aws

ユーザーガイド

AWS Database Migration Service

AWS Database Migration Service: ユーザーガイド

Table of Contents

AWS Database Migration Service とは

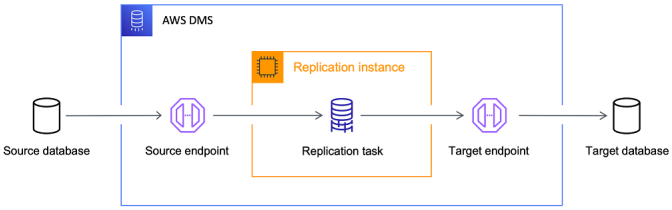
AWS Database Migration Service (AWS DMS) は、リレーショナルデータベース、データウェアハウス、NoSQL データベース、その他の種類のデータストアを移行できるようにするクラウドサービスです。AWS DMS を使用して、データの AWS クラウド への移行や、クラウドとオンプレミスセットアップを組み合わせたものの間でのデータ移行ができます。

AWS DMS を使用すると、ソースデータストアの検出、ソーススキーマの変換、データの移行ができます。

- ソースデータインフラストラクチャの検出には、DMS Fleet Advisor を利用できます。このサービスは、オンプレミスのデータベースと分析サーバーからデータを収集し、AWS クラウドに移行できるサーバー、データベース、スキーマのインベントリを構築します。
- 別のデータベースエンジンに移行する場合、DMS Schema Conversion を使用できます。このサービスは、ソーススキーマを自動的に評価して、新しいターゲットエンジンに変換します。AWS Schema Conversion Tool (AWS SCT) をローカル PC にダウンロードして、ソーススキーマを変換するという方法もあります。
- ソーススキーマを変換し、変換したコードをターゲットデータベースに適用した後、AWS DMS を使用してデータを移行できます。1 回限りの移行を実行したり、継続的に変更をレプリケートしてソースとターゲットの同期を維持したりすることができます。AWS DMS は AWS クラウド の一部であるため、AWS のサービスが提供するコスト効率性、市場投入の迅速化、セキュリティ、柔軟性も得られます。

基本レベルの場合、AWS DMS はレプリケーションソフトウェアを実行する AWS クラウド 内のサーバーとなります。お客様は、ソースとターゲット間の接続を作成して、データの抽出元とロード先を AWS DMS に指示します。その後、このサーバーで実行するタスクのスケジュールを設定して、データを移行します。ターゲットにテーブルとテーブルのプライマリキーが存在しない場合、AWS DMS が自動的に作成します。必要に応じて、ターゲットテーブルを手動で事前に作成することもできます。または、 AWS Schema Conversion Tool (AWS SCT) を使用して、ターゲットテーブル、インデックス、ビュー、トリガーなどの一部またはすべてを作成することもできます。

次の図は、AWS DMS のレプリケーションプロセスを説明しています。



リファレンス

- AWS DMS をサポートするAWS リージョン – AWS DMS をサポートする AWS リージョンの詳細100については、「AWS DMS レプリケーションインスタンスの使用」を参照してください。
- データベース移行のコスト – データベース移行のコストについては、「AWS Database Migration Service の料金ページ」を参照してください。
- AWS DMS の特徴と利点 – AWS DMS の特徴と利点については、「AWS Database Migration Service の特徴」を参照してください。
- 利用できるデータベースオプション – Amazon Web Services で利用できるさまざまなデータベースのオプションの詳細については、「データベースサービスの選択」を参照してください。

AWS DMS が実行する移行タスク

AWS DMS は、移行プロジェクトに伴う困難なタスクや手間のかかるタスクの多くをユーザーに代わって実行します。

- 従来のソリューションでは、キャパシティ分析の実行、ハードウェアとソフトウェアの調達、システムのインストールと管理、インストールのテストとデバッグが必要となります。AWS DMS は、移行に必要となるすべてのハードウェアとソフトウェアのデプロイ、管理、モニタリングを自動的に管理します。AWS DMS 設定プロセスを開始してから数分以内に移行を実行できます。
- AWS DMS を使用すると、実際のワークロードに合わせて、必要に応じて移行リソースをスケールアップ (またはスケールダウン) できます。例えば、ストレージの追加が必要であると判断した場合、割り当て済みのストレージを簡単に増やして、移行を、通常は数分以内に再開できます。
- AWS DMS は従量制料金モデルを採用しています。前払いの購入費用や継続的なメンテナンスコストが必要となる従来のライセンスモデルとは異なり、使用した AWS DMS リソースに対してのみ課金されます。
- AWS DMS は、ハードウェアとソフトウェア、ソフトウェアのパッチ適用、エラーレポートなど、移行サーバーをサポートするすべてのインフラストラクチャを自動的に管理します。
- AWS DMS は、自動フェイルオーバーを提供します。何らかの理由でプライマリレプリケーションサーバーに障害が発生した場合、サービスをほとんどまたはまったく中断させることなく、バックアップレプリケーションサーバーが引き継ぐことができます。
- AWS DMS Fleet Advisor は、データインフラストラクチャのインベントリを自動的に作成します。DMS Fleet Advisor は、移行候補を特定して、移行計画に役立つレポートを作成します。

- AWS DMS Schema Conversion は、ソースデータプロバイダーの移行の複雑さを自動的に評価します。また、データベーススキーマとコードオブジェクトをターゲットデータベースと互換性のある形式に変換して、変換済みのコードを適用します。
- AWS DMS を使用すると、現在実行しているデータベースエンジンよりコスト効率に優れた最新のデータベースエンジンに切り替えられます。例えば、AWS DMS を使用すると、Amazon Relational Database Service (Amazon RDS) や Amazon Aurora が提供するマネージドデータベースサービスを活用できます。または、Amazon Redshift が提供するマネージドデータウェアハウスサービス、Amazon DynamoDB などの NoSQL プラットフォーム、または Amazon Simple Storage Service (Amazon S3) などの低コストストレージプラットフォームに移行することもできます。逆に、古いインフラストラクチャから移行しても同じデータベースエンジンを引き続き使用する場合のプロセスも AWS DMS はサポートします。
- AWS DMS は、現在最も人気のある DBMS エンジンのほぼすべてをソースエンドポイントとしてサポートしています。詳細については、「データの移行のソース」を参照してください。
- AWS DMS では、幅広い範囲のターゲットエンジンが利用できます。詳細については、「「データ移行のターゲット」」を参照してください。
- サポート対象の任意のデータソースから、サポート対象の任意のデータターゲットに移行できます。AWS DMS は、サポートされているエンジン間での異種データ移行を完全にサポートしています。
- AWS DMS を使用すると、データ移行の安全性が確保できます。保存中のデータは AWS Key Management Service (AWS KMS) 暗号化を使用して暗号化されます。移行時は、Secure Socket Layer (SSL) を使用して、ソースからターゲットに移動する転送中のデータを暗号化できます。

