まります。階層型ストレージモードを有効にするかどうかにかかわらず、これらの設定はクラスターレベルで設定します。トピックレベルの設定はトピック設定でオーバーライドできます。
- 階層型ストレージを有効にすると、プライマリの高性能ストレージ階層にデータを保存する期間を追加で指定できます。例えば、トピックの全体保持期間 (log.retention.ms) 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せん。
- 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。

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。

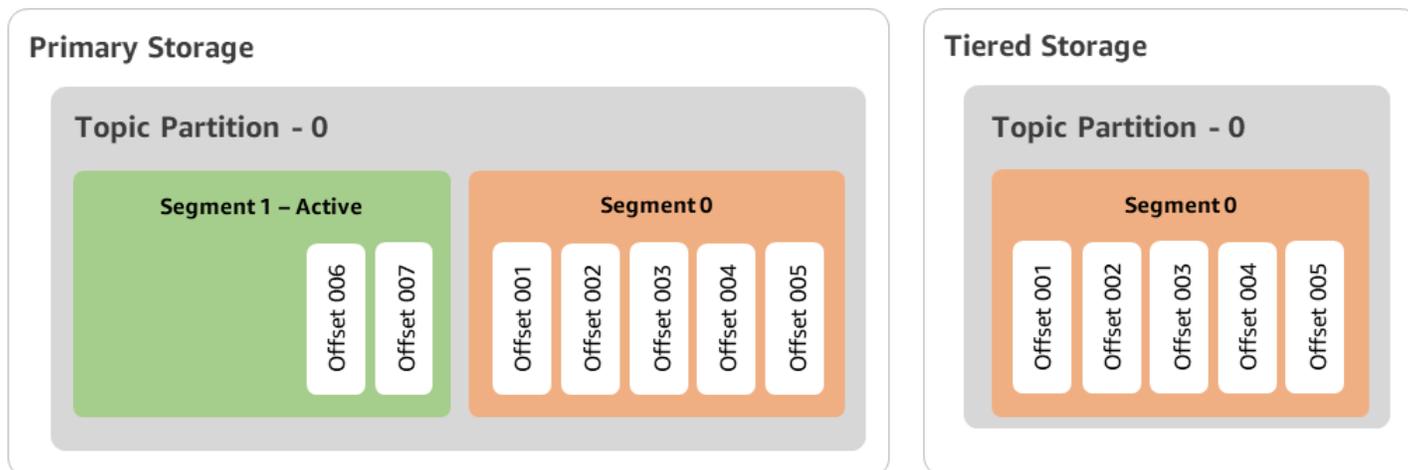
Time T0 - 階層型ストレージを有効にする前。

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。



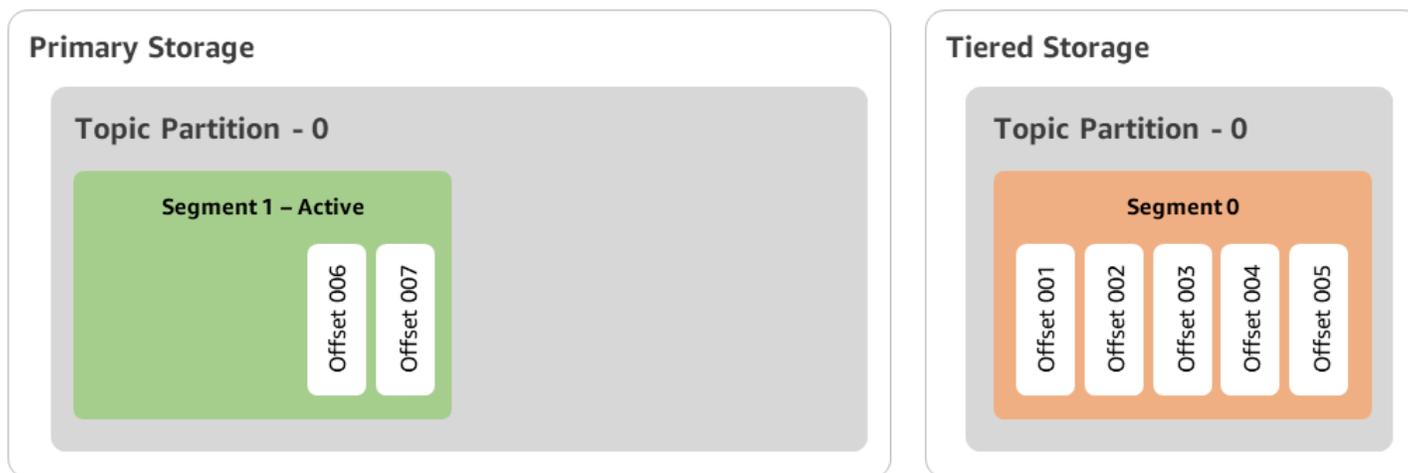
Time T1 (2 日未満) - 階層型ストレージが有効。セグメント 0 が階層型ストレージにコピーされます。

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、Amazon MSK は、セグメント 0 とセグメント 1 のメッセージにまだ保持設定を適用していません (local.retention.bytes/ms、retention.ms/bytes)。



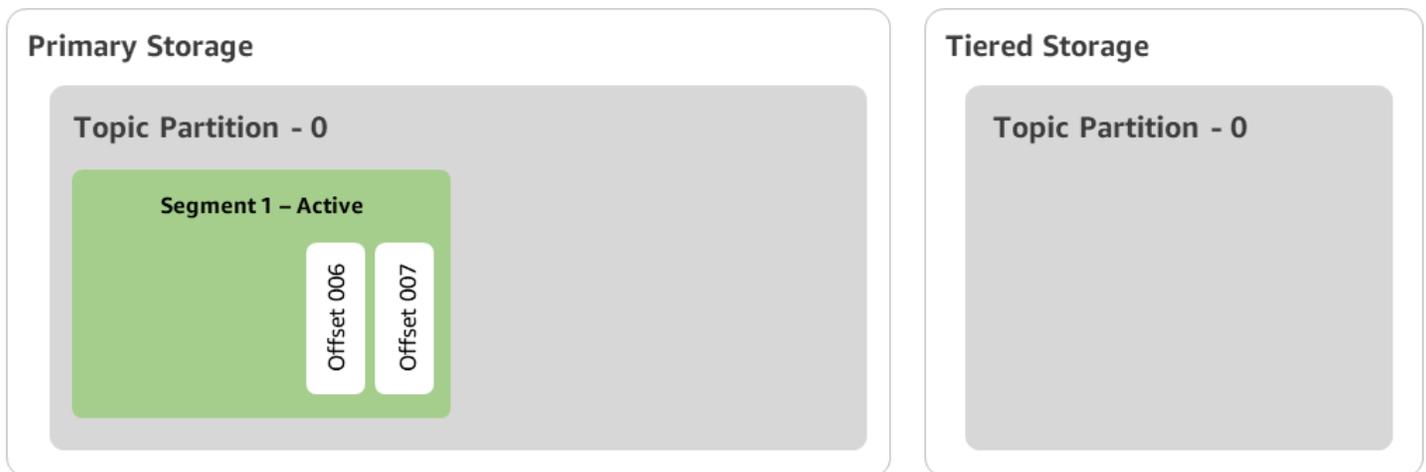
Time T2 - ローカル保持が有効。

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。



を使用した階層型ストレージを使用した Amazon MSK クラスターの作成 AWS Management Console

1. <https://console.aws.amazon.com/msk/> で Amazon MSK コンソールを開きます。
2. 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 以外のブローカーのサイズを指定します。
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指定します。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。

を使用した階層型ストレージを使用した Amazon MSK クラスターの作成 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「[階層型ストレージ対応の Kafka トピックの作成](#)」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 tiered-storage 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"
```

階層型ストレージ対応の Kafka トピックの作成

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)、またはロ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
```

既存のトピックでの階層型ストレージの有効化と無効化

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=1555000000'
```

既存のトピックでの階層型ストレージの無効化

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 `KTVDPKIKX0DER` 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た既存のクラスターでの階層型ストレージの更新

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。

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 クラスタ用にストレージをスケールアップすると、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ルアップイベントが発生するたびに、クラスタ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を最適化します。その後、クラスタは再びスケールアップできるようになります。この期間は、クラスタ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、[UpdateBrokerストレージ](#) オペレーションを使用した自動スケールアップイベントと手動スケールアップ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、「[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。「[階層型ストレージ](#)」を参照してください。

トピック

- [Auto Scaling](#)
- [手動スケールアップ](#)

Auto Scaling

使用量の増加に応じてクラスタ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ルアップを設定できます。自動スケールアップポリシーでは、ディスク使用率のターゲットと最大スケールアップ容量を設定します。

Amazon MSK の自動スケールアップを使用する前に、次の点を考慮する必要があります。

⚠ Important

ストレージスケールアップアクションは、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。クラスタの適切なサイズ設定のガイダンスについては、「[クラスタの適切なサイズ設定: クラスタあたりのブローカー数](#)」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、既存のクラスタを小さいストレージのクラスタに移行してください。クラスタの移行については、「[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ルアップをサポートしていません。
- 自動スケールアップポリシーをクラスタに関連付けると、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。自動スケールアップポリシーを使用してクラスタを削除すると、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、クラスタを削除する前にクラスタから自動スケールアップ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ルアップポリシー](#)」を参照してください。

自動スケールアップポリシーの詳細

自動スケールアップポリシーでは、クラスタ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 がオートスケールアップ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ルアップの上限。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。さらにスケールアップ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。

自動スケールアップオペレーションが発生したかどうかを判断するには、[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ルアップの設定

Amazon MSK コンソール、Amazon MSK API、または [awscli](#) を使用して AWS CloudFormation、ストレージの自動スケールアップを実装できます。CloudFormation サポートは [こちら](#) から利用できます [Application Auto Scaling](#)。

Note

クラスターの作成時に自動スケールアップ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ルアップ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せん。

AWS Management Consoleを使用した自動スケールアップの設定

1. <https://console.aws.amazon.com/msk/home?region=us-east-1#/home/> 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ルアップ) セクションで、[Configure] (設定) を選択します。
4. オートスケールアップ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] を選択します。

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 を使用した自動スケーリングの設定

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 を使用した自動スケーリングの設定

1. [RegisterScalableTarget](#) API 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](#) API を使用して自動拡張ポリシーを作成します。

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. [変更の保存] を選択します。

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

次のコマンドを実行し、 をクラスターの作成時に取得した Amazon リソースネーム (ARN) `ClusterArn` に置き換えます。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 “クラスターの一覧表示”](#)」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 `KTVDPKIKX0DER` 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、[UpdateBroker 「ストレージ」](#) を参照してください。

プロビジョニングストレージのスループット

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は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` 以上のクラスターとストレージボリュームが 10 GiB 以上のクラスターでは、プロビジョニングされたスループットレートを 1 秒あたり MiB 単位で指定できます。GiB 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。

スループットのボトルネック

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、ブローカーサイズが大きいほど、Amazon EC2 から Amazon EBS へのネットワークスループットと Amazon EC2 の出力スループットの両方が高くなります。

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 kafka.m5.4xlarge 以上 (または kafka.m7g.2xlarge 以上) を選択し、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

ストレージスループットの測定

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。

設定の更新

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。

を使用したストレージスループットのプロビジョニング 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. 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] をクリックします。
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。

を使用したストレージスループットのプロビジョニング AWS CLI

このセクションでは、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 を使用したストレージスループットのプロビジョニング

クラスターの作成中にプロビジョニングされたストレージスループットを設定するには、[CreateClusterV2](#) を使用します。

ブローカーサイズの更新

Apache Kafka パーティションを再割り当てせずに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でスケーリングできます。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 は、特定のクラスター内のすべてのブローカーに同じブローカーサイズを使用します。

このセクションでは、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。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。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、ブローカーごとに達成可能な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、クラスターの起動中および実行中にローリング方式で行われます。つまり、Amazon MSK はブローカーサイズの更新を実行するために一度に 1 つのブローカーをダウンします。ブローカーサイズの更新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”](#)。生産性への潜在的な影響をさらに軽減するために、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できます。

ブローカーサイズの更新中、引き続きデータを生成して使用できます。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なブローカーサイズに更新する場合は、最初にテストクラスターで更新を試して、シナリオにどのように影響するかを確認することをお勧めします。

Important

ブローカーあたりのパーティション数が指定された最大数を超えた場合、クラスターを小さなブローカーサイズに更新することはできません [the section called “クラスターの適切なサイズ設定: ブローカーあたりのパーティション数”](#)。

を使用したブローカーサイズの更新 AWS Management Console

1. <https://console.aws.amazon.com/msk/> で Amazon MSK コンソールを開きます。
2. 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を見つけ、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。

を使用したブローカーサイズの更新 AWS CLI

1. 次のコマンドを実行し、をクラスターの作成時に取得した Amazon リソースネーム (ARN) `ClusterArn` に置き換えます。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 “クラスターの一覧表示”](#)」を参照してください。

Current-Cluster-Version をクラスターの最新バージョン *TargetType* に置き換え、をブローカーにする新しいサイズに置き換えます。ブローカーサイズの詳細については、「」を参照してください [the section called “ブローカーサイズ”](#)。

```
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-1:0123456789012: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、次のコマンドを実行し、*ClusterOperationArn* を `update-broker-type` コマンドの出力で取得した ARN に置き換えます。

```
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、[UpdateBroker 「タイプ」](#) を参照してください。

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

Amazon MSK クラスター設定の更新

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: 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、「[構成](#)」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。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)

*ConfigurationArn*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. 次のコマンドを実行し、*ClusterArn* をクラスターの作成時に取得した ARN *ClusterArn* に置き換えます。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かります。詳細については、「[the section called “クラスターの一覧表示”](#)」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 `K1X5R6FKA87` 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、次のコマンドを実行し、`ClusterOperationArn` を `update-cluster-configuration` コマンドの出力で取得した ARN に置き換えます。

```
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 を使用したクラスターの設定の更新

API を使用してクラスターの設定を更新するには、[UpdateCluster 「設定」](#) を参照してください。

Amazon MSK クラスターの拡張

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。

を使用したクラスターの拡張 AWS Management Console

1. <https://console.aws.amazon.com/msk/> で Amazon MSK コンソールを開きます。
2. ブローカー数を増やしたい 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、[Cluster-Level Broker Details] (クラスターレベルのブローカーの詳細) の見出しの隣にある[Edit] (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に入力したら、[Save changes] (変更を保存) を選択します。

を使用したクラスターの拡張 AWS CLI

1. 次のコマンドを実行し、 をクラスターの作成時に取得した Amazon リソースネーム (ARN) *ClusterArn* に置き換えます。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れられます。詳細については、「[the section called “クラスターの一覧表示”](#)」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 `KTVDPKIKX0DER` 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、次のコマンドを実行し、*ClusterOperationArn* を `update-broker-count` コマンドの出力で取得した ARN に置き換えます。

```
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 を使用したクラスターの拡張

API を使用してクラスター内のブローカーの数を増やすには、[UpdateBroker 「カウント」](#) 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) で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。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、一連のブローカーを削除することで、クラスターのストレージとコンピューティングの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を処理し、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と、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 [AWS CLI 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 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は、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、1 回のブローカー削除オペレーションで、2 つのアベイラビリティーゾーンクラスターから 2 つのブローカーを削除したり、3 つのアベイラビリティーゾーンクラスターから 3 つのブローカーを削除したりできます。

クラスターからブローカーを削除した後でパーティションを再調整する方法については、「[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、すべての M5 および M7g ベースの MSK プロビジョニング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でブローカーを削除する](#)
- [AWS CLI 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

ブローカーの削除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 `__amazon_msk_canary_state` からトピック `__amazon_msk_canary` と `__amazon_msk_canary` を除くすべてのパー

パーティションを移動します。これらは、Amazon MSK がクラスターのヘルスと診断メトリクス用に作成する内部トピックです。

Kafka 管理 APIs 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、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仮定します。

- `__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の作成」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、、、`__amazon_msk_canary`、の 4 つのトピック名が表示されます `__consumer_offsets` `__amazon_msk_connect_offsets_my-mskc-connector_12345678-09e7-c657f7e4ff32-2` `msk-brk-rmv`。

3. クライアントマシン `topics.json` に という名前の 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トピック名を追加します。`__amazon_msk_canary` トピック名を含める必要はありません。これは、必要に応じて自動的に移動されるサービスマネージドトピックであるためです。

```
{
  "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 つのブローカーにのみ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. という名前のファイル `reassignment-file.json` を作成し、上記のコマンドから `proposed partition reassignment configuration` 取得したものをコピーします。
6. 次のコマンドを実行して、で指定したパーティションを移動します `reassignment-file.json`。

```
./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. ステータスが各パーティションのパーティションの再割り当てが完了したことを示したら、`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0に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で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、ブローカー数の編集オプションを選択します。
4. アベイラビリティーゾーンごとにクラスターに含めるブローカーの数を入力します。コンソールは、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を要約します。これを希望どおりに実行してください。
5. [変更の保存] を選択します。

ブローカーが誤って削除されないように、ブローカーを削除するかどうかを確認するメッセージが表示されます。

AWS CLI を使用してブローカーを削除する

次のコマンドを実行し、 をクラスターの作成時に取得した Amazon リソースネーム (ARN) ClusterArnに置き換えます。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 クラスターの一覧表示」を参照してください](#)。 をクラスターの最新バージョンCurrent-Cluster-Version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 KTVDPKIKX0DER 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も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 Reference」の [UpdateBroker「Count」](#) を参照してください。

クラスターのセキュリティ設定の更新

この Amazon MSK オペレーションでは、お客様の MSK クラスターにおける認証とクライアントブローカーの暗号化設定を更新することができます。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、「[セキュリティ](#)」を参照してください。

を使用したクラスターのセキュリティ設定の更新 AWS Management Console

1. <https://console.aws.amazon.com/msk/> で Amazon MSK コンソールを開きます。
2. 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。次に例を示します。

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。次に例を示します。

```
{"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 オペレーションのステータスを確認するには、次のコマンドを実行し、*ClusterOperationArn* を update-security コマンドの出力で取得した ARN に置き換えます。

```
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。

API を使用したクラスターのセキュリティ設定の更新

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

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。

Amazon MSK クラスターのブローカーの再起動

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。

を使用したブローカーの再起動 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。

を使用したブローカーの再起動 AWS CLI

1. をクラスターの作成時に取得した Amazon リソースネーム (ARN) *ClusterArn* に、 を再起動するブローカーの ID *BrokerId* 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かります。詳細については、「[the section called “クラスターの一覧表示”](#)」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かります。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、次のコマンドを実行し、`reboot-broker` コマンドの出力で取得した ARN `ClusterOperationArn` に置き換えます。

```
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 を使用したブローカーの再起動

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、「」を参照してください [RebootBroker](#)。

パッチ適用やその他のメンテナンス中のブローカーの再起動の影響

Amazon MSK はブローカーのソフトウェアを定期的に更新します。これらの更新は、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 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 は自動的にリーダーシップを別のオンラインブローカーに移行します。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化を自動的に検出し、MSK クラスターへのデータの書き込みと読み取りを続行するメカニズムが組み込まれています。

ブローカーがオフラインになると、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、短いウィンドウ (最大 2 分、通常はそれ未満) で、p99 の読み取りおよび書き込みレイテンシー (通常はミリ秒単位の高レイテンシー、最大 2 秒) が急増するのを観察します。これらのスパイクは予想され、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発生します。生産や消費には影響せず、再接続後に解決されます。

また、シャットダウンされたブローカーのパーティション `UnderReplicatedPartitions` がデータをレプリケートしていないことが予想されるメトリクスの増加も見られます。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がリクエストを処理するようになったため、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。

ソフトウェアの更新後、ブローカーがオンラインに戻ると、オフライン中に生成されたメッセージに「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 の使用量が増加することもあります。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。

Amazon MSK クラスターのタグ付け

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

- [タグの基本](#)
- [タグ付けを使用したコストの追跡](#)
- [タグの制限](#)
- [Amazon MSK API を使用したリソースのタグ付け](#)

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、特定のニ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

タグ付けを使用したコストの追跡

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 を使用したリソースのタグ付け

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 設定

Amazon Managed Streaming for Apache Kafka は、ブローカー、トピック、および Apache ZooKeeper 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。

トピック

- [カスタム MSK 設定](#)
- [Amazon MSK のデフォルト設定](#)
- [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 設定オペレーション](#)

カスタム 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
<code>allow.everyone.if.no.acl.found</code>	このプロパティを <code>false</code> 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 <code>false</code> 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 <code>true</code> 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。
<code>auto.create.topics.enable</code>	サーバーでトピックの自動作成を有効にします。

名前	説明
<code>compression.type</code>	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、標準の圧縮コーデック (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。
<code>connections.max.idle.ms</code>	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。
<code>default.replication.factor</code>	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。
<code>delete.topic.enable</code>	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。
<code>group.initial.rebalance.delay.ms</code>	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。処理が開始されるまでの時間が長くなります。
<code>group.max.session.timeout.ms</code>	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。

名前	説明
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。
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。

名前	説明
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。
log.message.timestamp.difference.max.ms	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= の場合CreateTime、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。

名前	説明
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。
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	<p>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。</p> <p>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。</p> <p>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。</p>

名前	説明
min.insync.replicas	<p>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 (NotEnough Replicas または) を発生させます NotEnough ReplicasAfterAppend。</p> <p>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。</p>
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り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	<p>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。</p>
replica.lag.time.max.ms	<p>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。</p> <p>MinValue: 10,000</p> <p>MaxValue = 30,000</p>
replica.selector.class	<p>を実装する完全修飾クラス名 ReplicaSelector。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。</p>
replica.socket.receive.buffer.bytes	<p>ネットワークリクエスト用のソケット受信バッファ。</p>

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。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。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されたトランザクション時間がこの値を超えると、ブローカーはエラーを返します InitProducerIdRequest。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	<p>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。</p>
unclean.leader.election.enable	<p>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。</p>
zookeeper.connection.timeout.ms	<p>ZooKeeper モードクラスター。クライアントがへの接続を確立するのを待機する最大時間 ZooKeeper。この値を設定しない場合、zookeeper.session.timeout.ms の値が使用されます。</p> <p>MinValue = 6000</p> <p>MaxValue (包括的) = 18000</p>
zookeeper.session.timeout.ms	<p>ZooKeeper モードクラスター。Apache ZooKeeper セッションのミリ秒単位のタイムアウト。</p> <p>MinValue = 6000</p> <p>MaxValue (包括的) = 18000</p>

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 “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、「[仕組み](#)」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。

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

Amazon MSK のデフォルト設定

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。	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
default.replicatio n.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	<p>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、ローカルストレージ</p>	-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	<p>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。</p> <p>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</p>	-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。		
log.message.timest amp.difference.max .ms	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.timestam p.type= の場合Create Time、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.tim estamp.type=LogApp endTime 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。	922337203 6854775807	Kafka 2.8.2.tiered の 場合は 86400000
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	<p>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さない場合、プロデューサーは例外 (NotEnoughReplicas または) を発生させます NotEnoughReplicasAfterAppend。</p> <p>min.insync.replicas と acks の値を組み合わせると、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</p>	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
	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	DELETE
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.msck.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	true
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	<p>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。</p> <p>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.renti on.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。retenti on.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</p>	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.buf fer.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.ele ction.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 セッションのミリ秒 単位のタイムアウト 。	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 設定”](#)」を参照してください。

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テーション時間は 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。

Amazon MSK 設定オペレーション

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、「[仕組み](#)」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。

- [MSK 設定を作成するには](#)
- [MSK 設定を更新するには](#)
- [MSK 設定を削除するには](#)
- [MSK 設定を説明するには](#)
- [MSK 設定リビジョンの説明](#)
- [現在のリージョンのアカウントにあるすべての MSK 設定を一覧表示するには](#)

MSK 設定を作成するには

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。

MSK 設定を更新するには

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
  }
}
```

MSK 設定を削除するには

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"
}
```

MSK 設定を説明するには

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"
}
```

MSK 設定リビジョンの説明

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":
  "YXV0by5jcmVhdGUudG9waWNzLmVuYWJsZSA9IHRydWUKCgp6b29rZWVwZXIuY29ubmVjdGlvbi50aW11b3V0Lm1zI
}
```

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にあるすべての MSK 設定を一覧表示するには

- 以下のコマンドを実行します。

```
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"
    }
  ]
}
```

```
}
```

MSK サーバーレス

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ルするため、ユーザーはクラスターの適切なサイジングやスケールアップ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 言語と Java 以外の言語を使用した認証と認可。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 Schema Registry
- Apache Flink ベースのストリーム処理対応の Amazon Managed Service for Apache Flink
- AWS Lambda イベント処理用の

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のクォータ”](#)」を参照してください。

サーバーレスクラスタ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の利用を開始する](#)
- [サーバーレスクラスタの設定](#)
- [サーバーレスクラスタのモニタリング](#)

MSK サーバーレスクラスタの利用を開始する

このチュートリアルでは、MSK サーバーレスクラスタの作成方法と、作成したクラスタ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

- [ステップ 1: MSK サーバーレスクラスタの作成](#)
- [ステップ 2: 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: Apache Kafka 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: リソースを削除する](#)

ステップ 1: MSK サーバーレスクラスターの作成

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

[ステップ 2: IAM ロールを作成する](#)

ステップ 2: 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. [ポリシーの作成] 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。

region をクラスターを作成した AWS リージョンのコードに置き換えます。[*Account-ID*] (アカウント ID) をお客様のアカウント ID に置き換えます。をサーバーレスクラスターの名前 *msk-serverless-tutorial-cluster* に置き換えます。

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

```
        "kafka-cluster:Connect",
        "kafka-cluster:AlterCluster",
        "kafka-cluster:DescribeCluster"
    ],
    "Resource": [
        "arn:aws:kafka:region:Account-ID:cluster/msk-serverless-tutorial-  
cluster/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "kafka-cluster:*Topic*",
        "kafka-cluster:WriteData",
        "kafka-cluster:ReadData"
    ],
    "Resource": [
        "arn:aws:kafka:region:Account-ID:topic/msk-serverless-tutorial-  
cluster/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "kafka-cluster:AlterGroup",
        "kafka-cluster:DescribeGroup"
    ],
    "Resource": [
        "arn:aws:kafka:region:Account-ID:group/msk-serverless-tutorial-  
cluster/*"
    ]
}
]
```

セキュアポリシーを書き込む手順については、[「the section called “IAM アクセスコントロール”」](#)を参照してください。

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. [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. Create role (ロールの作成) 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

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] を選択します。
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、[ステップ 2: 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. 左側のナビゲーションペインで、[Instances] (インスタンス) をクリックします。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
```

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/v1.1.1/aws-msk-iam-auth-1.1.1-all.jar
```

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

[ステップ 4: Apache Kafka トピックを作成する](#)

ステップ 4: 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。

トピックを作成してデータを書き込む

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

[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

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。

次のステップ

[ステップ 6: リソースを削除する](#)

ステップ 6: 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 (<https://console.aws.amazon.com/ec2/>) を開きます。
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] をクリックします。

サーバーレスクラスターの設定

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、次のトピック構成プロパティを設定できます。

設定プロパティ	デフォルト値	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せん	
compression.type	プロデューサー	はい	
max.message.bytes	1048588	はい	8 MiB
message.timestamp.difference.max.ms	long.max	はい	
message.timestamp.type	CreateTime	はい	
retention.bytes	250 GiB	はい	250 GiB
retention.ms	7 日間	はい	無制限

Apache Kafka コマンドを使用して、新規または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または変更することもできます。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 の公式ドキュメントの「[トピックレベルの設定](#)」を参照してください。

サーバーレスクラスターのモニタリング

Amazon MSK は Amazon と統合 CloudWatch されているため、MSK Serverless クラスターのメトリクスを収集、表示、分析できます。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 CloudWatch で PerSec メトリクスを発行します。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 分以上の秒単位のデータを収集するには、数 CloudWatch 式を使用します $m1 * 60 / \text{PERIOD}(m1)$ 。

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 メトリクスの時間見積もり。
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。
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。

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。

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. [AWS/Kafka / Cluster Name, Topic] (AWS/Kafka / クラスター名、トピック) または [AWS/Kafka / Cluster Name, Consumer Group, Topic] (AWS/Kafka / クラスター名、コンシューマーグループ、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

MSK Connect とは何ですか？

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、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ある Kafka Connect 2.7.1 を使用します。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取得したりできます。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 “開始”](#)」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 “プラグイン”](#)」、「[the section called “Connector”](#)」を参照してください。[the section called “ワーカー”](#)。

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。

MSK Connect の使用を開始する

これは、を使用して MSK クラスターを作成し AWS Management Console、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

- [ステップ 1: 必要なリソースを設定する](#)

- [ステップ 2: カスタム プラグインを作成する](#)
- [ステップ 3: クライアントマシンと Apache Kafka トピックを作成する](#)
- [ステップ 4: コネクタを作成する](#)
- [ステップ 5: データを送信する](#)

ステップ 1: 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宛先として機能する 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。
- 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。
- 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、mkc-tutorial-destination-bucket 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 (クラスタの作成) を選択します。
4. Custom create (カスタム作成) を選択します。
5. クラスタ名には `mkc-tutorial-cluster` と入力します。
6. [一般的なクラスタのプロパティ] で、クラスタタイプに [プロビジョンド] を選択します。
7. [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。
8. 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。
9. 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。
10. ウィザードを続行し、[クラスタの作成] を選択します。そのクラスタ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。

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 <https://console.aws.amazon.com/iam/> を開きます。
2. 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。
3. Create role (ロールの作成) を選択します。
4. Or select a service to view its use cases (またはサービスを選択してそのユースケースを表示する) で、S3 を選択します。
5. 下にスクロールして、Select your use case (ユースケースの選択) でもう一度 S3 を選択します。
6. Next: Permissions (次へ : 許可) を選択します。
7. Create policy (ポリシーの作成) を選択します。これにより、ポリシーを作成する新しいタブがブラウザに開きます。後で戻るため、元のロール作成タブは開いたままにしておきます。
8. [JSON] タブを選択し、ウィンドウ内のテキストを次のポリシーに置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets"
      ]
    }
  ]
}
```

```
    ],
    "Resource": "arn:aws:s3:::*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:ListBucket",
      "s3:GetBucketLocation",
      "s3:DeleteObject"
    ],
    "Resource": "arn:aws:s3:::<my-tutorial-destination-bucket>"
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:GetObject",
      "s3:AbortMultipartUpload",
      "s3:ListMultipartUploadParts",
      "s3:ListBucketMultipartUploads"
    ],
    "Resource": "*"
  }
]
```

9. [次へ: タグ] を選択します。
10. Next: Review (次へ: レビュー) を選択します。
11. ポリシー名に `mkc-tutorial-policy` を入力し、下にスクロールして Create policy (ポリシーの作成) を選択します。
12. ロールを作成していたブラウザタブに戻り、更新ボタンを選択します。
13. `mkc-tutorial-policy` を見つけて、左側のボタンを選択して選択します。
14. Next: Tags (次へ: タグ) を選択します。
15. Next: Review (次へ: レビュー) を選択します。
16. ロール名に `mkc-tutorial-role` と入力し、説明ボックスのテキストを削除します。
17. Create role (ロールの作成) 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。

```
{
  "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

[ステップ 2: カスタム プラグインを作成する](#)

ステップ 2: 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に AWS 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

[ステップ 3: クライアントマシンと Apache Kafka トピックを作成する](#)

ステップ 3: 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 (<https://console.aws.amazon.com/ec2/>) を開きます。
2. [Launch Instances] (インスタンスの起動) を選択します。
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 “ステップ 1: 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 (ク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。

- 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、 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 `bootstrapServerString` 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

[ステップ4: コネクタを作成する](#)

ステップ4: コネクタを作成する

コネクタを作成するには

- 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。
- 左側のペインで、MSK Connect を展開し、Connectors (コネクタ) を選択します。
- Create connector (コネクタの作成) を選択します。
- 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。
- コネクタ名に mkc-tutorial-connector と入力します。
- クラスターのリストから、mkc-tutorial-cluster を選択します。
- 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。

```
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.partitionner.DefaultPartitioner
storage.class=io.confluent.connect.s3.storage.S3Storage
s3.bucket.name=<my-tutorial-destination-bucket>
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

[ステップ5: データを送信する](#)

ステップ5: 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=PLAINTEXT
```

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。を、前のコマンドを実行したときに取得した値 `BootstrapBrokerString` 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。

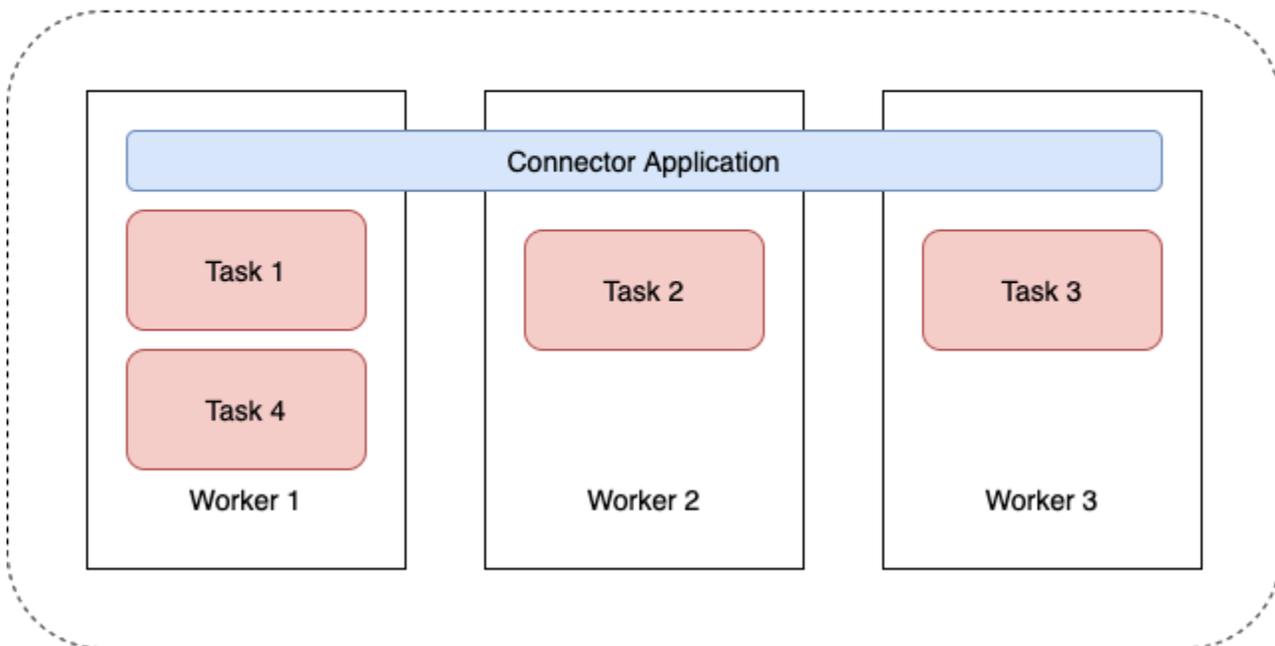
Connector

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。

Connector Architecture



コネクタ容量

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、お客様が用意したサブネットに IP アドレスを使用します。各ワーカーは、お客様が用意したサブネットの 1 つの IP アドレスを使用します。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ルリングする場合は、指定された容量を考慮して CreateConnector リクエストに提供されたサブネットに十分な使用可能な IP アドレスがあることを確認する必要があります。