AWS Database Migration Service の仕組み

AWS Database Migration Service （ AWS DMS) は、ソースデータストアからターゲットデータストアにデータを移行するために使用できるウェブサービスです。この 2 つのデータストアはエンドポイントと呼ばれます。移行は、同じデータベースエンジンを使用するソースエンドポイントとターゲットエンドポイント (Oracle データベースと Oracle データベースなど) の間で行うことができます。移行は、異なるデータベースエンジンを使用するソースエンドポイントとターゲットエンドポイント (Oracle データベースと PostgreSQL データベースなど) の間で行うこともできます。を使用する唯一の要件 AWS DMS は、エンドポイントの 1 つが AWS サービス上にある必要があることです。 AWS DMS を使用してオンプレミスデータベースから別のオンプレミスデータベースに移行することはできません。

データベース移行のコストについては、「AWS Database Migration Service の料金」をご参照ください。

をよりよく理解するには、次のトピックを使用します AWS DMS。

トピック

- の概要 AWS DMS
- のコンポーネント AWS DMS
- のソース AWS DMS
- のターゲット AWS DMS
- の Amazon リソースネーム (ARN) の構築 AWS DMS
- を他の AWS サービス AWS DMS で使用する

の概要 AWS DMS

データベース移行を実行するには、 をソースデータストア AWS DMS に接続し、ソースデータを読み取り、ターゲットデータストアが使用するためにデータをフォーマットします。次に、ターゲットデータストアにデータをロードします。この処理のほとんどはメモリ内で行われますが、大きいトランザクションではディスクへのバッファリングが必要になることがあります。キャッシュされたトランザクションとログファイルもディスクに書き込まれます。

大まかに言うと、 を使用するときは AWS DMS 次の操作を行います。

- ネットワーク環境内で移行の適切な候補となるデータベースを検出する。

- ソースデータベーススキーマとほとんどのデータベースコードオブジェクトを、ターゲットデータベースと互換性のある形式に自動的に変換する。
- レプリケーションサーバーを作成します。
- データストアに関する接続情報を持つソースエンドポイントとターゲットエンドポイントを作成します。
- ソースデータストアとターゲットデータストアの間でデータを移行するには、1 つ以上の移行タスクを作成します。

タスクは、3 つの主なフェーズで構成できます。

- 既存データの移行 (フルロード)
- キャッシュされた変更の適用
- 継続的なレプリケーション (変更データキャプチャ)

フルロード移行中、ソースからの既存のデータがターゲットに移動されるときに、 はソースデータストアのテーブルからターゲットデータストアのテーブルにデータを AWS DMS ロードします。全ロードの進行中、ロードするテーブルに加えられた変更はすべてレプリケーションサーバーにキャッシュされます。これらがキャッシュされた変更点です。は、そのテーブルの全ロードが開始されるまで、特定のテーブルの変更をキャプチャ AWS DMS しないことに注意してください。つまり、変更キャプチャが開始されるポイントは、個々のテーブルごとに異なります。

特定のテーブルの全ロードが完了すると、 AWS DMS はすぐにそのテーブルにキャッシュされた変更を適用し始めます。テーブルがロードされ、キャッシュされた変更が適用されると、 は進行中のレプリケーションフェーズのトランザクションとして変更を収集し AWS DMS 始めます。トランザクションにテーブルがまだ完全にロードされていない場合、変更はレプリケーション インスタンスにローカルに保存されます。 AWS DMS がキャッシュされたすべての変更をすべてのテーブルに適用すると、テーブルはトランザクション整合性が保たれます。この時点で、 は進行中のレプリケーションフェーズ AWS DMS に移行し、変更をトランザクションとして適用します。

継続的なレプリケーションフェーズの開始時、トランザクションのバックログにより、ソースデータベースとターゲットデータベースの間に通常いくらかの遅延が発生します。このトランザクションバックログが終わると、移行は最終的に安定した状態になります。この時点で、アプリケーションをシャットダウンして、残りのトランザクションをターゲットに適用できるようにし、ターゲットデータベースをポイントするようになったアプリケーションを起動できます。

AWS DMS は、データ移行を実行するために必要なターゲットスキーマオブジェクトを作成します。 AWS DMS を使用して最小限のアプローチを取り、データを効率的に移行するために必要なオ

ブジェクトのみを作成できます。このアプローチを使用して、 はテーブル、プライマリキー、および場合によっては一意のインデックス AWS DMS を作成しますが、ソースからデータを効率的に移行するのに必要のない他のオブジェクトは作成しません。

または、 内の DMS Schema Conversion を使用して AWS DMS 、ソースデータベーススキーマとほとんどのデータベースコードオブジェクトをターゲットデータベースと互換性のある形式に自動的に変換することもできます。この変換は、テーブル、ビュー、ストアドプロシージャ、関数、データ型、シノニムなどを対象としています。DMS Schema Conversion が自動的に変換できないオブジェクトには、判別しやすいマークが付けられます。移行を完了するには、このようなオブジェクトは手動で変換します。

のコンポーネント AWS DMS

このセクションでは、 の内部コンポーネント AWS DMS と、それらがデータ移行を達成するためにどのように連携するかについて説明します。 AWS DMS の基礎となるコンポーネントを理解することで、効率的にデータを移行し、問題のトラブルシューティング時および調査時に優れた洞察を得ることができます。

AWS DMS 移行は、移行するデータベースの検出、自動スキーマ変換、レプリケーションインスタンス、ソースエンドポイントとターゲットエンドポイント、レプリケーションタスクの 5 つのコンポーネントで構成されます。 AWS DMS 移行を作成するには、 で必要なレプリケーションインスタンス、エンドポイント、タスクを作成します AWS リージョン。

データベースの検出

DMS Fleet Advisor は、複数のデータベース環境からデータを収集し、データインフラストラクチャに関するインサイトを提供します。DMS Fleet Advisor は、すべてのコンピュータにエージェントをインストールする必要なく、単一または複数の中央ロケーションからオンプレミスのデータベースと分析サーバーのデータを収集します。現時点で DMS Fleet Advisor は Microsoft SQL Server、MySQL、Oracle、PostgreSQL のデータベースサーバーをサポートしています。