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 “ワーカー”](#)」を参照してください。MSK Connect メトリクスについては、「[the section called “モニタリング”](#)」を参照してください。

コネクタの作成

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 “プラグイン”](#)」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。

- 名前と、オプションで説明を入力します。
- 接続するクラスターを選択します。
- コネクタ構成を指定します。指定する必要が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=my-destination-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.partitioners.DefaultPartitioner
key.converter=org.apache.kafka.connect.storage.StringConverter
value.converter=org.apache.kafka.connect.storage.StringConverter
schema.compatibility=NONE
```

- 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 “容量”](#)」を参照してください。
- 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 “ワーカー”](#)」を参照してください。
- 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 “サービス実行のロール”](#)」を参照してください。
- 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。
- 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 “ログ記録”](#)」を参照してください。
- 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、「」を参照してください [CreateConnector](#)。

プラグイン

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 one-to-many 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、「」を参照してください [CreateCustomPlugin](#)。

ワーカー

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 “容量”](#)」を参照してください。

MSK Connect ワーカーが IP アドレスを使用する

MSK Connect ワーカーは、お客様が用意したサブネットで IP アドレスを使用します。各ワーカーは、お客様が用意したサブネットの 1 つから 1 つの IP アドレスを使用します。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、指定された容量を考慮して CreateConnector リクエストに提供されたサブネットに十分な使用可能な IP アドレスがあることを確認する必要があります。

トピック

- [デフォルトのワーカー構成](#)
- [サポートされているワーカー設定プロパティ](#)
- [カスタムワーカー設定を作成する](#)
- [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](#)

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.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.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を作成する

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設定を作成するには、「」を参照してください [CreateWorkerConfiguration](#)。

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。

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` です。

独自のオフセットストレージトピックを指定する

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 クラスター用のブートストラップブローカーの取得](#)」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を作成する](#)」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=us-east-1
key.converter=<org.apache.kafka.connect.storage.StringConverter>
value.converter=<org.apache.kafka.connect.storage.StringConverter>
config.providers.secretManager.class=com.github.jcstenborder.kafka.config.aws.SecretsManager
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、「[コネクタの作成](#)」を参照してください。

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。

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、および 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。

トピック

- [ステップ 1: カスタムプラグインを作成して S3 にアップロードする](#)
- [ステップ 2: 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](#)
- [ステップ 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する](#)
- [ステップ 4: コネクタを作成する](#)
- [考慮事項](#)

ステップ 1: 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、ローカルマシンで次のコマンド `msk-config-provider` を実行して、コネクタとを含む 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

AWS CLI を使用して、AWS S3 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。AWS CLI のインストールとセットアップの詳細については、[ユーザーガイドの「AWS CLI の開始方法AWS Command Line Interface」](#)を参照してください。Amazon S3 で AWS CLI を使用する方法については、「[ユーザーガイド」の「CLI で Amazon S3 AWS を使用するAWS Command Line Interface」](#)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/r0.1.0/msk-config-providers-0.1.0-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-0.1.0-with-dependencies.zip
```

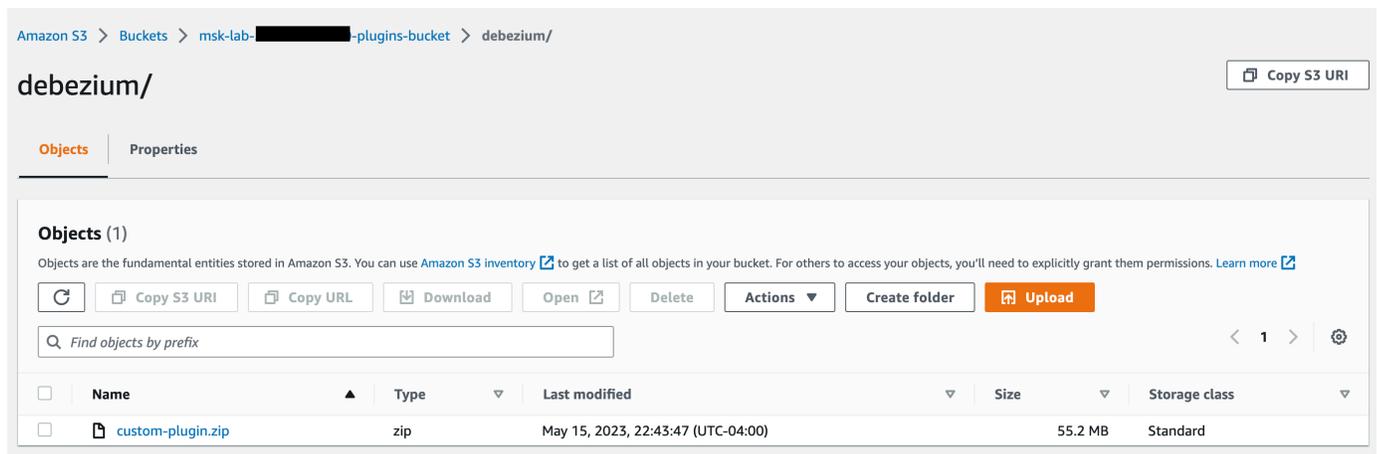
- 上記のステップで取得した MSK 設定プロバイダーとカスタムコネクタの内容を、custom-plugin.zip という名前を付けた単一のファイルに圧縮します。

```
zip -r ../custom-plugin.zip *
```

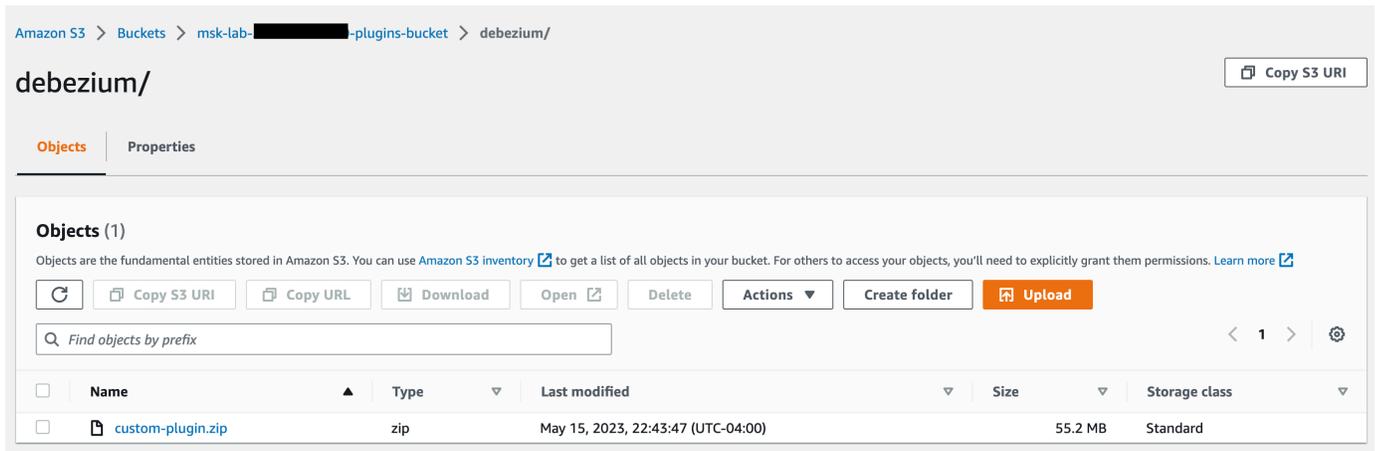
- このファイルを後で参照できるように S3 にアップロードします。

```
aws s3 cp ../custom-plugin.zip s3:<S3_URI_BUCKET_LOCATION>
```

- 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。



- 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。



ステップ 2: 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すると、そのサービスのパラメータと関連するアクセス許可の設定方法が表示されます。

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、「[ユーザーガイド](#)」の「[AWS Secrets Manager シークレットの作成](#) AWS Secrets Manager」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。 `<arn:aws:secretsmanager:us-east-1:123456789000:secret:MySecret-1234>` をシークレットの 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-1:123456789000: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.hostname=${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、「[ユーザーガイド](#)」の「[Systems Manager パラメータの作成 \(コンソール\)](#)」AWS Systems Manager」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 に置き換えます。

```
{
  "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。<*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/
truststore_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。

ステップ 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する

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。

ステップ 4: 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.hostname=${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/truststore_unique_filename.jks}
```

5. 「カスタム設定を使用する」を選択し、「ワーカー設定」ドロップダウンSourceDebeziumCustomConfigから選択します。
6. 「[Create connector](#)」の残りの手順に従います。

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。

MSK Connect の IAM のロールとポリシー

トピック

- [サービス実行のロール](#)
- [MSK Connect の IAM ポリシーの例](#)

- [サービス間の混乱した代理の防止](#)
- [AWS MSK Connect の マネージドポリシー](#)
- [MSK Connect でのサービスにリンクされたロールの使用](#)

サービス実行のロール

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 を使用してコネクタを作成する場合、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 “サービス間の混乱した代理の防止”](#)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kafkaconnect.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "Account-ID"
        },
        "ArnLike": {
          "aws:SourceArn": "MSK-Connector-ARN"
        }
      }
    }
  ]
}
```

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 を見つける方法と、トピック ARN を作成する方法については、[the section called “リソース”](#) を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kafka-cluster:Connect",
        "kafka-cluster:DescribeCluster"
      ],
      "Resource": [
        "cluster-arn"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "kafka-cluster:ReadData",
        "kafka-cluster:DescribeTopic"
      ],
      "Resource": [
        "ARN of the topic that you want a sink connector to read from"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "kafka-cluster:WriteData",
        "kafka-cluster:DescribeTopic"
      ],
      "Resource": [
        "ARN of the topic that you want a source connector to write to"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "kafka-cluster:CreateTopic",
        "kafka-cluster:WriteData",

```

```

        "kafka-cluster:ReadData",
        "kafka-cluster:DescribeTopic"
    ],
    "Resource": [
        "arn:aws:kafka:region:account-id:topic/cluster-name/cluster-uuid/__amazon_msk_connect_*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "kafka-cluster:AlterGroup",
        "kafka-cluster:DescribeGroup"
    ],
    "Resource": [
        "arn:aws:kafka:region:account-id:group/cluster-name/cluster-uuid/__amazon_msk_connect_*",
        "arn:aws:kafka:region:account-id:group/cluster-name/cluster-uuid/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がある場合もあります。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。

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",

```

```

    "Action": [
      "kafkaconnect:*",
      "ec2:CreateNetworkInterface",
      "ec2:DescribeSubnets",
      "ec2:DescribeVpcs",
      "ec2:DescribeSecurityGroups",
      "logs:CreateLogDelivery",
      "logs:GetLogDelivery",
      "logs>DeleteLogDelivery",
      "logs:ListLogDeliveries",
      "logs:PutResourcePolicy",
      "logs:DescribeResourcePolicies",
      "logs:DescribeLogGroups"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "arn:aws:iam::*:role/aws-service-role/
kafkaconnect.amazonaws.com/AWSServiceRoleForKafkaConnect*",
    "Condition": {
      "StringLike": {
        "iam:AWSServiceName": "kafkaconnect.amazonaws.com"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:AttachRolePolicy",
      "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/
kafkaconnect.amazonaws.com/AWSServiceRoleForKafkaConnect*"
  },
  {
    "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"
      }
    }
  }
}

```

```
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:PutBucketPolicy",
      "s3:GetBucketPolicy"
    ],
    "Resource": "ARN of the Amazon S3 bucket to which you want MSK Connect to deliver logs"
  },
  {
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "ARN of the service execution role"
  },
  {
    "Effect": "Allow",
    "Action": "s3:GetObject",
    "Resource": "ARN of the Amazon S3 object that corresponds to the custom plugin that you want to use for creating connectors"
  },
  {
    "Effect": "Allow",
    "Action": "firehose:TagDeliveryStream",
    "Resource": "ARN of the Firehose delivery stream to which you want MSK Connect to deliver logs"
  }
]
}
```

サービス間の混乱した代理の防止

混乱した代理問題は、アクションを実行するためのアクセス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の実行を強制できてしまう場合に生じる、セキュリティ上の問題です。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。これを防ぐために、AWSには、アカウント内

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が用意されています。

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。`Account-ID` と `MSK-Connector-ARN` は、ユーザー自身の情報に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": " kafkaconnect.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "Account-ID"
        },
        "ArnLike": {
```

```
        "aws:SourceArn": "MSK-Connector-ARN"
    }
}
]
```

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、多くの一般的なユースケースに対する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、ロールへの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りますことに注意してください。ユースケース別に[カスタマー マネージドポリシー](#)を定義して、マネージドポリシー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) が更新されます。は、新しい AWS のサービスが起動されたとき、または既存のサービスで新しい API AWS オペレーションが使用可能になったときに、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: 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。

```
{
  "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イスにより、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。

IAM エンティティ `KafkaConnectServiceRolePolicy` に を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。

```
{
  "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 での AWS マネージドポリシーの更新

MSK Connect の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。

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 が AWS マネージド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 で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、ユーザーに代わってサービスから他の AWS の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 のみがそのロ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 AWS サービス](#)」を参照して、サービスにリンクされたロール 列が [はい] になっているサービスを見つけてください。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、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 `AWSServiceRoleForKafkaConnect`。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。サービスリンク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参照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を設定します。
- 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 ゲートウェイのセットアップ

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。

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が、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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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。

プロパティ	値
内部トラフィックのルート	<ul style="list-style-type: none">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

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。

トピック

- [コネクタ用の VPC DHCP オプションセットの設定](#)
- [VPC の DNS 属性](#)
- [障害処理](#)

コネクタ用の VPC DHCP オプションセットの設定

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に必要な場合は、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

「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。

障害処理

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
<p>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。</p>	<p>コネクタにこれらの CloudWatch ログを設定している場合、DNS 解決クエリが失敗したためにコネクタの作成に失敗することがあります。</p> <p>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。</p>
<p>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。</p>	<p>コネクタにこれらの CloudWatch ログを設定している場合、DNS 解決クエリが失敗したためにコネクタの作成に失敗することがあります。</p> <p>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が、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。</p>

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。ロググループの作成方法については、[「ログユーザーガイド」の「ロググループ CloudWatch 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、ログに送信される CloudWatch 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。

MSK Connect のモニタリング

モニタリングは、MSK Connect およびその他の 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 は、AWS リソースと、で実行しているアプリケーションを AWS リアルタイムで CloudWatch 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、指定されたメトリック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、でコネクタの CPU 使用率やその他のメトリクス CloudWatch を追跡し、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 ユーザーガイド CloudWatch」](#) を参照してください。

次の表は、MSK Connect が ConnectorName デイメンション CloudWatch で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配信します。はこれらのメトリクスを 15 か月間 CloudWatch 保持するため、履歴情報にアクセスしてコネクタの動作をよりの確に把握できます。また、特定のしきい値をモニタリングするアラームを設定し、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できます。詳細については、[「Amazon ユーザーガイド CloudWatch」](#) を参照してください。

MSK Connect メトリクス

メトリクス名	説明
BytesInPerSec	コネクタが受信した合計バイト数。
BytesOutPerSec	コネクタによって配信された合計バイト数。

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、最小ヒープサイズで始まり、約 80~90% の安定した最大値まで段階的に増加します。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。
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。
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。

メトリクス名	説明
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。
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。

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](#)
- [Debezium ソースコネクタ \(設定プロバイダー付き\)](#)

Amazon S3 シンクコネクタ

この例では、Confluent [Amazon S3 シンクコネクタ](#)とを使用して MSK Connect で Amazon S3 シンクコネクタ AWS CLI 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.partitionner.DefaultPartitioner",
    "storage.class": "io.confluent.connect.s3.storage.S3Storage",
    "s3.bucket.name": "my-test-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"  
}
```

Debezium ソースコネクタ (設定プロバイダー付き)

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、「[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](#)」を参照してください。

開始する前に

コネクタは、の AWS Secrets Manager 外部にある 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、以下のものがが必要です。

- 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が、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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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

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` です。 `#example-custom-plugin-name#` をプラグインに含める名前に置き換え、 `#arn-of-your-s3-bucket>` を ZIP ファイルをアップロードした S3 バケットの ARN `<file-key-of-ZIP-object>` に置き換え、 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に置き換えます。

```
{
  "name": "<example-custom-plugin-name>",
  "contentType": "ZIP",
  "location": {
    "s3Location": {
      "bucketArn": "<arn-of-your-s3-bucket>",
      "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 して作成する
 - 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 [サービス実行のロール](#) 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 に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetResourcePolicy",
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret",
        "secretsmanager:ListSecretVersionIds"
      ],
      "Resource": [
        "<arn:aws:secretsmanager:us-east-1:123456789000: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。Secrets Manager Config プロバイダーの設定プロパティ AWS の詳細については、プラグインのドキュメント [SecretsManagerConfigProvider](#) の「」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。<code>MySecret-1234</code> 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、<code><arn-of-config-provider-worker-configuration></code> 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。<code><placeholder></code> 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"
}
```

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。

ベストプラクティス

この資料は、Amazon MSK Connect を使用してパフォーマンスを最大にするための推奨事項をすばやく検索できるリファレンスとしてご使用ください。

トピック

- [コネクタからの接続](#)

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"

Amazon MSK Connect 移行ガイド

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。

トピック

- [Amazon MSK Connect を使用する利点](#)
- [Amazon MSK Connect への移行](#)