ネットワークで検出したデータに基づいて、DMS Fleet Advisor は、ユーザーがモニタリング対象のデータベースサーバーとオブジェクトを決定するために確認できるインベントリを構築します。これらのサーバー、データベース、およびスキーマの詳細が収集されるため、目的とするデータベース移行を実行可能かを分析できます。

スキーマおよびコード移行

の DMS Schema Conversion AWS DMS は、異なるタイプのデータベース間のデータベース移行をより予測しやすくします。DMS Schema Conversion を使用して、ソースデータプロバイダー

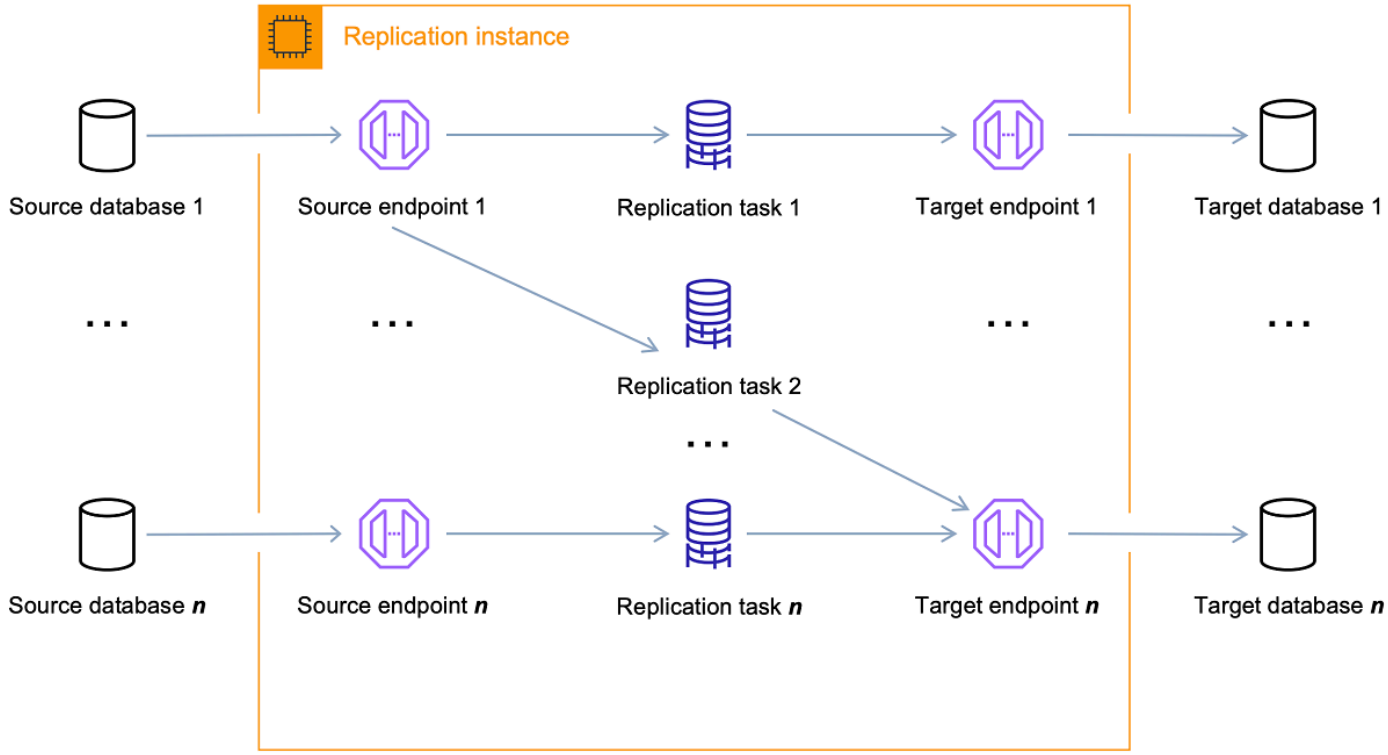
の移行の複雑さの評価、データベーススキーマとコードオブジェクトの変換ができます。その後、変換したコードをターゲットデータベースに適用できます。

DMS Schema Conversion は概括すると、インスタンスプロファイル、データプロバイダー、移行プロジェクトの 3 つのコンポーネントで動作します。[インスタンスプロファイル] では、ネットワークとセキュリティの設定を指定します。[データプロバイダー] には、データベース接続の認証情報を保存します。移行プロジェクトには、データプロバイダー、インスタンスプロファイル、移行ルールが含まれています。 は、データプロバイダーとインスタンスプロファイル AWS DMS を使用して、データベーススキーマとコードオブジェクトを変換するプロセスを設計します。

レプリケーションインスタンス

大まかに言うと、 AWS DMS レプリケーションインスタンスは、1 つ以上のレプリケーションタスクをホストするマネージド Amazon Elastic Compute Cloud (Amazon EC2) インスタンスです。

関連付けられたいくつかのレプリケーションタスクを実行するレプリケーションインスタンスの例を以下の図に示します。



単一のレプリケーションインスタンスは、移行の特性とレプリケーションサーバーの容量に応じて、1 つ以上のレプリケーションタスクをホストできます。 AWS DMS は、ユースケースに最適な設定を選択できるように、さまざまなレプリケーションインスタンスを提供します。レプリ

ケーションインスタンスのさまざまなクラスの詳細については、「移行に適した AWS DMS レプリケーションインスタンスの選択」をご参照ください。

AWS DMS は Amazon EC2 インスタンスにレプリケーションインスタンスを作成します。サービスのテストや小規模な移行の場合、いくつかの小さいインスタンスクラスで十分です。移行に多数のテーブルが関与する場合や、複数の同時レプリケーションタスクを実行する予定の場合、大きいインスタンスを 1 つ使用することを検討してください。 AWS DMS はかなりのメモリと CPU を消費する可能性があるため、このアプローチが推奨されます。

選択したAmazon EC2インスタンスクラスに応じて、レプリケーション インスタンスには 50 GB または 100 GB のデータストレージが付属しています。ほとんどのお客様にとって、通常このストレージ容量は十分な量です。ただし、移行で大規模なトランザクションや大量のデータ変更が発生する場合は、基本ストレージの割り当てを増やすことを検討します。変更データキャプチャ (CDC) によって、データがディスクに書き込まれる可能性があります。これは、ターゲットから変更が書き込まれる速度によって異なります。ログファイルもディスクに書き込まれるため、ログ記録の重要度を上げるとストレージ消費量も増えます。