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 で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、デバッグ、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に合わせて Kafka Connect の機能をカスタマイズおよび拡張できます。

Apache Kafka Connect クラスターを独自に運用している場合、またはオープンソースの Apache Kafka Connect アプリケーションを に移行しようとする、問題が発生する可能性があります AWS。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 Apache Kafka Connect クラスターのセットアップ時のエンジニアリングの障害、管理運用オーバーヘッドなどがあります。

これらの課題に対処するには、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 に移行することをお勧めします AWS。Amazon MSK Connect は、Kafka Connect を使用して、Apache Kafka クラスターとデータベース、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との間でデータをストリーミングすることを簡素化します。

Amazon MSK Connect への移行には次のような利点があります。

- 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、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、ワークロードを中断することなく、Connect クラスターの状態を継続的にモニタリングし、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、障害が発生したタスクを自動的に復旧して、本番環境の中断を軽減できます。タスクの失敗は、Kafka の TCP 接続制限の超過や、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するときのタスクの再調整などの一時的なエラーが原因で発生する可能性があります。
- 自動水平スケーリングと垂直スケーリング — Amazon MSK Connect を使用すると、コネクタアプリケーションを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ます。Amazon MSK Connect がスケーリングを管理します。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を指定するだけで済みます。Amazon MSK Connect UpdateConnector API オペレーションを使用すると、可変スループットをサポートするために、1~8 個の vCPUs 間で vCPUs 垂直方向にスケールアップまたはスケールダウンできます。
- プライベートネットワーク接続 — Amazon MSK Connect は、 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 AWS PrivateLink プライベートに接続します。

Amazon MSK Connect への移行

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トピックを簡単に説明します。このセクションでは、ソースコネクタとシンクコネクタを移行する手順についても説明します。

トピック

- [Kafka Connect で使用される内部トピック](#)
- [Amazon MSK Connect アプリケーションのステート管理](#)
- [ソースコネクタの Amazon MSK Connect への移行](#)
- [シンクコネクタを Amazon MSK Connect に移行する](#)

Kafka Connect で使用される内部トピック

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シップの内部トピックを使用して状態を保存します。以下は、Kafka Connect アプリケーションに使用される内部トピックに対応する設定値です。

- で指定された設定トピック `config.storage.topic`

設定トピックでは、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をリクエストするたびに (例えば、コネクタがより多くのタスクを開始できることを検出すると)、レコードがこのトピックに出力されます。このトピックでは圧縮が有効になっているため、エンティティごとに常に最後の状態が維持されます。

- で指定されたオフセットトピック `offset.storage.topic`

オフセットトピックでは、Kafka Connect はソースコネクタのオフセットを保存します。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ります。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書き込むために使用されます。Kafka からデータを読み取って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。

- で指定されたステータストピック `status.storage.topic`

ステータストピックでは、Kafka Connect はコネクタとタスクの現在の状態を保存します。このトピックは、REST API のユーザーがクエリするデータの中心的な場所として使用されます。このトピックでは、ユーザーは任意のワーカーにクエリを実行しても、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ります。

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シップ API を幅広く活用しています。グループの名前はコネクタ名にちなんで付けられます。例えば、file-sink という名前

のコネクタの場合、グループの名前は `connect-file-sink`。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、`connect-file-sink` などの通常のコンシューマーグループツールを使用して取得できます `Kafka-consumer-group.sh`。シンクコネクタごとに、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ステート管理

デフォルトでは、Amazon MSK Connect は、Amazon MSK Connector ごとに Kafka クラスターに 3 つの個別のトピックを作成し、コネクタの設定、オフセット、ステータスを保存します。デフォルトのトピック名は、次のように構成されます。

- `__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継続性を提供するには、デフォルトの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。

ソースコネクタの Amazon MSK Connect への移行

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、`connect-file-sink` 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 クラスターで実行されている 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` に設定された値を持つという名前のトピックにオフセットを保存します `offset.storage.topic`。以下は、`movies` および `shows` という名前の 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 コネクタのサンプルオフセットメッセージです `shows`。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 です18343。shows テーブルからインポートされた最新の行のプライマリ ID は です732。

```
["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 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ているものと同じ値に設定します。
3. 既存の Kafka Connect クラスターで `PUT /connectors/connector-name/pause` リクエストを実行して、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タスクがすべて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 クラスターで `GET /connectors/connector-name/status` リクエストを行うか、プロパティ に設定されているトピック名からメッセージを消費します `status.storage.topic`。
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、既存のクラスターで `GET /connectors/connector-name/config` リクエストを行うか、プロパティ に設定されているトピック名からメッセージを消費します `config.storage.topic`。
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 Connector のステータスが の場合 `active`、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` リクエストを実行して、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、 で実行されているオンプレミスまたはセルフマネージ

ト Kafka Connect クラスターで実行されている 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 シンクコネクタは Kafka グループメンバーシップ 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、セルフマネージドクラスターから Amazon MSK Connect へのシンクコネクタ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 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ているものと同じ値に設定します。
3. 既存の Kafka Connect クラスターで `PUT /connectors/connector-name/pause` リクエストを実行して、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タスクがすべて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 クラスターで `GET /connectors/connector-name/status` リクエストを行うか、プロパティ に設定されているトピック名からメッセージを消費します `status.storage.topic`。
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、既存のクラスターで `GET /connectors/connector-name/config` リクエストを行うか、プロパティ に設定されているトピック名からメッセージを消費します `config.storage.topic`。
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 Connector のステータスが の場合 `active`、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` リクエストを実行して、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。

MSK レプリケーター

Amazon MSK Replicator とは

Amazon MSK レプリケーターは、異なるリージョンまたは同じ AWS リージョン (複数可) の 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。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、北カリフォルニア)、カナダ (ca-central-1、中央)。

ここでは、Amazon MSK Replicator の一般的な使用法をいくつか示します。

- 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。
- ローカルで書き込み、データを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、ある 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、同じリージョンまたは異なる AWS リージョンに配置できます。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 レプリケーターは、ソースクラスタートピックパーティションの最新のオフセットからターゲットクラスターにすべてのデータを非同期的にコピーします。「新しいトピックの

検出とコピー」設定が有効になっている場合、MSK レプリケーター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で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。または、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、[作成時にレプリケーターを設定](#)して、ソースクラスタートピックパーティションの最も早いオフセットからレプリケーションを開始することもできます。

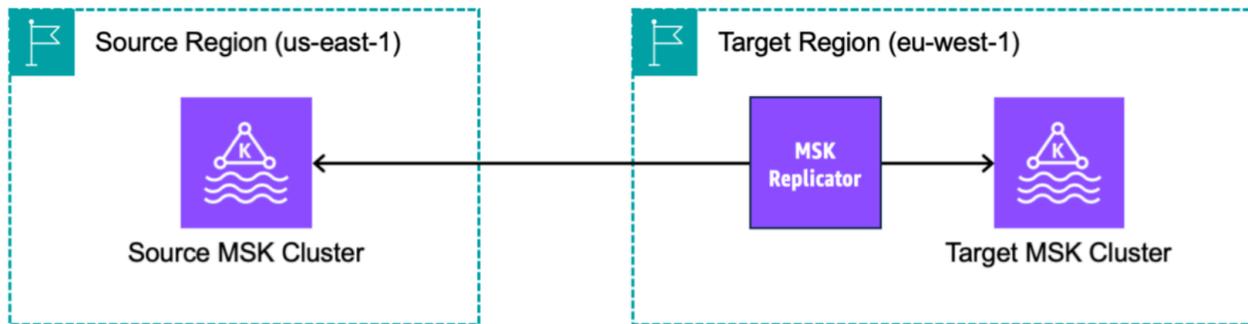
MSK レプリケーターはデータを保存しません。データはソースクラスターから消費され、メモリ内にバッファされ、ターゲットクラスターに書き込まれます。バッファは、データが正常に書き込まれるか、再試行後に失敗すると自動的にクリアされます。MSK レプリケーターとクラスター間のすべての通信とデータは、転送中に常に暗号化されます。DescribeClusterV2、などのすべての MSK レプリケーター API コールDescribeTopicDynamicConfigurationはCreateTopic、にキャプチャされます AWS CloudTrail。MSK ブローカーログにも同じ内容が反映されます。

MSK レプリケーターは、レプリケーター係数が 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で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 レプリケーターは、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、読み取りアクセスコントロールリスト (ACLs 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 レプリケーター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。

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 レプリケーターは、ストリームの先端 (トピックパーティションの終わり) に近い位置から読み取りを行うソースクラスター上のコンシューマーに合わせて最適化します。コンシューマーグループがソースクラスターで遅延している場合、ターゲット上のコンシューマーグループの遅延がソースよりも大きくなる場合があります。つまり、ターゲットクラスターへのフェイルオーバー後、コンシューマーはより重複したメッセージを再処理します。この遅延を減らすには、ソースクラスターのコンシューマーが追いついて、ストリームのティップ (トピックパーティションの終わり) から消費を開始する必要があります。コンシューマーが追いつくと、MSK レプリケーターは自動的に遅延を減らします。



Amazon MSK Replicator を作成するための要件と考慮事項

Amazon MSK Replicator を実行するための以下の MSK クラスター要件に注意してください。

トピック

- [MSK レプリケーターの作成に必要なアクセス許可](#)
- [サポートされるクラスタータイプとバージョン](#)
- [MSK サーバーレスクラスター設定](#)
- [クラスター設定の変更](#)

MSK レプリケーターの作成に必要なアクセス許可

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。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole",
        "iam:CreateServiceLinkedRole",

```

```

        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:CreateNetworkInterface",
        "ec2:DescribeVpcs",
        "kafka:CreateReplicator",
        "kafka:TagResource"
    ],
    "Resource": "*"
}
]
}

```

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。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
        "kafka:DescribeReplicator",
        "kafka:ListTagsForResource"
      ],
      "Resource": "*"
    }
  ]
}

```

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](#) 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リージョンにあるか異なる AWS リージョンにあるかに関係なく、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。
- 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ます。t3.small クラスターはサポートされません。
- 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方式でマルチ 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、ソースクラスターでのその他の認証方式には必要ありません。同じ AWS リージョン内のクラスター間でデータをレプリケートする場合、マルチ VPC は必要ありません。[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。

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.ms。
- 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。

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。

Amazon MSK Replicator の使用を開始する

このチュートリアルでは、同じ AWS リージョンまたは異なる AWS リージョンでソースクラスターとターゲット クローラを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。

ステップ 1: 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

ソース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。

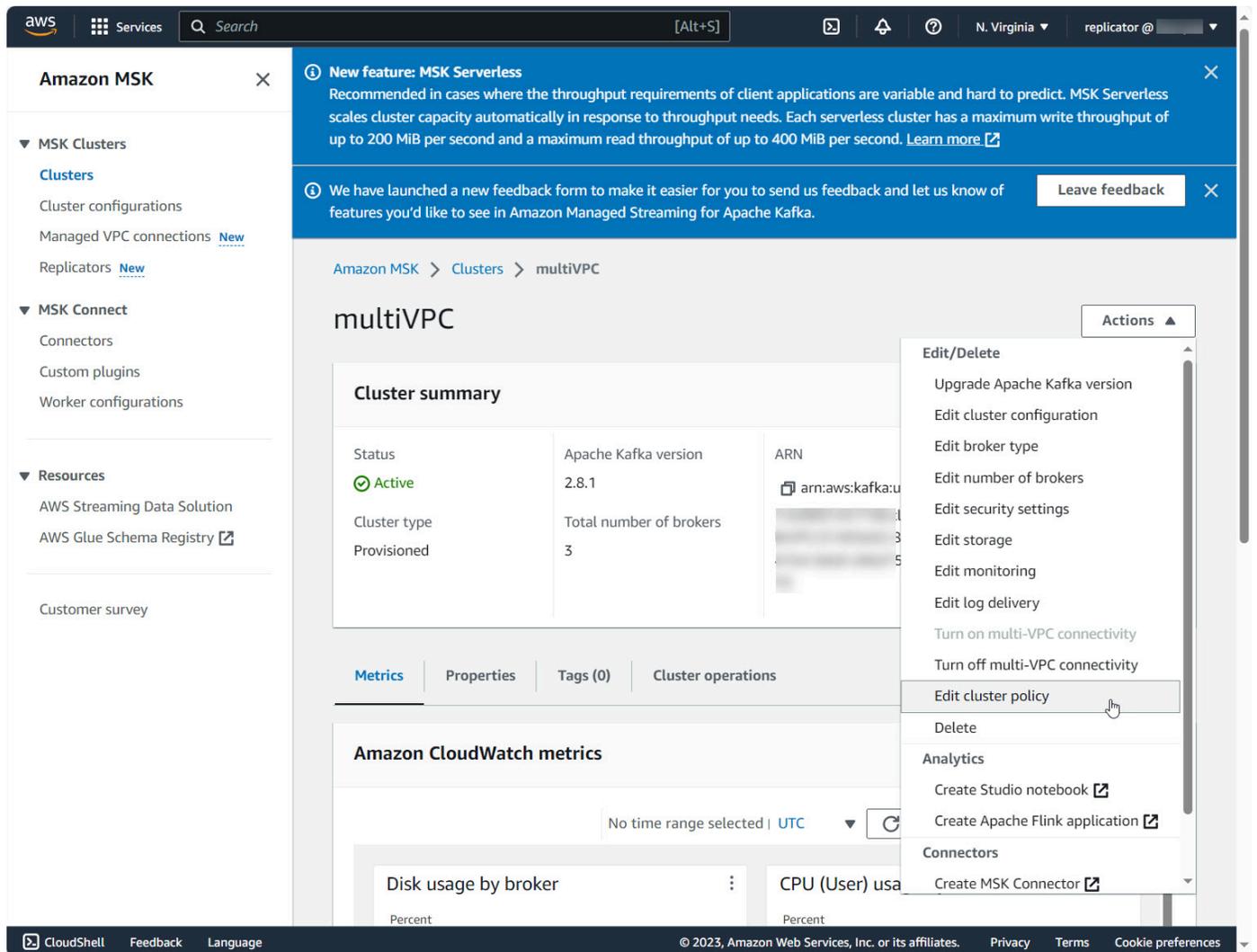
3. クロスリージョン MSK レプリケーターの場合、ソースクラスターにリソースベースのアクセス許可ポリシーをアタッチする必要があります。これにより、MSK がこのクラスターに接続してデータをレプリケートできます。これを行うには、以下の CLI または AWS コンソールの手順を使用します。「[Amazon MSK のリソースベースのポリシー](#)」も参照してください。同一リージョン MSK レプリケーターの場合は、このステップを実行する必要はありません。

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。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "kafka.amazonaws.com"
        ]
      },
      "Action": [
        "kafka:CreateVpcConnection",
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeClusterV2"
      ],
      "Resource": "<sourceClusterARN>"
    }
  ]
}
```

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すると、は必要な信頼ポリシーをロールにア AWS 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>"
    }
  ]
}
```

ステップ 2: 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配置できます。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。

ステップ 3: Amazon MSK Replicator を作成する

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

- [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ーの作成](#)
- [ソースクラスターの選択](#)
- [ターゲットクラスターの選択](#)
- [レプリケーターの設定とアクセス許可を設定します。](#)

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ーの作成

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. [レプリケーターの作成] を選択します。
4. [レプリケーターの詳細] 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、ソースクラスターにセキュリティグループを提供する必要はありません。
 - 同じリージョンレプリケーション (SRR) の場合は、<https://console.aws.amazon.com/ec2/> の Amazon EC2 コンソールに移動し、レプリケーターに提供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。また、ソースクラスターのセキュリティグループに、ソースに提供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、 と入力します9098。MSK レプリケーターは IAM アクセスコントロールを使用して、ポート 9098 を使用するクラスターに接続します。
7. ソース 列で、ソースクラスター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の名前を入力し (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である可能性があります)、ルールの保存 を選択します。

ソースに提供され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、と入力します9098。MSK レプリケーターは IAM アクセスコントロールを使用して、ポート 9098 を使用するクラスターに接続します。
6. ソース 列で、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

または、セキュリティグループを使用してトラフィックを制限しない場合は、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を選択

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。DescribeReplicator API または MSK コンソールのレプリケーターの詳細ページを使用して、ターゲットクラスターのsourceKafkaClusterエイリアスフィールドでトピック名に追加されるプレフィックスを見つけることができます。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配置されている 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た AWS リソースへのトラフィックの制御](#)」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) と同じリージョンレプリケーション (SRR) の両方について、<https://console.aws.amazon.com/ec2/> の Amazon EC2 コンソールに移動し、レプリケーターに提供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、適用された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、 と入力します9098。MSK レプリケーターは IAM アクセスコントロールを使用して、ポート 9098 を使用するクラスターに接続します。
7. ソース 列で、ターゲットクラスター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の名前を入力し (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である可能性があります)、ルールの保存 を選択します。

ターゲットに提供される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、適用された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、 と入力します9098。MSK レプリケーターは IAM アクセスコントロールを使用して、ポート 9098 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

または、セキュリティグループを使用してトラフィックを制限しない場合は、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。

レプリケーターの設定とアクセス許可を設定します。

1. [レプリケーター設定] 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 レプリケーターは、ソートされた順序で最大 750 個のトピックのみをレプリケートします。さらに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、AWS コンソールの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。 <https://console.aws.amazon.com/support/home#/TopicCount> 「」

メトリクスを使用して、レプリケートされるトピックの数をモニタリングできません。[Amazon MSK クォータ](#) を参照してください。

- デフォルトでは、MSK レプリケーターは、選択したトピックの最新 (最新の) オフセットからレプリケーションを開始します。または、トピックに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早い (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 APIs [startingPosition](#) フィールドに対応します。

Note

MSK レプリケーターは、ソースクラスターの新しいコンシューマーとして機能します。レプリケートするデータの量とソースクラスターの消費容量によっては、ソースクラスター上の他のコンシューマーがスロットリングされる可能性があります。レプリケーターを最も早い開始位置に設定した場合、MSK レプリケーターは最初にデータのバーストを読み取り、ソースクラスターのすべての消費容量を消費する可能性があります。レプリケーターが追いついた後、ソースクラスタートピックのスループットに合わせて消費レートが下がるはずですが、最も早い位置からレプリケートする場合は、[Kafka クォータを使用してレプリケーターのスループットを管理し](#)、他のコンシューマーがスロットリングされないようにすることをお勧めします。

- 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の実行](#)」を参照してください。

4. [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。
5. [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。
6. [アクセス許可] ペインで、次のいずれかを実行します。
 - a. 必要なポリシーを使用して 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。

Access permissions

Replicator uses IAM access control to connect to source and target MSK clusters. Your source and target clusters should be turned on for IAM access control with permissions for the IAM role. See [permissions required to successfully create a replicator](#).

 You can't change the access permissions after you create the replicator.

Access to cluster resources

- Create or update IAM role **MSKReplicatorServiceRole-** with required policies
- Choose from IAM roles that Amazon MSK can assume

- b. Amazon MSK が引き受けることができる IAM ロールから選択して、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。
 - レプリケーターがソースおよびターゲット MSK クラスターに対する読み取りと書き込みに使用する IAM ロールを作成します。このロールには、信頼ポリシーの一部として次の JSON が含まれ、ロールに AWSMSKReplicatorExecutionRole がアタッチされます。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。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kafka.amazonaws.com"
      }
    }
  ]
}
```

```
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "<yourAccountID>"
      }
    }
  }
]
```

7. [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 クラスターのタグ付け](#)」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。
8. [作成] を選択します。

アクセスkafka-cluster:WriteData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。
<https://docs.aws.amazon.com/msk/latest/developerguide/iam-access-control.html#how-to-use-iam-access-control>ソースクラスターとターゲットクラスターの両方にアクセスkafka-cluster:WriteDataIdempotently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 レプリケーター設定の編集

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、またはレプリケーターの開始位置を変更することはできません。ただし、トピックやコンシューマーグループなど、他のレプリケーター設定を編集してレプリケートすることはできます。

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. 左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. [レプリケーター設定] 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 レプリケーター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。
6. 変更を保存します。

 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の実行](#)」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。

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 AWS があるリージョンで、 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ReplicatorNameの 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 メトリクスは、トピックレベルの認証エラーには適用されません。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をモニタリングします MessagesInPerSec。トピックが 0 に ReplicationLatency ドロップされたが、トピックにまだデータが生成されている場合は、レプリケーターにトピックの認証の問題があることを示します。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が増加する場合は、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ます。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	ReplicationName	ミリ秒	パーティション	最大値
			ReplicationName、トピック	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	
		ンシーが高くな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。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が消費したメッセージの間の遅延 MessageLag 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/中断されていても、レプリケーターはターゲットクラスターに消費したメッセージの書き込みを完了します。停止後、はレプリケーターがソースクラス	Replicat rName	カウン ト	パー ティ ション	合計
			Replicat rName、 トピッ ク	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
		ターの背後にあるメッセージの数を示す増加 MessageLag を示し、メッセージ数が 0 になるまでモニタリングでき、レプリケーターがソースクラスターに追いついたことを示します。				

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。ガトピックに対して ReplicatorThroughput を削除した場合は、KafkaClusterPingSuccessCount および 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、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、トピック	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。	ReplicaName, 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	<p>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。</p>	ReplicatorName, ClusterA ias	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特定できます。	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/パッシブのクラスタートポロジをセットアップし、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 レプリケーターはデータをレプリケート at-least-once し、スタンバイクラスタ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/パッシブのクラスタートポロジの使用

アクティブ/アクティブのクラスター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に障害が発生しても、もう一方のリージョンのクラスタ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。

アクティブ/パッシブ Kafka クラスター設定の作成とレプリケートされたトピックの命名規則

アクティブ/パッシブ設定では、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。

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。例えば、MSK レプリケーターは、プライマリクラスターからスタンバイクラスター内の「<sourceKafkaClusterAlias>.topic1」という新しいトピックに「topic1」をレプリケートします。例えば、「topic1」に書き込むようにプロデューサーを設定し、両方のリージョンで「*.topic1」を使用して消費するようにコンシューマーを設定することができます。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。

セカンダリ AWS リージョンにフェイルオーバーするタイミング

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 CloudWatch。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。イベントが発生した場合は、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の実行

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障害耐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。

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 (`.*topic` 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。

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。

1. 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。
2. 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の実行

ソース MSK クラスターがあるプライマリ AWS リージョンにサービスイベントがあり、ターゲット MSK クラスターがあるセカンダリ AWS 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、計画外のフェイルオーバーを実行できます。計画外のフェイルオーバーでは、一部のデータが失われる可能性があります。

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。これは失敗する可能性があります。

2. セカンダリリージョンのターゲット 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。
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て、レプリケーターの開始位置を最も早い 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最も早い 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。MSK レプリケーターは、フェイルバック中にデータをプライマリリージョンにレプリケー

トするときに、ソースクラスターエイリアスをプレフィックスとして持つトピックを自動的にスキップします。

[計画外のフェイルオーバーステップ](#) に従った場合は、プライマリリージョンからセカンダリリージョンへのフェイルオーバーの最後のステップの一部として、フェイルバックレプリケーターがすでに作成されているはずで

計画外のフェイルオーバー手順に従わなかった場合は、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て、レプリケーターの開始位置を最も早い 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最新の のデフォルト値から最も古い に変更しない場合、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。

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完了し、 の MessageLag メトリクス CloudWatch が 0 に近い場合にのみ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. 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 状態であり、ReplicatorThroughput および レイテンシーメトリクスを使用して期待どおりに動作することを確認します。

MSK レプリケーターを使用したアクティブ/アクティブ設定の作成

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. 2 つのコンシューマーセットを作成し、それぞれが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。

MSK レプリケーターのトラブルシューティング

トピック

- [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](#)
- [ターゲットクラスターのメッセージオフセットがソースクラスターと異なる](#)
- [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同期していないが、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ません](#)
- [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](#)

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。[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

ターゲットポリシーが設定されていることも確認します。[ステップ 1: Amazon MSK ソースクラスターを準備する](#) を参照してください。

- 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。「[レプリケーターの設定とアクセス許可を設定します。](#)」を参照
- 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との接続をブロックしていないことを確認します。
- 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。

- MSK レプリケーターが提供する AuthError メトリクスを使用して、レプリケーターが認証エラーに陥っていないことを確認します CloudWatch。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 デイ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。
- 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. の `KafkaClusterPingSuccessCount` メトリクスを使用して、MSK レプリケーターからソースクラスターとターゲットクラスターにアクセスできることを確認します CloudWatch。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. 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がないためにレプリケーターが失敗していないことを確認します CloudWatch。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で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 秒かかる場合があることに注意してください。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、レプリケーターの開始位置が最新 (デフォルト) の場合、レプリケートされません。または、ターゲットクラスターのトピックに既存のメッセージをレプリケートする場合は、ソースクラスタートピックパーティションの最も早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。

ターゲットクラスターのメッセージオフセットがソースクラスターと異なる

データのレプリケートの一環として、MSK レプリケーター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メッセージを生成します。これにより、送信元クラスターとターゲットクラスターでメッセージに異なるオフセットが発生する可能性があります。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同期を有効にした場合、MSK レプリケーター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し、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から処理を再開できます。

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同期していないか、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ません

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、次の手順に従います。

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で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してい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を確認します。
3. MSK レプリケーターがターゲットクラスターにトピックを作成済みであることを確認します。レプリケーターがターゲットクラスターで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、レプリケーターの開始位置が最新 (デフォルト) の場合、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 レプリケーターによってレプリケートされていないメッセージのみを使用した場合、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。ターゲットクラスターでトピックが正常に作成されると、MSK レプリケーター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取りを開始すると、MSK レプリケーター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。または、ターゲットクラスターのトピックに既存のメッセージをレプリケートする場合は、ソースクラスタートピックパーティションの最も早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 レプリケーターは、トピックパーティションの末尾に近い位置から読み取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。コンシューマーグループがソースクラスターで遅延している場合、ソースと比較してターゲット上のコンシューマーグループの遅延が大きくなる場合があります。つまり、ターゲットクラスターへのフェイルオーバー後、コンシューマーはより重複したメッセージを再処理します。この遅延を減らすには、ソースクラスターのコンシューマーが追いついて、ストリームのティップ (トピックパーティションの終わり) から消費を開始する必要があります。コンシューマーが追いつくと、MSK レプリケーターは自動的に遅延を減らします。

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、クラスターが互いに地理的にどの程度離れているか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 レプリケーターが提供するメトリクスを使用して ThrottleTime、ソース/ターゲットクラスターの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をミリ秒単位で CloudWatch 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. ReplicationLatency および は、AWS リージョンのパフォーマンスが低下すると増加する MessageLag 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。

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、MSK レプリケーター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 クラスター用のブートストラップブローカーの取得](#) 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 を取得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「[MSK サーバーレスクラスター設定](#)」を参照してください。

クラスターの状態

次の表に、クラスターの各状態と、それらの意味を示します。また、クラスターがこれらの状態のいずれかの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。

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 はクラスターをセットアップしています。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するには、クラスターが ACTIVE 状態になるまで待つ必要があります。
DELETING	クラスターを削除しています。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	クラスターの作成または削除プロセスが失敗しました。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。クラスターは削除できますが、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。

クラスタースタtus	意味と実行可能なアクション
	<p>例えば、ブローカーが応答しない場合があります。引き続き、クラスタースタtus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、クラスタースタtusが ACTIVE 状態に戻るまでは、Amazon MSK API を実行したり、クラスタースタtus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。</p>
MAINTENANCE	<p>Amazon MSK は、クラスタースタtus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、クラスタースタtus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、クラスタースタtusが ACTIVE 状態に戻るまでは、Amazon MSK API を実行したり、クラスタースタtus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。</p>
REBOOTING_BROKER	<p>Amazon MSK はブローカーを再起動しています。引き続き、クラスタースタtus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、クラスタースタtusが ACTIVE 状態に戻るまでは、Amazon MSK API を実行したり、クラスタースタtus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。</p>
UPDATING	<p>ユーザーが開始した Amazon MSK API または AWS CLI オペレーションがクラスタースタtusを更新しています。引き続き、クラスタースタtus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、クラスタースタtusが ACTIVE 状態に戻るまで、クラスタースタtusに対して追加の 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。</p>

Amazon Managed Streaming for Apache Kafka 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からメリットを得られます。

セキュリティは、AWS とユーザーの間で共有される責任です。[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責任 AWS を担います。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。コンプライアンス [AWS プログラム](#) コンプライアンスプログラム の一環として、サードパーティーの監査者は定期的にセキュリティの有効性をテストおよび検証。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](#)
- [Apache へのアクセスの制御 ZooKeeper](#)
- [ログ記録](#)
- [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。お客様は、このインフラストラクチャでホストされてい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のよくある質問](#)」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 は必須であり TLS 1.3 がお勧めです。
- を使用して 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。
- 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 などの高度なマネージド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 140-2 検証済みの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 \(FIPS\) 140-2](#)」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は、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配置し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 名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や

診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

トピック

- [Amazon MSK 暗号化](#)
- [暗号化を開始する方法](#)

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。証明書の更新オペレーションを開始すると、クラスターの状態が MAINTENANCE に設定されます。更新が完了すると、この状態は再び ACTIVE に設定されます。クラスターが MAINTENANCE 状態の間は、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は実行できません。

保管中の暗号化

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。

暗号化を開始する方法

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。次に例を示します。

```
{
  "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。を指定しない場合でも EncryptionAtRest、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。

クラスターの作成時に暗号化設定を指定するには

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。次に例を示します。 `{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、「[To find the version of your MSK cluster](#)」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"
}
```

TLS 暗号化をテストするには

1. [the section called “ステップ 3: 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、を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 *BootstrapBrokerStringTls* に置き換えます。

```
<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。

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 AWS のサービス 使用できる です。

このページでは、IAM を使用して、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説明します。クラスターで Apache Kafka 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 “Apache Kafka API の認証と認可”](#)」を参照してください。

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の使用](#)
- [AWS Amazon MSK の マネージドポリシー](#)
- [Amazon MSK の ID と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を把握するには、「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](#)
- [AWS マネージドポリシー](#)
- [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 ポリシー要素のリファレンス](#)」を参照してください。

アクション

管理者は AWS JSON 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んなリソースにどんな条件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。

リソース

管理者は AWS JSON 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どのような条件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。

条件キー

管理者は AWS JSON 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んなリソースにどんな条件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することで、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 要素を指定するか、1つの Condition 要素に複数のキーを指定すると、AWS は AND 論理演算子を使用してそれら进行评估します。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 を评估します。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 [AWS](#) 」の「[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。

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。

AWS マネージドポリシー

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管理するには、`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の詳細については、「[the section called “クラスターのタグ付け”](#)」を参照してください。

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、「[タグに基づく Amazon MSK クラスターへのアクセス](#)」を参照してください。

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 ロールを引き受ける、またはクロスアカウントロール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 [GetFederationトークン](#) 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。

サービスリンクロール

[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 “サービスリンクロール”](#)」。

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

- [ポリシーの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](#)

ポリシーの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、または削除できるか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する – ユーザーとワークロードにアクセス許可を付与するには、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 力

スタマー管理ポリシーを定義して、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[AWS ジョブ機能の管理ポリシー](#)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、などの特定の を介してサービスアクションが使用される場合に 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 policy elements: Condition](#) (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 ポリシーの検証](#)を参照してください。
- 多要素認証 (MFA) を要求する – 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、セキュリティを強化するために MFA を有効にします。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 保護 API アクセスの設定](#)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。

```
{
  "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。

```
{
  "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の使用

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、ユーザーに代わってサービスから他の AWS の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の作成](#)
- [Amazon MSK のサービスリンクロールの編集](#)
- [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になります。
- `*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、「」を参照してください [KafkaServiceRolePolicy](#)。

`AWSServiceRoleForKafka` サービスにリンクされたロールは、ロールの引き受けについて以下のサービスを信頼します。

- `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。

```
{
  "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権限](#)」を参照してください。

Amazon MSK のサービスリンクロールの作成

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の編集

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 を使用したロールの説明の編集はできます。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。

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、AWS 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 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。ユースケース別に [カスタマー マネージドポリシー](#) を定義して、マネージドポリシー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) が更新されます。は、新しい AWS のサービスが起動されたとき、または既存のサービスで新しい API AWS オペレーションが使用可能になったときに、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 管理ポリシー](#)」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。
- **アクセス Amazon EC2 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。
- **アクセス AWS KMS 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。
- **アクセス CloudWatch Logs, Amazon S3, and Amazon Data Firehose 許可** – Amazon MSK がログ配信先に到達可能であり、ブローカーログの使用に有効であることを確認するために必要です。
- **アクセス IAM 許可** — Amazon MSK がアカウントでサービスにリンクされたロールを作成でき、Amazon MSK にサービス実行ロールを渡すことができるようにするには、**が**必要です。

```
{
  "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: AmazonMSKReadOnly アクセス

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。

- **アクセスAmazon MSK許可** — Amazon MSK リソースを一覧表示し、説明し、それらの情報を取得できます。
- **アクセスAmazon EC2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 ENIs を記述するために使用されます。

- **アクセスAWS KMS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。

```
{
  "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

IAM エンティティ KafkaServiceRolePolicy に を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 “サービスリンクロール”](#)」を参照してください。

AWS 管理ポリシー: 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 レプリケーターに付与します。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 レプリケーターに付与します。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 レプリケーターに付与します。

- **consumer group** — Amazon MSK レプリケーターに、コンシューマーグループの説明と変更、MSK クラスターからの読み取りと書き込みの日付、レプリケーターによって作成された内部トピックの削除を行うアクセス許可を付与します。

```
{
  "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

Amazon MSK の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。

変更	説明	日付
アクセスWriteData Idempotently 許可が追加 AWSMSKReplicatorExecutionRole – 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 AWSMSKReplicatorExecutionRole アクセス WriteDataIdempotently 許可をポリシーに追加しました。	2024 年 3 月 12 日
AWSMSKReplicatorExecutionRole - 新しいポリシー	Amazon MSK は、Amazon MSK レプリケーターをサポートする AWSMSKReplicatorExecutionRole ポリシーを追加しました。	2023 年 12 月 4 日

変更	説明	日付
AmazonMSKFullAccess – 既存のポリシーの更新	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 – 既存ポリシーへの更新	Amazon MSK に、マルチ VPC プライベート接続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 3 月 8 日
AmazonMSKFullAccess – 既存のポリシーの更新	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 – 既存のポリシーの更新	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 が AWS マネージド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 と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

から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](#)
- [相互 TLS 認証](#)
- [AWS Secrets Manager での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、Python、Go、およ

び .NET で記述された Kafka クライアントを含む JavaScriptJava および Java 以外のクライアントで機能します。

Amazon MSK はアクセスイベントをログに記録するため、それらを監査できます。詳細については、「[the section called “CloudTrail イベント”](#)」を参照してください。

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。

Important

IAM アクセスコントロールは 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、「[the section called “Apache へのアクセスの制御 ZooKeeper”](#)」を参照してください。

Important

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。

Important

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 ロールの承認には影響しません。IAM ロール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、このセクションの残りの部分で詳しく説明します。

- [the section called “IAM アクセス制御を使用するクラスターを作成する”](#)
- [the section called “IAM アクセス制御用にクライアントを設定する”](#)
- [the section called “認可ポリシーを作成する”](#)

- [the section called “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クラスターを作成する

このセクションでは、API AWS Management Console、または 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 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、「[the section called “セキュリティの更新”](#)」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 アクセスコントロール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. 以下の例の Python Kafka クライアントで強調表示されている構文をガイドとして使用し、`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。