AWS DMS は、マルチ AZ 配置を使用して、高可用性とフェイルオーバーのサポートを提供します。マルチ AZ 配置では、 はレプリケーションインスタンスのスタンバイレプリカ AWS DMS を別のアベイラビリティーゾーンに自動的にプロビジョニングして維持します。プライマリレプリケーション インスタンスは、同期的にスタンバイレプリカにレプリケートされます。プライマリレプリケーション インスタンスに障害が発生するか、応答しない場合、スタンバイ状態で中断時間をできる限り抑えて、実行中のタスクを再開します。プライマリはその状態を常にスタンバイにレプリケーションしているため、マルチ AZ 配置ではパフォーマンス上のオーバーヘッドが発生します。

AWS DMS レプリケーションインスタンスの詳細については、「」を参照してくださいAWS DMS レプリケーションインスタンスの使用。

レプリケーション インスタンスを作成および管理するのではなく、 AWS DMS サーバーレスを使用してレプリケーションを自動的に AWS DMS プロビジョニングできます。詳細については、「Serverless AWS DMS の使用」を参照してください。

エンドポイント

AWS DMS は、エンドポイントを使用してソースまたはターゲットのデータストアにアクセスします。具体的な接続情報はデータストアによって異なりますが、一般的にエンドポイントを作成するときは次の情報を指定します。

- エンドポイントタイプ - ソースまたはターゲット。

- エンジンタイプ - Oracle、PostgreSQL などのデータベースエンジンの種類
- サーバー名 – が到達 AWS DMS できるサーバー名または IP アドレス。
- ポート - データベースサーバー接続に使用されるポート番号。
- 暗号化 - Secure Sockets Layer (SSL)モード（ SSL を使用して接続を暗号化する場合 ）。
- 認証情報 - 必要なアクセス権限を持つアカウントのユーザー名とパスワード。

AWS DMS コンソールを使用してエンドポイントを作成する場合、コンソールではエンドポイント接続をテストする必要があります。 AWS DMS タスクでエンドポイントを使用する前に、テストが成功している必要があります。接続情報と同様に、特定のテスト基準はエンジンの種類によって異なります。一般的に、 AWS DMS では、指定されたサーバー名とポートにデータベースが存在することと、提供された認証情報を使用して、移行を実行するために必要なアクセス許可を持つデータベースに接続できることを検証します。接続テストが成功すると、 はスキーマ情報 AWS DMS をダウンロードして保存し、後でタスク設定で使用します。スキーマ情報には、テーブル定義、プライマリキー定義、一意キー定義などがあります。

複数のレプリケーションタスクで 1 つのエンドポイントを使用することもできます。たとえば、2 つの論理的に異なるアプリケーションがあり、これらが同じソースデータベースにホストされていて、これらを別々に移行するとします。この場合は、アプリケーションテーブルのセットごとに 1 つずつ、2 つのレプリケーションタスクを作成します。両方のタスクで同じ AWS DMS エンドポイントを使用できます。

エンドポイントの動作は、エンドポイント設定を使用してカスタマイズできます。[エンドポイント設定] を使用して、ログの詳細、ファイルサイズ、その他のパラメータなどのさまざまな動作を管理できます。データストアのエンジンタイプにより、使用できるエンドポイントの設定は異なります。各データストアの具体的なエンドポイントの設定については、データストアのソースセクションまたはターゲットセクションで調べられます。サポートされているソースとターゲットデータストアのリストについては、「のソース AWS DMS」と「のターゲット AWS DMS」をご参照ください。

AWS DMS エンドポイントの詳細については、「」を参照してくださいAWS DMS エンドポイントの使用。

レプリケーションタスク

AWS DMS レプリケーションタスクを使用して、ソースエンドポイントからターゲットエンドポイントに一連のデータを移動します。レプリケーションタスクの作成は、移行を開始する前に最後に実行する必要があります。