```
#!/usr/bin/python3from kafka import KafkaProducer
from kafka.errors import KafkaError
import socket
import time
from aws_msk_iam_sasl_signer import MSKAuthTokenProvider

class MSKTokenProvider():
    def token(self):
        token, _ = MSKAuthTokenProvider.generate_auth_token('<my aws region>')
        return token

tp = MSKTokenProvider()

producer = KafkaProducer(
    bootstrap_servers='<my bootstrap string>',
    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にある「はじめに」セクションの指示に従ってください。

- 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。

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。 [the section called “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、 という名前のクラスターの承認ポリシーの例です MyTestCluster。 Action 要素と Resource 要素のセマンティクスを理解するには、 [「the section called “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。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kafka-cluster:Connect",
        "kafka-cluster:AlterCluster",
        "kafka-cluster:DescribeCluster"
      ],
      "Resource": [
        "arn:aws:kafka:us-east-1:0123456789012: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-1:0123456789012:topic/MyTestCluster/*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "kafka-cluster:AlterGroup",
      "kafka-cluster:DescribeGroup"
    ],
    "Resource": [
      "arn:aws:kafka:us-east-1:0123456789012: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、「[the section called “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 ACL と同じ機能を提供しません。トピックへの WriteData アクセスを提供するだけでは、そのトピックに対して WriteDataIdempotently を実行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ている場合には該当しません。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 ACL の実装方法が異なるためです。

これを回避するために、以下のサンプルのようなポリシーを使用することが推奨されます。

```
{
  "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-1:0123456789012: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-1:0123456789012:topic/MyTestCluster/
abcd1234-0123-abcd-5678-1234abcd-1/TestTopic"
    ]
  }
]
```

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。WriteDataIdempotently はクラスターレベルのアクセス許可であることに注意する必要があります。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

[the section called “ブートストラップブローカーの取得”](#) を参照してください。

アクションとリソースのセマンティクス

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、「[the section called “認可ポリシーを作成する”](#)」を参照してください。

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	はい
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	はい
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	いいえ

[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:DescribeClusterDynamicConfiguration	Apache Kafka の DESCRIBE_CLUSTER_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	いいえ
kafka-cluster:AlterClusterDynamicConfiguration	Apache Kafka の ALTER_CONFIGS_CLUSTER_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	いいえ
kafka-cluster:WriteDataIdempotently	Apache Kafka の IDEMPOTENT_WRITE_CLUSTER_ACL に相当する、クラスターにデータをべき等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	はい

[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:CreateTopic	Apache Kafka の CREATECLUSTER/TOPIC ACL 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	はい
kafka-cluster:DescribeTopic	Apache Kafka の DESCRIBETOPIC ACL 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	はい
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	はい

[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に適用可能
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	はい
kafka-cluster:DescribeTopicDynamicConfiguration	Apache Kafka の DESCRIBE_CONFIGTOPIC 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	はい
kafka-cluster:AlterTopicDynamicConfiguration	Apache Kafka の ALTER_CONFIGTOPIC 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	はい

[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に適用可能
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	はい
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	はい
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	はい

[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に適用可能
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	はい
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	はい
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	はい

[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に適用可能
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	はい

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。

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、[DescribeCluster](#) API AWS Management Console または [describe-cluster](#) AWS CLI コマンドを使用してまたは から取得できます。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	<code>arn:aws:kafka:region:account-id :cluster/cluster-name /cluster-uuid</code>
トピック	<code>arn:aws:kafka:region:account-id :topic/cluster-name /cluster-uuid /topic-name</code>
グループ	<code>arn:aws:kafka:region:account-id :topic/cluster-name /cluster-uuid /group-name</code>
トランザクション ID	<code>arn:aws:kafka:region:account-id :transactional-id/cluster-name /cluster-uuid /transactional-id</code>

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 `_test` で、UUID が `abcd1234-0123-abcdabcd-1` `MyTestCluster` であるクラスター内の名前が `_test-5678-1234` 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 `MyTestCluster` 内のという名前のクラスターのすべての文字起こしで、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てから削除し、同じ名前で別のクラスターを作成する場合、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。

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、「[the section called “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	<code>kafka-cluster:*</code>
トピックの作成	<code>kafka-cluster:Connect</code> <code>kafka-cluster:CreateTopic</code>
データを生成する	<code>kafka-cluster:Connect</code> <code>kafka-cluster:DescribeTopic</code> <code>kafka-cluster:WriteData</code>
データの使用	<code>kafka-cluster:Connect</code> <code>kafka-cluster:DescribeTopic</code> <code>kafka-cluster:DescribeGroup</code> <code>kafka-cluster:AlterGroup</code> <code>kafka-cluster:ReadData</code>
データを無差別に生成する	<code>kafka-cluster:Connect</code> <code>kafka-cluster:DescribeTopic</code> <code>kafka-cluster:WriteData</code> <code>kafka-cluster:WriteDataIdempotently</code>

ユースケース	必須アクション
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
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

ユースケース	必須アクション
トピックを変更する	kafka-cluster:Connect kafka-cluster:DescribeTopic kafka-cluster:AlterTopic

相互 TLS 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 による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か AWS Private CA のいずれかです。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クラスターを作成するには](#)
- [認証を使用するようにクライアントを設定するには](#)
- [認証を使用してメッセージを生成および消費するには](#)

クライアント認証をサポートするクラスターを作成するには

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、MSK クラスター AWS Private CA 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 “を使用したクラスターの作成 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 “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、認証と転送時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には

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 “ステップ 3: 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 “ステップ 4: トピックを作成する”](#)」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
```

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
```

- 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。
- 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。

- 次のコマンドを実行して、AWS Private CA 署名され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
```

- 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 2 つの文字列を という名前の新しいファイルに貼り付けます signed-certificate-from-acm。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には

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 での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。

このトピックには、次のセクションが含まれています。

- [仕組み](#)
- [Amazon MSK クラスターの SASL/SCRAM 認証の設定](#)
- [ユーザーの使用](#)
- [制限事項](#)

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 認証の設定

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: 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。

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。以下のサンプルステップでは、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と、サンプルトピックのデータを生成したり消費したりする方法を示します。

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、*clusterARN* をクラスターの ARN に置き換えます。

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

2. 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、*BootstrapServer###*を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 つに置き換えます。

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --create --bootstrap-server BootstrapServerString --replication-factor 3 --partitions 1 --topic ExampleTopicName
```

3. 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";  
};
```

4. 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
```

5. *./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。
6. 次のコマンドを使用して、JVM *cacerts* フォルダの 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/jre/lib/security/cacerts /tmp/kafka.client.truststore.jks
```

7. 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
ssl.truststore.location=<path-to-keystore-file>/kafka.client.truststore.jks
```

8. 次のコマンドを使用して、ブートストラップブローカーの文字列を取得します。をクラスターの Amazon リソースネーム (ARN) `ClusterArn` に置き換えます。

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

コマンドの JSON 結果から、`BootstrapBrokerStringSaslScram` という名前の文字列に関連付けられた値を保存します。

9. 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` を前のステップで取得した値に置き換えます。

```
<path-to-your-kafka-installation>/bin/kafka-console-producer.sh --broker-list BootstrapBrokerStringSaslScram --topic ExampleTopicName --producer.config client_sasl.properties
```

10. 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` を以前に取得した値に置き換えます。

```
<path-to-your-kafka-installation>/bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerStringSaslScram --topic ExampleTopicName --from-beginning --consumer.config client_sasl.properties
```

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では、ZooKeeper ノードへのアクセスを制限して、ユーザーが ACLs を変更できないようにすることをお勧めします。詳細については、「[Apache へのアクセスの制御 ZooKeeper](#)」を参照してください。

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 にはプラグブルオーソライザーがあり、out-of-box オーソライザーの実装が付属しています。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 ACLs 「プリンシパル P は [許可/拒否] オペレーション O From Host H on any Resource R matching ResourcePattern RP」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承認する場合は、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。

クライアントに加えて、ブローカーがプライマリパーティションからメッセージを複製できるように、すべてのブローカーにトピックへのアクセスを許可する必要があります。ブローカーがトピックにアクセスできない場合、トピックのレプリケーションは失敗します。

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の取得”](#)」を参照してください。

```
<path-to-your-kafka-installation>/bin/kafka-acls.sh --authorizer-properties
--bootstrap-server BootstrapServerString --add --allow-principal
"User:CN=Distinguished-Name" --operation Read --group=* --topic Topic-Name
```

2. トピックへの読み取り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 --authorizer-properties  
--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 --authorizer-properties  
--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. API [ListNodes](#) または の [list-nodes](#) コマンド AWS CLI を使用して、クラスター内のブローカーのリストを取得します。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。

Apache へのアクセスの制御 ZooKeeper

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 での TLS セキュリティの使用 ZooKeeper](#)

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

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. 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のステップで取得した IP アドレスの 1 つと一致する場合、このネットワークインターフェイスはクラスターの一部である Apache ZooKeeper ノードに割り当てられます。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。次のステップでは、Apache ZooKeeper ノードに割り当てられた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ときは、ページ上部のアクションメニューを選択し、セキュリティグループの変更を選択します。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 での TLS セキュリティの使用 ZooKeeper

クライアントと Apache ZooKeeper ノード間の転送中の暗号化には 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 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 ZooKeeper。
- 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。TLS を有効にして Apache Kafka バージョン 2.5.1 以降で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、「[暗号化を開始する方法](#)」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、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、`--zk-tls-config-file` パラメータを使用して Apache ZooKeeper 設定を指定します。

次の例は、一般的な 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 (など `kafka-topics`) では、`KAFKA_OPTS`環境変数を使用して Apache ZooKeeper パラメータを設定する必要があります。次の例は、Apache ZooKeeper パラメータ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\n-Dzookeeper.client.secure=true\n-Dzookeeper.ssl.trustStore.location=/home/ec2-user/kafka.client.truststore.jks\n-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 --\nzookeeper ZooKeeperTLSConnectString --replication-factor 3 --partitions 1 --topic\nAWSKafkaTutorialTopic
```

Note

Apache ZooKeeper 設定ファイルで使用するパラメータの名前と `KAFKA_OPTS`環境変数で使用するパラメータの名前に一貫性がありません。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。

ログ記録

Apache Kafka ブローカーログは、Amazon CloudWatch Logs、Amazon S3、Amazon Data Firehose の 1 つ以上の送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。

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。ロググループ、S3 バケット、Firehose 配信スト

リームの 1 つ以上のタイプの宛先リソースに INFO レベルのブローカー CloudWatch ログを配信するように、新規または既存の MSK クラスターを設定できます。Firehose を通じて、配信ストリームから OpenSearch サービス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 AWS KMS マネージドキーで マネージドキー (SSE-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": "*"
}
```

を使用したブローカーログの設定 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。

を使用したブローカーログの設定 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": "ExampleBucketName",
      "Prefix": "ExamplePrefix",
      "Enabled": true
    },
    "Firehose": {
      "DeliveryStream": "ExampleDeliveryStreamName",
      "Enabled": true
    },
    "CloudWatchLogs": {
      "Enabled": true,
      "LogGroup": "ExampleLogGroupName"
    }
  }
}
```

API を使用したブローカーログの設定

[CreateCluster](#) または [UpdateMonitoring](#) オペレーションに渡す JSON でオプション `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 をログの送信先として使用し、DEBUG または TRACE レベルのログ記録を動的に有効にすると、Amazon MSK はログのサンプルを継続的に配信することがあります。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。

での AWS CloudTrail API コールのログ記録

Note

AWS CloudTrail ログは、[AWS CloudTrail 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. CloudTrail captures API コールのユーザー、ロール、または サービスによって実行されたアクションをイベントとして記録する AWS サービスであると統合されています。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。

証跡を作成する場合は、Amazon MSK の CloudTrail イベントなど、Amazon S3 バケットへのイベントの継続的な配信を有効にすることができます。Amazon S3 証跡を設定しない場合でも、CloudTrail コンソールのイベント履歴 で最新のイベントを表示できます。によって収集された情報

を使用して CloudTrail、Amazon MSK または Apache Kafka アクションに対して行われたリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の設定方法や有効化方法など 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。

の Amazon MSK 情報 CloudTrail

CloudTrail アカウントを作成すると、が Amazon Web Services アカウントで有効になります。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 CloudTrail イベント履歴の他の AWS サービスイベントとともに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、「[イベント履歴を使用した CloudTrail 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。証跡により 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。証跡は、AWS パーティションのすべてのリージョンからのイベントをログに記録し、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、他の Amazon サービスを設定して、CloudTrail ログで収集されたイベントデータをさらに分析し、それに基づく対応を行うことができます。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 サポートされているサービスと統合](#)
- [の Amazon SNS 通知の設定 CloudTrail](#)
- [複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信](#)

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) ユーザーの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 Element](#)」を参照してください。

例 : Amazon MSK ログ ファイルエントリ

証跡は、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できるようにする設定です。CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは任意ソースからの単一リクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどの情報を含みます。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、特定の順序では表示されません。

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"
  }
}

```

```
]
}
```

次の例は、kafka-cluster:CreateTopicアクションを示す CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "ABCDEFGH1IJKLMNOP34Q5",
    "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"
}
```

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 Amazon 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「コンプライアンスプログラム」](#)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、「[でのレポートのダウンロード AWS Artifact](#)」の」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。では、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- [「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。
- [「HIPAA セキュリティとコンプライアンスの設計」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、リソース設定が社内プラクティス、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、および高度に冗長な

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の手順で保護されています。

が AWS 公開した 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 クラスターへの接続

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 クラスターにアクセスできます。Kafka クライアントと MSK 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経由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 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てください。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、「[開始](#)」を参照してください。

クラスターの VPC の外部にあるクライアントから MSK クラスターに接続するには、「[クラスターの VPC 内からのアクセス AWS](#)」を参照してください。

トピック

- [公開アクセス](#)
- [クラスターの VPC 内からのアクセス AWS とクラスターの VPC 外からのアクセス](#)

公開アクセス

Amazon MSK には、Apache Kafka 2.6.0 以降のバージョンを実行している MSK クラスターのブローカーへの公開アクセスをオン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 AWS データ転送には標準データ転送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。

クラスターへの公開アクセスをオンにするには、最初にクラスターが次のすべての条件を満たしていることを確認してください。

- 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。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接続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 “セキュリティの更新”](#)」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 “セキュリティの更新”](#)」を参照してください。
- 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。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、「[the section called “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。

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 CLI、または Amazon AWS Management Console 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の取得”](#)」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。

を使用したパブリックアクセスの有効化 AWS CLI

1. *ClusterArn* および *Current-Cluster-Version* をクラスターの ARN と最新バージョンに置き換えて、次の AWS CLI コマンドを実行します。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 KTVDPKIKX0DER 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、次のコマンドを実行し、`ClusterOperationArn` を `update-connectivity` コマンドの出力で取得した ARN に置き換えます。

```
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パブリックアクセスをオンまたはオフにするには、「」を参照してください [UpdateConnectivity](#)。

Note

セキュリティ上の理由から、Amazon MSK は Apache ZooKeeper または KRaft コントローラーノードへのパブリックアクセスを許可しません。

クラスターの VPC 内からのアクセス AWS とクラスターの VPC 外からのアクセス

クラスターの Amazon VPC 内から AWS MSK クラスターに接続するには、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。ケーブルの一方の端はルーターに接続され、もう一方の端は AWS Direct Connect ルーターに接続されます。この接続を使用すると、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して、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、AW 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[AWS PrivateLink](#)) クラスターのマルチ VPC プライベート接続 (を使用) は、さまざまな Virtual Private Cloud (VPCs) および AWS アカウントでホストされている Kafka クライアントを Amazon MSK クラスターにすばやく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 ネットワークは廃止されました

Amazon MSK は、Amazon EC2-Classic ネットワークで実行されている Amazon EC2 インスタンスをサポートしなくなりました。

[EC2-Classic Networking is Retiring – Here's How to prepare](#)」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 [AWS PrivateLink](#)) クラスターのマルチ VPC プライベート接続 (を使用) は、さまざまな Virtual Private Cloud (VPCs) でホストされている Kafka クライアントと AWS アカウントをより迅速に 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、AWS ネットワーク内のすべてのトラフィック PrivateLink を維持しながら、を介して Amazon MSK クラスターに接続できます。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方法](#)
- [クラスターの認証スキームの更新](#)
- [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 Virtual Private Cloud (VPCs) および 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、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他の AWS アカウントに付与できます。

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 “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。
- 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方法

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 クライアントを内から 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。このプロセスでは、MSK クラスターの所有者が 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があります。