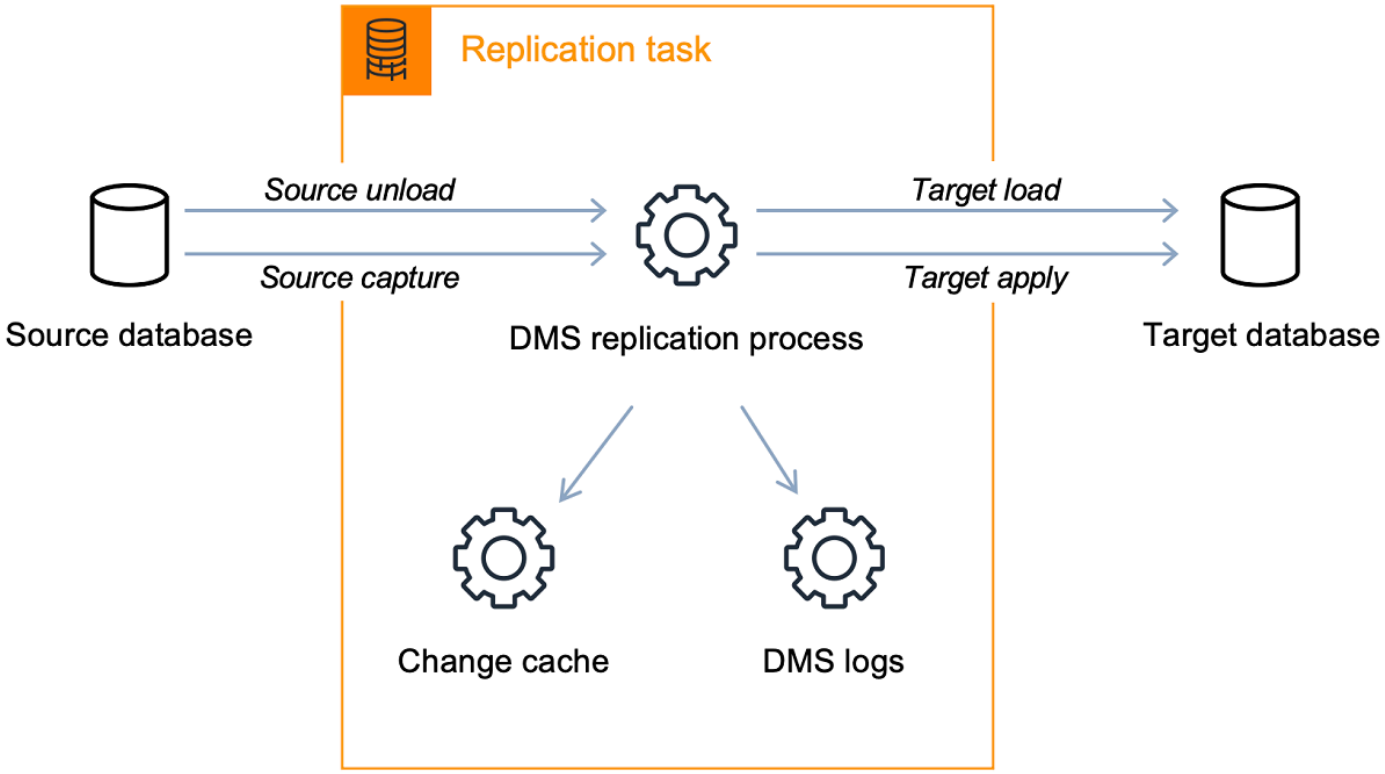
レプリケーションタスクの作成時に以下のタスク設定を指定します。

- レプリケーションインスタンス - タスクをホストして実行するインスタンス
- ソースエンドポイント
- ターゲットエンドポイント
- 移行タイプオプションは次にリストされるとおりです。移行タイプのオプションの詳細な説明については、「[Creating a task] (タスクの作成)」をご参照ください。
  - 全ロード (既存データの移行) - 既存のデータをコピーできる長さの停止が許容される場合は、このオプションが選択に適します。このオプションでは、ソースデータベースからターゲットデータベースにデータがそのまま移行され、必要に応じてテーブルが作成されます。
  - 全ロード + CDC (既存のデータを移行して、継続的な変更をレプリケート) - このオプションでは、ソースで変更をキャプチャしながら、全データロードが実行されます。フルロードが完了すると、キャプチャされた変更がターゲットに適用されます。最終的に、変更の適用は安定した状態に到達します。この時点で、アプリケーションをシャットダウンし、残りの変更がターゲットに移動するようにした後、ターゲットをポイントするアプリケーションを再起動できます。
  - CDC のみ (データ変更のみレプリケート) - 状況によっては、 AWS DMS以外の方法を使用して既存のデータをコピーした方が効率的である場合があります。たとえば、同機種間移行では、一括データのロードにネイティブのエクスポート/インポートツールを使用する方が効率的になる可能性があります。このような状況では、 AWS DMS を使用して、一括ロードを開始したときから変更をレプリケートし、ソースデータベースとターゲットデータベースを同期して維持できます。
- ターゲットテーブル準備モードオプションは以下にリストされるとおりです。ターゲットテーブルモードの詳細な説明については、「[Creating a task] (タスクの作成)」をご参照ください。
  - 何もしない - ターゲットテーブルがターゲットに事前に作成されていることを AWS DMS 前提としています。
  - ターゲットにテーブルを削除 – ターゲットテーブルを AWS DMS ドロップして再作成します。
  - 切り捨て – ターゲットでテーブルを作成済みである場合、 AWS DMS は移行のスタート前にそれらを切り捨てます。テーブルが存在しない場合にこのオプションを選択すると、 は欠落しているテーブル AWS DMS を作成します。
- LOB モードオプションは以下にリストされるとおりです。LOB モードの詳細な説明については、「AWS DMS タスクでのソースデータベースの LOB サポートの設定」をご参照ください。
  - LOB 列を含めない – LOB 列は移行対象から除外されます。

    - フル LOB モード – size に関係なく、完全な LOBs を移行します。 は、最大 LOBs サイズパラメータによって制御されるチャンクで LOB を分割して AWS DMS 移行します。 このモードは制限付き LOB モードを使用するよりも低速です。
    - 制限付き LOB モード – LOB を[Max LOB Size] (最大 LOB サイズ) で指定された値まで切り詰めます。このモードは完全 LOB モードを使用するよりも高速です。
- テーブルマッピング – 移行するテーブルと移行方法を示します。詳細については、「テーブルマッピングを使用して、タスクの設定を指定する」を参照してください。
- データ変換は以下にリストされるとおりです。データ変換の詳細については、「 JSON を使用するテーブル選択および変換を指定する」をご参照ください。
    - スキーマ、テーブル、および列の名前を変更します。
    - テーブルスペース名の変更 (Oracle ターゲットエンドポイント用)。
    - ターゲットのプライマリキーと一意のインデックスを定義します。
- [Data validation] (データ検証)
- Amazon CloudWatch ログ記録

タスクを使用してソースエンドポイントからターゲットエンドポイントにデータを移行します。タスク処理はレプリケーションインスタンスで実行されます。ログ記録要件、制御テーブルデータ、エラー処理など、移行するテーブルとスキーマ、および特別な処理を指定します。

概念的には、 AWS DMS レプリケーションタスクは、次の図に示すように 2 つの異なる機能を実行します。



全ロードプロセスは、簡単に理解できます。データは、ソースから一括抽出で抽出され、ターゲットに直接ロードされます。 AWS DMS コンソールの詳細設定 で、抽出およびロードするテーブルの数を並行して指定できます。

AWS DMS タスクの詳細については、「」を参照してくださいAWS DMS タスクの使用。

継続的なレプリケーションまたは変更データキャプチャ (CDC)

また、 AWS DMS タスクを使用して、データをターゲットに移行している間のソースデータストアへの継続的な変更をキャプチャすることもできます。ソースエンドポイントから進行中の変更をレプリケートするときに が AWS DMS 使用する変更キャプチャプロセスは、データベースエンジンのネイティブ API を使用してデータベースログへの変更を収集します。

CDC プロセスで、レプリケーションタスクは、メモリ内のバッファを使用して転送中のデータを保持することによって、ソースからターゲットへの変更をストリーミングするように設計されています。何らかの理由でメモリ内バッファが枯渇すると、レプリケーションタスクによって、ディスク上のキャッシュの変更は保留中に変更されます。これは、 AWS DMS がソースからの変更をターゲットに適用できるよりも速くキャプチャする場合などに発生する可能性があります。この場合、タスクのターゲットレイテンシーが、タスクのソースレイテンシーを超えていることがわかります。

これを確認するには、 AWS DMS コンソールでタスクに移動し、タスクモニタリングタブを開きます。CDC グラフLatencyTarget と CDCLatencySource グラフはページの下部に表示されます。ターゲットのレイテンシーを示すタスクがある場合は、アプリケーションの速度を上げるために必要なターゲットエンドポイントのチューニングがある可能性があります。

上記で説明したように、レプリケーションタスクではタスクログ用のストレージも使用されます。通常、レプリケーションインスタンスで事前設定されているディスク容量では、十分に変更を口グ記録および反映することができます。追加のディスク容量が必要な場合 (例: 詳細なデバッグを使用して移行の問題を調査する場合)、レプリケーションインスタンスを変更してより多くの容量を割り当てることができます。

のソース AWS DMS

異なる AWS DMS 機能で異なるソースデータストアを使用できます。以下のセクションには、各 AWS DMS 機能でサポートされているソースデータストアのリストが含まれています。

トピック

- データ移行のソースエンドポイント
- DMS Fleet Advisor のソースデータベース
- DMS Schema Conversion のソースデータプロバイダー
- DMS 同種データ移行のソースデータプロバイダー

データ移行のソースエンドポイント

AWS DMSを使用したデータ移行のソースエンドポイントとして、以下のデータストアを使用できます。

オンプレミスおよび EC2 インスタンスデータベース

- Oracle バージョン 10.2 以降 (バージョン 10.x の場合)、11g 以降 12.2 までと 18c、19c (Enterprise、Standard、Standard One、Standard Two エディションの場合)
- Microsoft SQL Server バージョン
  2005、2008、2008R2、2012、2014、2016、2017、2019、2022。
  - Enterprise、Standard、Workgroup、Developer、および Web エディションは、フルロードレプリケーションをサポートしています。

  - Enterprise、Standard (バージョン 2016 以降) 、および Developer エディションは、フルロードに加えて CDC (継続的な) レプリケーションをサポートしています。
  - Express エディションはサポート対象外
- MySQL バージョン 5.5、5.6、5.7、8.0

> **Note**
>
> ソースとしての MySQL 8.0 のサポートは、トランザクションペイロードが圧縮されている場合を除き、 AWS DMS バージョン 3.4.0 以降で利用できます。ソースとしての Google Cloud for MySQL 8.0 のサポートは、 AWS DMS バージョン 3.4.6 以降で利用できます。

- MariaDB (MySQL 互換のデータソースとしてサポート) バージョン 10.0 (バージョン 10.0.24 以降のみ)、10.2、10.3、10.4、10.5、10.6

> **Note**
>
> ソースとしての MariaDB のサポートは、MySQL がサポートされているすべての AWS DMS バージョンで利用できます。

- PostgreSQL バージョン 9.4 以降 (バージョン 9.x の場合) 、10.x、11.x、12.x、13.x 14.x、15.x、および 16.x。

> **Note**
>
> AWS DMS は、バージョン 3.5.1 以降の PostgreSQL バージョン 15.x のみをサポートします。 は、バージョン 3.5.3 以降の PostgreSQL バージョン 16.x AWS DMS のみをサポートします。

- MongoDB バージョン 3.x、4.0、4.2、4.4、5.0、6.0

> **Note**
>
> AWS DMS バージョン 3.5.0 以降は、3.6 より前の MongoDB バージョンをサポートしていません。

- SAP Adaptive Server Enterprise (ASE) バージョン 12.5、15、15.5、15.7、16 以降
- IBM DB 2 for Linux、UNIX、および Windows (Db2 LUW) バージョン:

  - バージョン 9.7、すべてのフィックスパック
  - バージョン 10.1、すべてのフィックスパック
  - バージョン 10.5、フィックスパック 5 を除くすべてのフィックスパック
  - バージョン 11.1、すべてのフィックスパック
  - バージョン 11.5 Mods (1 ~ 8)、フィックスパック 0 のみ
- IBM Db2 for z/OS バージョン 12

サードパーティーのマネージドデータベースサービス:

- Microsoft Azure SQL データベース
- Microsoft Azure PostgreSQL フレキシブルサーバーのバージョン 11.2、12.15、13.11、14.8、15.3
- Microsoft Azure MySQL フレキシブルサーバーのバージョン 5.7、8
- Google Cloud for MySQL のバージョン 5.6、5.7、8.0
- Google Cloud for PostgreSQL のバージョン 9.6、10、11、12、13、14、15
- OCI MySQL Heatwave のバージョン 8.0.34

Amazon RDS インスタンスデータベースおよび Amazon Simple Storage Service (Amazon S3)

- Oracle Enterprise、Standard、Standard One、Standard Two エディションのバージョン 11g (バージョン 11.2.0.4 以降)、12.2、18c、19c
- Microsoft SQL Server の Enterprise、Standard、Workgroup、Developer エディションのバージョン 2012、2014、2016、2017、2019、2022

> **Note**
>
> AWS DMS は SQL Server Express をサポートしていません。Web エディションは、フルロードのみのレプリケーションでのみサポートされます。

- MySQL バージョン 5.5、5.6、5.7、8.0

> **Note**
>
> ソースとしての MySQL 8.0 のサポートは、トランザクションペイロードが圧縮されている場合を除き、 AWS DMS バージョン 3.4.0 以降で利用できます。

- MariaDB (MySQL 互換のデータソースとしてサポート) バージョン 10.0.24 から 10.0.28、10.2、10.3、10.4、10.5、10.6

> **Note**
>
> ソースとしての MariaDB のサポートは、MySQL がサポートされているすべての AWS DMS バージョンで利用できます。

- PostgreSQL バージョン 10.x、11.x、12.x、13.x、14.x、15.x、および 16.x。

> **Note**
>
> AWS DMS は、バージョン 3.5.1 以降の PostgreSQL バージョン 15.x のみをサポートします。 は、バージョン 3.5.3 以降の PostgreSQL バージョン 16.x AWS DMS のみをサポートします。

- MySQL との互換性を持つ Amazon Aurora (MySQL 互換のデータソースとしてサポート)
- PostgreSQL との互換性を持つ Amazon Aurora (PostgreSQL 互換のデータソースとしてサポート)
- Amazon S3
- Amazon DocumentDB (MongoDB 互換) バージョン 3.6、4.0、および 5.0。
- Amazon RDS for IBM Db2 LUW

特定のソースの操作については、AWS DMS 「エンドポイントの使用」を参照してください。

サポート対象のターゲットエンドポイントの詳細については、「データ移行のターゲットエンドポイント」を参照してください。

DMS Fleet Advisor のソースデータベース

DMS Fleet Advisor は、次のソースデータベースをサポートしています。

- Microsoft SQL Server バージョン 2012 から 2019

- MySQL バージョン 5.6 から 8
- Oracle バージョン 11g リリース 2 から 12c、19c、21c
- PostgreSQL バージョン 9.6 から 13