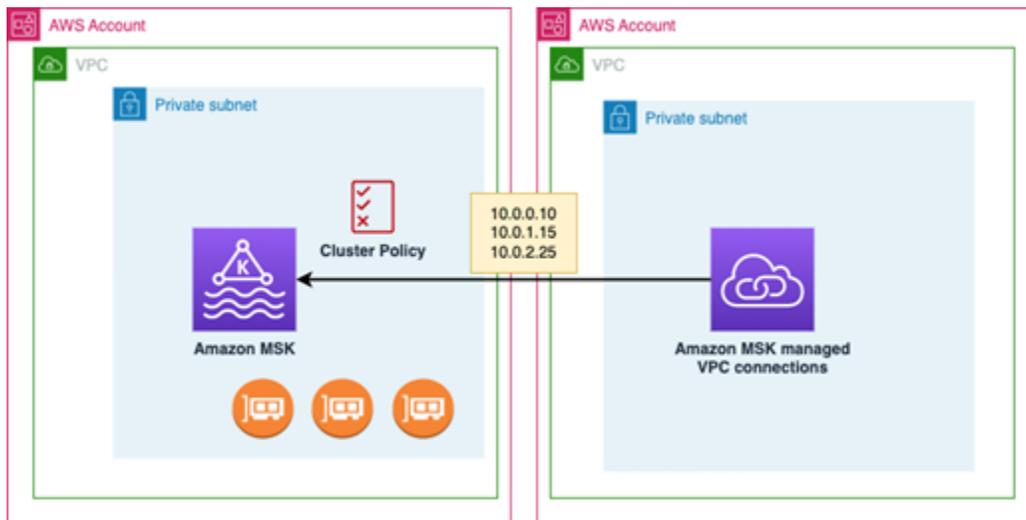
このチュートリアルの各部では、この例に適用するオプションをこちらで選択しています。これは、MSK クラスターまたはクライアント インスタンスを設定するために機能する唯一のオプションであることを意味するものではありません。

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 API [UpdateSecurity](#) 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 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。

- CreateVpc接続
- GetBootstrapブローカー
- 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。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "123456789012"
        ]
      },
      "Action": [
        "kafka:CreateVpcConnection",
```

```
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2"
    ],
    "Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/testing/
de8982fa-8222-4e87-8b20-9bf3cdfa1521-2"
}
]
}
```

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る `kafka:CreateVpcConnection`、`ec2:CreateTags`、`ec2:CreateVPCEndpoint` および `ec2:DescribeVpcAttribute` 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。

```
{
  "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 Availability ゾーンと、関連付けられたサブネットを選択します。Availability IDs は、AWS マネジメントコンソールの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、ブローカー API [GetBootstrap](#) 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することで、ブートストラップブローカーのリストを表示できます。
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. VPC の PrivateLink 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。

認証が完了すると、Amazon MSK は指定された VPC および認証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。

クラスターの認証スキームの更新

マルチ VPC プライベート接続は、複数の認証スキーム (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があります。

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 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があります。

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",
  "connectivityInfo": {
    "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 クラスター管理者は 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があります。

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、「[Amazon MSK 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。

PutClusterポリシー

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。

GetClusterポリシー

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。

DeleteClusterポリシー

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。

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

```
"Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/testing/
de8982fa-8222-4e87-8b20-9bf3cdfa1521-2"
  }
]
}
```

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。

```
{
  "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 を使用し AWS、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 を使用し AWS、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”](#)、内からのアクセスにポート 9098 を使用し AWS、パブリックアクセスにポート 9198 を使用します。
- TLS 暗号化 ZooKeeper を使用して Apache 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、非 MSK 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。これを行う方法の例については、「[「を使用してオンプレミスの Apache Kafka クラスターを Amazon MSK に移行する MirrorMaker」](#)」を参照してください。の使用方法については MirrorMaker、Apache Kafka 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 MirrorMaker で を設定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実行する手順の概要

1. 宛先 MSK クラスターを作成します
2. 送信先クラスターと同じ Amazon VPC 内の Amazon EC2 インスタンス MirrorMaker から開始します。
3. MirrorMaker ラグを検査します。
4. が MirrorMaker 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. をシャットダウンします MirrorMaker。

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、実行する必要が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。

このステップ MirrorMaker では、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、を使用することはできません。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。

- への読み取りアクセス権 `CLUSTER_ONPREM` と書き込みアクセス権を持つインスタンス `MirrorMaker` から開始します `CLUSTER_AWSMSK`。
- 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、は `CLUSTER_` のブートストラップブローカー `config/mirrormaker-producer.properties` を指します `AWSMSK`。例えば、で `bootstrap.servers=10.0.0.237:9092,10.0.2.196:9092,10.0.1.233:9092`。

- バックグラウンドで `MirrorMaker` 実行し続け、新しいデータはすべて `CLUSTER_ONPREM` `MirrorMaker mirrors` を引き続き使用します。
- ミラーリングの進行状況を確認するには、各トピックの最後のオフセットと `MirrorMaker` が消費している現在のオフセットの間の遅延を調べます。

`MirrorMaker` はコンシューマーとプロデューサーのみ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。

- がすべてのトピックのミラーリング `MirrorMaker` を完了したら、すべてのプロデューサーとコンシューマーを停止し、を停止します `MirrorMaker`。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クラスターに移行する

Apache `MirrorMaker 2.0` を使用して、非 MSK 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。これを行う方法の例については、[「を使用してオンプレミスの Apache Kafka クラスターを Amazon MSK に移行](#)

[する 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、「[MSK レプリケーター](#)」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リストは MirrorMaker 1.0 に適用されます。

- 送信先クラスター MirrorMaker で 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 MirrorMaker で を実行し、イベントがプロデューサーでバッファリングされ、ネットワークに問題がある場合、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.MaxValue`
 - `acks=all`
 - `max.block.ms = Long.MaxValue`
- 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`
- データ損失を回避するには、ソースで自動コミットをオフにします。これにより、 はコミットを制御 MirrorMaker できます。通常、送信先クラスターから `ack` を受信した後に行われます。プロデューサーに `acks=all` があり、レプリケート先クラスターに `min.insync.replicas` が 1 つ以上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、メッセージは宛先の複数のブローカーに保持されます。
- 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せん。

- 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 プロセスあたりのスレッド/コンシューマーの数を増やす — `num.streams`。
 - 高可用性を実現するためにスレッドを増やす前に、最初にマシン間で MirrorMaker プロセスの数を増やします。
 - 最初に同じマシンでプロセスの数を増やし、次に別のマシン (同じグループ ID) で MirrorMaker プロセスの数を増やします。
 - スループットが非常に高いトピックを分離し、個別の MirrorMaker インスタンスを使用します。
- 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。
 - コンテナを使用すると、インスタンスの MirrorMaker スケーリングと管理が容易になります。
- 通常、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 MirrorMaker。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、ストリーム (`num.streams`) の総数 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てください。

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やクラスターでの end-to-end 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 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 MirrorMaker 1 が所で指定するための高レベルの設定ファイルをサポートします。

Amazon MSK クラスターのモニタリング

Amazon MSK を使用して Amazon MSK クラスターのステータスをモニタリングする方法は複数あります。

- Amazon MSK は、クラスターがストレージ容量の制限に達しそうなときに自動的にストレージ容量アラートを送信することで、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](#)、AWS Health Dashboard、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。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](#) を参照してください。
- Amazon MSK は Apache Kafka メトリクスを収集し、Amazon CloudWatch に送信して表示できるようにします。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 でクラスターをモニタリングする方法については、「[the section called “Prometheus によるオープンモニタリング”](#)」を参照してください。

トピック

- [でモニタリングするための Amazon MSK メトリクス CloudWatch](#)
- [を使用した Amazon MSK メトリクスの表示 CloudWatch](#)
- [コンシューマーラグモニタリング](#)
- [Prometheus によるオープンモニタリング](#)
- [Amazon MSK ストレージ容量アラート](#)

でモニタリングするための Amazon MSK メトリクス CloudWatch

Amazon MSK は Amazon と統合 CloudWatch されているため、Amazon MSK クラスターのメトリクスを収集、表示、分析 CloudWatch できます。MSK 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、にプッシュされます CloudWatch。MSK クラスターのモニタリングレベルは、DEFAULT、PER_BROKER、PER_TOPIC_PER_BROKER、または PER_TOPIC_PER_PARTITION

のいずれかに設定できます。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。

DEFAULT 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
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。
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 」

名前	表示可能なタイミング	ディメンション	説明
			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。
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。

名前	表示可能なタイミング	ディメンション	説明
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。
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。

名前	表示可能なタイミング	ディメンション	説明
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。
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 GlobalPartitionCount 場合よりも PartitionCount 値の合計が高くなる可能性があります。

名前	表示可能なタイミング	ディメンション	説明
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)。
KafkaApplicationLogsDiskUsed	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。
KafkaDataLogsDiskUsed (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。
KafkaDataLogsDiskUsed (Cluster Name 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]	データログに使用されるディスク容量の割合。

名前	表示可能なタイミング	ディメンション	説明
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。
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メモリのサイズ (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
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) 。
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。

名前	表示可能なタイミング	ディメンション	説明
NetworkRx 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	ドロップされた受信パッケージの数。
NetworkRx 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受信エラーの数。
NetworkRx 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受信されたパケットの数。
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。

名前	表示可能なタイミング	ディメンション	説明
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。
OfflinePartitionsCount	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。
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。

名前	表示可能なタイミング	ディメンション	説明
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。
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。

名前	表示可能なタイミング	ディメンション	説明
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。
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。

名前	表示可能なタイミング	ディメンション	説明
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。
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をミリ秒単位で表します。
ZooKeeperSessionState	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。NOT_CONNECTED: '0.0'、ASSOCIATING: '0.1'、CONNECTING: '0.5'、CONNECTEDREADONLY: '0.8'、CONNECTED: '1.0'、CLOSED: '5.0'、AUTH_FAILED: '10.0' のいずれかであるブローカーの ZooKeeper セッションの接続ステータス。

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。

PER_BROKER モニタリングレベルからスタートして使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
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

名前	表示可能なタイミング	説明
		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)。
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)。
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。
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 認証リクエストの数。
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。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。

名前	表示可能なタイミング	説明
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。
ProduceLocalTimeMs Mean	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)。
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)。

名前	表示可能なタイミング	説明
ProduceTotalTimeMs Mean	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。
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。

名前	表示可能なタイミング	説明
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、スケジューリング待ちのタスクの数。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。
RemoteFetchErrorsPerSec (RemoteReaderErrorPerSec 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。

名前	表示可能なタイミング	説明
RemoteFetchRequestPerSec (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。
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)。

名前	表示可能なタイミング	説明
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。
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。

名前	表示可能なタイミング	説明
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。
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。

PER_TOPIC_PER_BROKER モニタリングレベルからスタートして使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
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。
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 (RemoteRe	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

名前	表示可能なタイ ミング	説明
adErrorPerSec in v2.8.2.tiered)		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。
RemoteFetchRequestPerSec (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。

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。

PER_TOPIC_PER_PARTITION モニタリングレベルからスタートして使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
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。

を使用した Amazon MSK メトリクスの表示 CloudWatch

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) を選択します。
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](#) および [GetMetric統計](#) オペレーションを使用します。

コンシューマーラグモニタリング

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 を使用する CloudWatch か、Prometheus でモニタリングを開きます。

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 によるオープンモニタリングを通じて取得できるコンシューマーラグメトリクスとして `EstimatedMaxTimeLag`、`EstimatedTimeLag`、`MaxOffsetLag`、`OffsetLag`、および `SumOffsetLag` を提供します。これらのメトリクスの詳細については、「[the section called “でモニタリングするための Amazon MSK メトリクス CloudWatch”](#)」を参照してください。

Note

コンシューマーラグメトリクスは、STABLE 状態のコンシューマーグループに対してのみ表示されます。再調整が正常に完了すると、コンシューマーグループは STABLE になり、パーティションがコンシューマー間で均等に分散されます。

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。

Prometheus によるオープンモニタリング

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 を使用して、MSK クラスターを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 でフォーマットされたメトリクスと互換性のあるツールや、[Datadog](#)、[Lenses](#)、[New Relic](#)、[Sumo Logic](#) など、Amazon MSK オープンモニタリングに統合された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。

オープンモニタリングを有効にして Amazon MSK クラスターを作成する

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 つのエクスポートを同時に無効にすることはできません。

API を使用する場合

- [CreateCluster](#) オペレーションを呼び出し、を指定します OpenMonitoring。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 つのエクスポートを同時に無効にすることはできません。

既存の Amazon MSK クラスターのオープンモニタリングの有効化

オープンモニタリングを有効にするには、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. Edit (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. [変更の保存] を選択します。

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 つのエクスポートを同時に無効にすることはできません。

API を使用する場合

- [UpdateMonitoring](#) オペレーションを呼び出し、`OpenMonitoring`、`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 つのエクスポートを同時に無効にすることはできません。

Amazon EC2 インスタンスでの Prometheus ホストの設定

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 ファイルのターゲットとして追加します。例: `controller_dns_1:11001`、を実際のコントローラー DNS 名 `controller_dns_1` 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

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。

Amazon Managed Service for Prometheus への Prometheus メトリクスの保存

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ル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。

Amazon MSK ストレージ容量アラート

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 CloudWatch、クラスターのストレージ容量のモニタリングに役立つ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](#)、および[アカウントの E EventBridge](#)メール連絡先に自動的に送信します。AWS これ

らのアラートを Slack または New Relic や Datadog などのツールに配信するように [Amazon を設定 EventBridge](#)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。

Amazon MSK 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](#) APIs を使用して `CustomerActionStatus`、クラスターのすべてのアラートを表示します。
- [AWS Health Dashboard](#) に移動して、MSK およびその他の AWS のサービスからのアラートを表示します。
- [AWS Health API](#) と [Amazon EventBridge](#) を設定して、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 での LinkedIn の Cruise Control for Apache Kafka の使用

LinkedIn の Cruise Control を使用して、Amazon MSK クラスターの再調整、異常の検出と修正、クラスターの状態と正常性のモニタリングを行うことができます。

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 によるオープンモニタリング”](#)」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 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.Prometheus

# 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
```

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。[DescribeCluster](#) API オペレーション (または同等の 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](#)。

```
{
  "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."
}
]
}
```

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-kafka-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。

Cruise Control for Amazon MSK の自動デプロイテンプレート

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に関するより深い洞察を得て、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 でプロビジョニングされたクラスターのサポート。
- [PlainText および IAM](#) による柔軟な認証。
- Cruise Control に Zookeeper の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提供することで、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。

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 AWS アカウント内でリージョン固有です。

Amazon MSK クォータ

- アカウントあたり最大 90 のブローカー。 ZooKeeper モードクラスターあたり 30 のブローカー。 KRaft モードクラスターあたり 60 のブローカー。クォータの引き上げをリクエストするには、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](#)。
- ブローカーごとに最低 1 GiB のストレージ。
- ブローカーごとに最大 16384 GiB のストレージ。
- [the section called “IAM アクセスコントロール”](#) を使用するクラスターは、ブローカーごとに常に最大 3000 の TCP 接続を持つことができます。この制限を引き上げるには、Kafka AlterConfig API `listener.name.client_iam.max.connections` 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。
- TCP 接続の制限。接続レートバーストを有効にすると、MSK は 1 秒あたり 100 接続を許可します。例外は、`kafka.t3.small` インスタンスタイプで、接続レートバーストを有効にして 1 秒あたり 4 つの接続が許可されます。接続レートバーストが有効になっていない古いクラスターでは、クラスターにパッチが適用されると、この機能が自動的に有効になります。

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。例えば、クライアントが 1 秒後に接続を再試行するようにする場合は、`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。

- アカウントごとに最大 100 の構成。クォータの調整をリクエストするには、AWS コンソールサポートセンターにアクセスし、[サポートケースを作成](#)してください。
- 設定あたり最大 50 のリビジョン。
- 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 [the section called “クラスターの適切なサイズ設定: 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。

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 レプリケーターは、ソート順に最大 750 個のトピックのみをレプリケートします。さらに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、AWS コンソールの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](#)。TopicCount「」メトリクスを使用して、レプリケートされるトピックの数を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。クォータの引き上げをリクエストするには、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](#)。
- MSK レプリケーターレコードサイズ - 最大 10MB のレコードサイズ (message.max.bytes)。クォータの引き上げをリクエストするには、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](#)。

MSK サーバーレス クォータ

Note

クォータの制限に問題がある場合は、AWS サポート [ケースを作成してサポート](#) にお問い合わせください。

特に明記されていない限り、制限はクラスターあたりの制限です。

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

ディメンション	クォータ	クォータ違反の結果
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 MB	リクエストが で失敗する ErrorCode: 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
リクエストあたりの最大フェッチバイト数	55 MB	リクエストが で失敗する ErrorCode: INVALID_REQUEST
ユーザーグループの最大数	500	JoinGroup リクエストが失敗する
パーティションの最大数 (リーダー)	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。クォータの調整をリクエストするには、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 。	リクエストが で失敗する ErrorCode: 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	リクエストが で失敗する ErrorCode: "PUT_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

ディメンション	クォータ	クォータ違反の結果
最大パーティションサイズ (圧縮されたトピック用)	250 GB	リクエストが で失敗する ErrorCode: "PUT_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. クォータの調整をリクエストするには、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 。	

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、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](#)。

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 “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。

MSK の統合

このセクションでは、Amazon MSK と統合する AWS 機能への参照を提供します。

トピック

- [Amazon MSK 用の Amazon Athena コネクタ](#)
- [Amazon Redshift ストリーミングデータインジェスト](#)
- [Firehose](#)
- [Amazon MSK コンソールから Amazon EventBridge Pipes にアクセスする](#)

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 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。

Firehose

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サーバーレスのノーコード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り、S3 ロケーションに配信するように 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の量に基づいて自動的にスケーリングし、Kafka から取り込まれたバイト数に対してのみ料金を支払います。

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 を使用して Amazon Kinesis Data Firehose に書き込む」](#)
- 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。Pipes は、サポートされているソースとターゲットの統合を目的として point-to-point 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、および 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、高度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てパイプの詳細を表示します。
- EventBridge コンソールを起動して、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 ユーザーガイド」の「Amazon Managed Streaming for Apache Kafka cluster as a source」](#)を参照してく