特定のソースの使用方法の詳細については、「AWS DMS Fleet Advisor のデータベースユーザーの作成」を参照してください。

DMS Fleet Advisor がターゲットレコメンデーションの生成に使用するデータベース一覧については、「DMS Fleet Advisor のターゲット」を参照してください。

DMS Schema Conversion のソースデータプロバイダー

DMS Schema Conversion は、移行プロジェクトのソースとして次のデータプロバイダーをサポートしています。

- Microsoft SQL Server バージョン 2008 R2、2012、2014、2016、2017、2019
- Oracle バージョン 10.2 以降、11g から 12.2、18c、19c、Oracle データウェアハウス
- PostgreSQL バージョン 9.2 以降
- MySQL バージョン 5.5 以降

ソースデータプロバイダーとして、オンプレミスまたは Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで実行されているセルフマネージド型エンジンを使用できます。

特定のソースの使用方法の詳細については、「DMS Schema Conversion でのソースデータプロバイダーの作成」を参照してください。

サポート対象のターゲットデータベースの詳細については、「DMS Schema Conversion のターゲットデータプロバイダー」を参照してください。

AWS Schema Conversion Tool（ AWS SCT) は、DMS Schema Conversion よりも多くのソースデータベースとターゲットデータベースをサポートします。が AWS SCT サポートするデータベースの詳細については、「 とは AWS Schema Conversion Tool」を参照してください。

DMS 同種データ移行のソースデータプロバイダー

同種データ移行のソースとして、次のデータプロバイダーを使用できます。

- MySQL バージョン 5.7 以降

- MariaDB バージョン 10.2 以降
- PostgreSQL バージョン 10.4 から 14.x
- MongoDB バージョン 4.x、5.x、6.0
- Amazon DocumentDB バージョン 3.6、4.0、5.0

ソースデータプロバイダーとして、オンプレミスまたは Amazon EC2 インスタンスで実行されているセルフマネージド型エンジンを使用できます。また、Amazon RDS DB インスタンスをソースデータプロバイダーとして使用することもできます。

特定のソースの使用方法の詳細については、「での同種データ移行用のソースデータプロバイダーの作成 AWS DMS」を参照してください。

サポート対象のターゲットデータベースの詳細については、「DMS 同種データ移行のターゲットデータプロバイダー」を参照してください。

のターゲット AWS DMS

異なる AWS DMS 機能で異なるターゲットデータストアを使用できます。以下のセクションには、各 AWS DMS 機能でサポートされているターゲットデータストアのリストが含まれています。

トピック

- データ移行のターゲットエンドポイント
- DMS Fleet Advisor のターゲットデータベース
- DMS Schema Conversion のターゲットデータプロバイダー
- DMS 同種データ移行のターゲットデータプロバイダー

データ移行のターゲットエンドポイント

AWS DMSを使用したデータ移行のターゲットエンドポイントとして、以下のデータストアを使用できます。

オンプレミスおよび Amazon EC2 インスタンスデータベース

- Oracle Enterprise、Standard、Standard One、Standard Two エディションのバージョン 10g、11g、12c、18c、19c

- Microsoft SQL Server の Enterprise。Standard、Workgroup、Developer エディションのバージョン 2005、2008、2008R2、2012、2014、2016、2017、2019、2022

> **Note**
>
> AWS DMS は SQL Server Web および Express エディションをサポートしていません。

- MySQL バージョン 5.5、5.6、5.7、8.0
- MariaDB (MySQL 互換データターゲットとしてサポート) バージョン 10.0.24 から 10.0.28、10.1、10.2、10.3、10.4。

> **Note**
>
> ターゲットとしての MariaDB のサポートは、MySQL がサポートされているすべての AWS DMS バージョンで利用できます。

- PostgreSQL バージョン 9.4 以降 (バージョン 9.x の場合) 、10.x、11.x、12.x、13.x、14.x、15.x、および 16.x。

> **Note**
>
> AWS DMS は、バージョン 3.5.1 以降の PostgreSQL 15.x のみをサポートします。 は、バージョン 3.5.3 以降の PostgreSQL バージョン 16.x AWS DMS のみをサポートします。

- SAP Adaptive Server Enterprise (ASE) バージョン 15、15.5、15.7、16 以降
- Redis バージョン 6.x

Amazon RDS インスタンスデータベース、Amazon Redshift、Amazon Redshift Serverless、Amazon DynamoDBAmazon S3、Amazon OpenSearch Service、Amazon ElastiCache for Redis、Amazon Kinesis Data Streams、Amazon DocumentDB、Amazon Neptune、および Apache Kafka

- Oracle Enterprise、Standard、Standard One、Standard Two エディションのバージョン 11g (バージョン 11.2.0.3.v1 以降)、12c、18c、19c
- Microsoft SQL Server の Enterprise、Standard、Workgroup、Developer エディションのバージョン 2012、2014、2016、2017、2019、2022

> **Note**
>
> AWS DMS は SQL Server Web および Express エディションをサポートしていません。

- MySQL バージョン 5.5、5.6、5.7、8.0
- MariaDB (MySQL 互換データターゲットとしてサポート) バージョン 10.0.24 から 10.0.28、10.1、10.2、10.3、10.4。

> **Note**
>
> ターゲットとしての MariaDB のサポートは、MySQL がサポートされているすべての AWS DMS バージョンで利用できます。

- PostgreSQL バージョン 10.x、11.x、12.x、13.x、14.x、15.x、および 16.x。

> **Note**
>
> AWS DMS は、バージョン 3.5.1 以降の PostgreSQL 15.x のみをサポートします。 は、バージョン 3.5.3 以降の PostgreSQL 16.x AWS DMS のみをサポートします。

- IBM Db2 LUW バージョン 11.1 と 11.5
- Amazon Aurora MySQL 互換エディション
- Amazon Aurora PostgreSQL 互換エディション
- Amazon Aurora Serverless v2
- Amazon Redshift
- Amazon Redshift Serverless
- Amazon S3
- Amazon DynamoDB
- Amazon OpenSearch サービス
- Amazon ElastiCache for Redis
- Amazon Kinesis Data Streams
- Amazon DocumentDB (MongoDB 互換性)
- Amazon Neptune

- Apache Kafka — Amazon Managed Streaming for Apache Kafka (Amazon MSK) と セルフマネージド Apache Kafka
- Babelfish (バージョン以降) for Aurora PostgreSQL (バージョン 15.3/14.8 以降)