ださい。 EventBridge パイプ全般の詳細については、EventBridge 「パイプ [EventBridge](#) 」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、Amazon MSK クラスターをパイプソースとして指定した EventBridge コンソールのパイプの作成ページが起動します。詳細については、「Amazon [ユーザーガイド](#)」の [EventBridge](#) 「パイプの作成」を参照してください。 EventBridge

- [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。

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 コミュニティは、リリース日から約 12 か月間、バージョンをサポートします。詳細については、[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

Apache Kafka バージョン	MSK リリース日	サポート終了日
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
2.7.2	2021-12-21	2024-09-11
2.8.0	--	2024-09-11
2.8.1	2022-10-28	2024-09-11
2.8.2 階層化	2022-10-28	発表予定
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	2025-06-17
3.5.1 (推奨)	2023-09-26	--
3.6.0	2023-11-16	--
3.7.x	2024-05-29	--

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 バージョンサポートポリシー](#)。

Apache Kafka バージョン 3.7.x (本番環境に対応した階層型ストレージ搭載)

MSK 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ようにクラスターを作成したり、既存のクラスターをアップグレードしたりできます。このバージョン命名の変更により、Apache Kafka コミュニティによってリリースされた 3.7.1 などの新しいパッチ修正バージョンを採用する必要がなくなりました。Amazon MSK は 3.7.x を自動的に更新して、将来のパッチバージョンが利用可能になるとサポートします。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で利用できるセキュリティと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は、バージョンの互換性を破るものではなく、クライアントアプリケーションの読み取り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利点を得ることができます。などのインフラストラクチャ自動化ツールが CloudFormation、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 コントローラーは追加料金なしで含まれ、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 では、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 30 ブローカークォータと比較して、制限の引き上げをリクエストすることなく、クラスターごとにより多くのパーティションをホストするために最大 60 のブローカーを追加できます。MSK での KRaft の詳細については、[KRaft モード](#) を参照してください。

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。主な改善点には、クライアントのリーダー検出の最適化とログセグメントフラッシュの最適化オプションが含まれます。改善点とバグ修正の完全なリストについては、「[3.7.0 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 3.6.0 (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 (本番移行可能階層型ストレージ) 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、コンシューマー向けの新しいラック対応パーティション割り当ての導入が含まれます。Amazon MSK は、このリリースでもクォーラム管理のために Zookeeper を引き続き使用および管理します。改善とバグ修正の完全なリストについては、3.5.1 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 3.5.1 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取得するため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のために Zookeeper を引き続き使用および管理します。改善とバグ修正の完全なリストについては、「3.4.0 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 3.4.0 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取得するため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のために Zookeeper を引き続き使用および管理します。改善とバグ修正の完全なリストについては、3.3.2 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 3.3.2 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完全なリストについては、「[3.3.1 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 3.3.1 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 IDs。MSK は、このリリースの安定性のために、クォーラム管理のために Zookeeper を引き続き使用および管理します。改善とバグ修正の完全なリストについては、3.1.1 および 3.2.0 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 3.1.1 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 リリースノート](#)と[3.1.1 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、「[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 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 ZooKeeper と新機能が含まれています。Amazon MSK には TLS ZooKeeper エンドポイントが用意されており、[DescribeCluster](#) オペレーションでクエリを実行できます。

[DescribeCluster](#) オペレーションの出力には、TLS ゾーンキーパー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.aws kafkatutorialc.abcd123.c3.kafka.us-east-1.amazonaws.com:2182,z-2.aws kafkatutorialc.abcd123.c3.kafka.us-east-1.amazonaws.com:2182,z-1.aws kafkatutorialc.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 での TLS セキュリティの使用 ZooKeeper](#)」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 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 バージョンの更新](#)」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 クラスターのトラブルシューティング](#) 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IDs、AWS Resource Access Manager ユーザーガイドの「[リソースの AZ IDs](#)」を参照してください。

設定プロパティの設定については、「[構成](#)」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 バージョンサポートポリシー](#) と の手順について説明します [Apache Kafka バージョンの更新](#)。Kafka バージョンをアップグレードする場合は、「」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 [バージョンアップグレードのベストプラクティス](#)。

Amazon MSK バージョンサポートポリシー

このセクションでは、Amazon MSK がサポートする Kafka バージョンのサポートポリシーについて説明します。

- Kafka のすべてのバージョンは、サポート終了日に達する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、「」を参照してください[サポート対象の Apache Kafka バージョン](#)。サポート終了日より前に、MSK クラスターを推奨 Kafka バージョン 以降にアップグレードします。Apache Kafka バージョンの更新の詳細については、「」を参照してください[Apache Kafka バージョンの更新](#)。サポート終了日以降に 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。
- MSK は、サポート終了日が公開された Kafka バージョンを使用する新しく作成されたクラスターのサポートを段階的に廃止します。

Apache Kafka バージョンの更新

既存の MSK クラスターを新しいバージョンの Apache Kafka に更新できます。以前のバージョンに更新することはできません。MSK クラスターの Apache Kafka バージョンを更新するときは、クライアント側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てください。Amazon MSK は、サーバーソフトウェアのみを更新します。クライアントは更新されません。

更新中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。

Important

[the section called “クラスターの適切なサイズ設定: 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 MSK クラスターの Apache Kafka バージョンを更新することはできません。

を使用した Apache Kafka バージョンの更新 AWS Management Console

1. <https://console.aws.amazon.com/msk/> で Amazon MSK コンソールを開きます。
2. Apache Kafka バージョンを更新する MSK クラスターを選択します。
3. [プロパティ] タブの [Apache Kafka バージョン] セクションで [アップグレード] を選択します。

を使用した Apache Kafka バージョンの更新 AWS CLI

1. 次のコマンドを実行し、 をクラスターの作成時に取得した Amazon リソースネーム (ARN) *ClusterArn* に置き換えます。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かります。詳細については、「[the section called “クラスターの一覧表示”](#)」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、クラスターを更新できる Apache Kafka バージョンのリストが含まれています。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. 次のコマンドを実行し、 をクラスターの作成時に取得した Amazon リソースネーム (ARN) *ClusterArn* に置き換えます。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かります。詳細については、「[the section called “クラスターの一覧表示”](#)」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、前のコマンドの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 `KTVDPKIKX0DER` 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、次のコマンドを実行し、`ClusterOperationArn` を `update-cluster-kafka-version` コマンドの出力で取得した ARN に置き換えます。

```
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 を使用した Apache Kafka バージョンの更新

1. [GetCompatibleKafkaVersions](#) オペレーションを呼び出して、クラスターを更新できる 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更新します。

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のクライアントの継続性を確保するには、クライアントと Apache Kafka トピックの設定を次のように確認します。

- トピックレプリケーション係数 (RF) を、2 つの AZ クラスター2の場合は最小値 に設定し、3 つの AZ クラスター3の場合は最小値 に設定します。の RF 値を指定すると、パッチ適用中にオフラインパーティションが発生する2可能性があります。
- 最小同期レプリカ (minISR) を最大値 RF - 1に設定して、パーティションレプリカセットが 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るようにします。
- 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あると、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可能になります。複数のブローカーとの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の取得」](#)を参照してください。
- 新しいバージョンで利用できる機能を利用するには、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) 日の対象ではなく、EOL 日までに完了する必要はありません。Apache Kafka は、[古いクライアントが新しいクラスターと連携できるようにする双方向のクライアント互換性ポリシー](#)を提供します。その逆も同様です。
- バージョン 3.x.x を使用する Kafka クライアントには、acks=allおよび のデフォルトが付属している可能性がありますenable.idempotence=true。acks=allは以前のデフォルト とは異なりacks=1、すべての同期レプリカがプロデュースリクエストを承認できるようにすることで、耐久性を高めます。同様に、 のデフォルトは以前は enable.idempotenceでしたfalse。デフォルトenable.idempotence=trueとしてに変更すると、メッセージが重複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レイテンシーが追加され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 および MSK 機能を利用できます。

Amazon MSK クラスターのトラブルシューティング

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。

トピック

- [ボリュームの置換により、レプリケーションの過負荷によるディスクの飽和が発生する](#)
- [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が、コンシューマーがデータを受信できない](#)
- [AWS CLI Amazon MSK を認識しない](#)
- [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](#)
- [プロデューサーが取得する NotLeaderForPartitionException](#)
- [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](#)
- [MSK サーバーレス: クラスターの作成に失敗する](#)

ボリュームの置換により、レプリケーションの過負荷によるディスクの飽和が発生する

計画外のボリュームハードウェア障害時に、Amazon MSK はボリュームを新しいインスタンスに置き換える場合があります。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、新しいボリュームを再入力します。パーティションがレプリケートされて追いつくと、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

ボリューム交換から回復するブローカーでは、さまざまなサイズのパーティションが他のパーティションよりもオンラインに戻る可能性があります。これは、これらのパーティションが、他のパーティションにまだ追いつい (レプリケート) ているのと同じブローカーからのトラフィックを処理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ことがあります。デフォルトの場合、これは 1 秒あたり 250 MiB です。この飽和状態が発生すると、既に追いついたパーティションが影響を受け、(リモートアックによるリーダーパーティションだけでなく) 追いついたパーティションと 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acks=all。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 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。
- 基盤となるボリュームが飽和する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。高スループットのレプリケーションケースでは、最小スループット値 500 MiB/秒が推奨されますが、必要な実際の値はスループットとユースケースによって異なります[プロビジョニングストレージのスループット](#)。
- レプリケーションプレッシャーを最小限に抑えるには、`num.replica.fetchers`をデフォルト値の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 バージョンの更新](#)」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ー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、コンシューマー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。

コーディネーターブローカーノードの再起動

コーディネーター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. [RebootBroker](#) API アクションを使用して、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再起動します。

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セットアップしようとすると、2つの例外のいずれかが発生する可能性があります。

`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、アカウント内の既存の Amazon CloudWatch Logs ポリシーを選択し、次の JSON を追加します。

```
{"Sid": "AWSLogDeliveryWrite", "Effect": "Allow", "Principal":
{"Service": "delivery.logs.amazonaws.com"}, "Action":
["logs:CreateLogStream", "logs:PutLogEvents"], "Resource": ["*"]}
```

上記の JSON を既存のポリシーに追加しようとしても、選択したポリシーの最大長に達したことを示すエラーが表示された場合は、JSON を別の Amazon CloudWatch Logs ポリシーに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配信を設定してみてください。

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 “ステップ 3: 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、「[移行](#)」を参照してください。

AWS CLI Amazon MSK を認識しない

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。をアップグレードする方法の詳細な手順については 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[仕組み](#)。

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。

プロデューサーが取得する NotLeaderForPartitionException

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。[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「[認証と 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 “を使用したブートストラップブローカーの取得 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の取得”](#)」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
```

port-number は次のようになります。

- 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は 9094。
- 9092 クラスターが TLS 認証を使用しない場合。
- 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、接続に問題がないことを意味します。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の取得”](#)」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は、クライアントマシンの設定に基づくガイダンスについて、次のサブセクション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、「[開始](#)」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セットアップ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、 「の [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は、「[Amazon MSK クォータ](#)」ページを参照してください。

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。

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。

クラスターの適切なサイズ設定: ブローカーあたりのパーティション数

次の表は、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) を示しています。

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)
kafka.t3.small	300
kafka.m5.large または kafka.m5.xlarge	1,000
kafka.m5.2xlarge	2000
kafka.m5.4xlarge 、 kafka.m5.8xlarge 、 kafka.m5.12xlarge 、 kafka.m5.16xlarge 、 または kafka.m5.24xlarge	4000
kafka.m7g.large または kafka.m7g.xlarge	1,000
kafka.m7g.2xlarge	2000
kafka.m7g.4xlarge 、 kafka.m7g.8xlarge 、 kafka.m7g.12xlarge 、 または kafka.m7g.16xlarge	4000

ブローカーあたりのパーティション数が推奨値を超え、クラスターが過負荷になると、以下のオペレーションを実行できなくなる可能性があります。

- クラスター設定の更新
- クラスターをより小さなブローカーサイズに更新する
- 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いと、Prometheus スクレイピング CloudWatch で Kafka メトリクスが欠落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、「[Apache Kafka Supports 200K Partitions Per Cluster](#)」を参照してください。また、ブローカーに適したサイズを決定するために、独自のテストを実行することをお勧めします。さまざまなブローカーサイズの詳細については、「」を参照してください[the section called “ブローカーサイズ”](#)。

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

MSK クラスターに適切なブローカーの数を決定し、コストを理解するには、[MSK サイジング設定と価格設定](#)のスプレッドシートを参照してください。このスプレッドシートは、MSK クラスターのサイジング設定と、同様のセルフマネージド EC2-based Apache Kafka クラスターと比較した 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。このシートに記載されている見積もりは保守的なものであり、新しいクラスターの出発点となります。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切に設定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、スケールアップまたはスケールアウトを行うべきタイミングなどの質問への回答や、本番稼働のクラスターのサイズを継続的に検証する方法についてのガイダンスも提供しています。

m5.4xl、m7g.4xl 以上のインスタンスのクラスタースループットを最適化する

m5.4xl、m7g.4xl、またはそれ以上のインスタンスを使用する場合、num.io.threads と num.network.threads の設定を調整することで、クラスターのスループットを最適化できます。

`num.io.threads` は、ブローカ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です。インスタンスサイズでサポートされる CPU コア数までスレッドを追加すると、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、すべての着信リクエストを受信してレスポンスを返すためにブローカーが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で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完全に使用できます。

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	<code>num.io.threads</code> の推奨値	<code>num.network.threads</code> 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

最新の Kafka AdminClient を使用してトピック ID の不一致の問題を回避する

Kafka バージョン 2.8.0 より前のバージョンと フラグを使用して Kafka AdminClient バージョン 2.8.0 以降を使用するクラスター--zookeeperのトピックパーティション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 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。
- 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、「[the section called “ブートストラップブローカーの取得”](#)」を参照してください。

CPU 使用率をモニタリングする

Amazon MSK では、ブローカーの合計 CPU 使用率 (CPU User + CPU System として定義) を 60% 未満に維持することを強く推奨しています。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。これが必要な場合の1つの例は、Amazon MSK がブローカーの障害を検出して回復する場合です。この場合、Amazon MSK はパッチ適用などの自動メンテナンスを実行します。もう1つの例は、ユーザーがブローカーのサイズ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場合です。これら2つの場合、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デプロイします。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、このような運用イベントに耐えるのに十分な CPU ヘッドルームをクラスターに確保できます。

[Amazon CloudWatch Metric Math](#) を使用して、である複合メトリクスを作成できます CPU User + CPU System。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ルリングします。

- オプション 1 (推奨): [ブローカーサイズを次に大きいサイズ](#)に更新します。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 `kafka.m5.xlarge`。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す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put 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、この形式のスケールリングでは通常、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クラスターを拡張し、`kafka-reassign-partitions.sh` 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。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。Amazon MSK は、[Cruise Control](#) を使用して、パーティション割り当てを介してロード ディストリビューション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

メッセージのディスク容量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 “Auto Scaling”](#) 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。

アラームをセットアップして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「[クラスターのモニタリング](#)」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 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% を超えたときにアクションを実行する CloudWatch アラームを作成することをお勧めします。

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 ZooKeeper は MSK クラスターに追加されず、Apache 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クラスターオペレーションについては、「[仕組み](#)」を参照してください。

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

同じクラスター上の異なるブローカーにパーティションを移動するには、`kafka-reassign-partitions.sh` 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。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、ブローカーを削除するためにパーティションを移動したりした後、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、そのクラスターのバランスを再調整できます。クラスターにブローカーを追加する方法については、「[the section called “クラスターの拡張”](#)」を参照してください。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。

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。

ドキュメントの最終更新日：2024年6月25日

変更	説明	日付
Graviton アップグレードインプレース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
ブローカー削除機能を追加しました。	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、一連のブローカーを削除することで、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の容量を減らすことができます。	2024-5-16
WriteDataIdempotently が追加されました AWSMSKReplicatorExecutionRole	WriteDataIdempotently 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、許可が AWSMSKReplicatorExecutionRole ポリシーに追加されました。	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、AWS Graviton プロセッサ (Amazon Web Services によって構築されたカスタム Arm ベースのブ	2024-2-07

変更	説明	日付
	ロセッサ) を使用する M7g ブローカーの南米 (サンパウロ) および中東 (バーレーン) リージョンの可用性をサポートするようになりました。	
Graviton M7g ブローカーを中国リージョンにリリースする	Amazon MSK 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 レプリケーター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 は、Amazon MSK レプリケーターをサポートする新しいAWSMSKReplicatorExecutionRole ポリシーを追加しました。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

変更	説明	日付
M7g Graviton のサポート	Amazon MSK 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
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 レプリケーターには、AmazonMSKFullAccess ポリシーの更新が含まれています。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の <code>AWSServiceRoleForKafka</code> 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、クラスターブローカーが VPC 内のクライアントにアクセスできるようになります。Amazon MSK は、 <code>DescribeVpcEndpoints</code> 、 <code>ModifyVpcEndpoint</code> 、および <code>DeleteVpcEndpoints</code> 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の使用」 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、「 ブローカーサイズの更新 」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 クラスターのモニタリング 」を参照してください。	2020-11-23
Cruise Control のサポート	Amazon MSK は、の Cruise Control LinkedIn をサポートするようになりました。詳細については、「 Amazon MSK での LinkedIn の Cruise Control for Apache Kafka の使用 」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、「 Auto Scaling 」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
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

変更	説明	日付
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
ブローカーログのストリーミングのサポート	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。Firehose は、これらのログを OpenSearch Service など、サポートする宛先に配信できます。	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
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

変更	説明	日付
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

AWS 用語集

最新の AWS 用語については、「AWS の用語集 リファレンス」の [AWS 「用語集」](#) を参照してください。

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。