特定のターゲットの操作については、AWS DMS 「エンドポイントの使用」を参照してください。

サポート対象のソースエンドポイントの詳細については、「データ移行のソースエンドポイント」を参照してください。

DMS Fleet Advisor のターゲットデータベース

DMS Fleet Advisor はターゲットレコメンデーションの生成に、次のターゲットデータベースの最新バージョンを使用します。

- Amazon Aurora MySQL
- Amazon Aurora PostgreSQL
- Amazon RDS for MySQL
- Amazon RDS for Oracle
- Amazon RDS for PostgreSQL
- Amazon RDS for SQL Server

DMS Fleet Advisor のターゲットレコメンデーションの詳細については、「AWS DMS Fleet Advisor ターゲットレコメンデーション機能の使用」を参照してください。

サポート対象のソースデータベースの詳細については、「DMS Fleet Advisor のソースデータベース」を参照してください。

DMS Schema Conversion のターゲットデータプロバイダー

DMS Schema Conversion は、移行プロジェクトのターゲットとして次のデータプロバイダーをサポートしています。

- Amazon Aurora MySQL 8.0.23
- Amazon Aurora PostgreSQL 14.5
- Amazon RDS for MySQL 8.0.23

- Amazon RDS for PostgreSQL 14.x
- Amazon Redshift

特定のターゲットの使用方法の詳細については、「DMS Schema Conversion でのターゲットデータプロバイダーの作成」を参照してください。

サポート対象のソースデータベースの詳細については、「DMS Schema Conversion のソースデータプロバイダー」を参照してください。

DMS 同種データ移行のターゲットデータプロバイダー

同種データ移行のターゲットとして、次のデータプロバイダーを使用できます。

- Amazon Aurora MySQL バージョン 5.7 以降
- Amazon Aurora PostgreSQL バージョン 10.4 ~ 14.x
- Amazon Aurora Serverless v2
- Amazon RDS for MySQL バージョン 5.7 以降
- Amazon RDS for MariaDB バージョン 10.2 以降
- Amazon RDS for PostgreSQL バージョン 10.4 ~ 14.x
- Amazon DocumentDB バージョン 4.0、5.0、および DocumentDB Elastic クラスター

特定のターゲットの使用方法の詳細については、「での同種データ移行のターゲットデータプロバイダーの作成 AWS DMS」を参照してください。

サポート対象のソースデータベースの詳細については、「DMS 同種データ移行のソースデータプロバイダー」を参照してください。

の Amazon リソースネーム (ARN) の構築 AWS DMS

AWS CLI または AWS DMS API を使用してデータベースの移行を自動化する場合は、Amazon リソースネーム (ARNsを使用します。Amazon Web Servicesで作成される各リソースは、一意の識別子である ARN によって識別されます。 AWS CLI または AWS DMS API を使用してデータベース移行を設定する場合は、使用するリソースの ARN を指定します。

AWS DMS リソースの ARN は、次の構文を使用します。

```
arn:aws:dms:region:account number:resourcetype:resourcename
```

この構文では、次の条件が適用されます。

- *region* は、 など、 AWS DMS リソースが作成された の AWS リージョン ID ですus-west-2。

  次の表は、ARN を構築するときに使用する AWS リージョン 名前と値を示しています。

リージョン	名前
アジアパシフィック (東京) リージョン	ap-northeast-1
アジアパシフィック (ソウル) リージョン	ap-northeast-2
アジアパシフィック (ムンバイ) リージョン	ap-south-1
アジアパシフィック (シンガポール) リージョン	ap-southeast-1
アジアパシフィック (シドニー) リージョン	ap-southeast-2
カナダ (中部) リージョン	ca-central-1
中国 (北京) リージョン	cn-north-1
中国 (寧夏) リージョン	cn-northwest-1
欧州 (ストックホルム) リージョン	eu-north-1
欧州 (ミラノ) リージョン	eu-south-1
欧州 (フランクフルト) リージョン	eu-central-1
欧州 (アイルランド) リージョン	eu-west-1
欧州 (ロンドン) リージョン	eu-west-2
欧州 (パリ) リージョン	eu-west-3
南米 (サンパウロ) リージョン	sa-east-1
米国東部 (バージニア州北部) リージョン	us-east-1

リージョン	名前
米国東部(オハイオ州)リージョン	us-east-2
US West (N. California) リージョン	us-west-1
米国西部 (オレゴン州) リージョン	us-west-2

- *account number* はダッシュ記号が省略されたアカウント番号です。アカウント番号を確認するには、http://aws.amazon.com, で AWS アカウントにサインインし、マイアカウント/コンソールを選択し、マイアカウント を選択します。
- *resourcetype* は AWS DMS リソースのタイプです。

  次の表は、特定のリソースの ARN を構築するときに使用する AWS DMS リソースタイプを示しています。

AWS DMS リソースタイプ	ARN 形式
レプリケーションインスタンス	`arn:aws:dms:` *region*`:` *account*`:rep:` *resourcename*
エンドポイント	`arn:aws:dms:` *region*`:`*account*`:endpoint:` *resourcename*
レプリケーションタスク	`arn:aws:dms:` *region*`:`*account*`:task:`*resourcename*
サブネットグループ	`arn:aws:dms:` *region*`:`*account*`:subgrp:`*resourcename*

- *resourcename* は、リソースに割り当てられた AWS DMS リソース名です。これは生成された任意の文字列です。

次の表は、 AWS DMS リソースの ARNs の例を示しています。ここでは、 AWS アカウントが 123456789012、米国東部 (バージニア北部) リージョンに作成され、リソース名を持つと仮定します。

リソースタイプ	サンプル ARN
レプリケーションインスタンス	arn:aws:dms:us-east-1:123456789012:rep:QLXQZ64MH7CXF4QCQMGRVYVXAI
エンドポイント	arn:aws:dms:us-east-1:123456789012:endpoint:D3HMZ2IGUCGFF3NTAXUXGF6S5A
レプリケーションタスク	arn:aws:dms:us-east-1:123456789012:task:2PVREMWNPGYJCVU2IBPTOYTIV4
サブネットグループ	arn:aws:dms:us-east-1:123456789012:subgrp:test-tag-grp

を他の AWS サービス AWS DMS で使用する

は、他のいくつかの AWS サービス AWS DMS で使用できます。

- Amazon EC2 インスタンスまたは Amazon RDS DB インスタンスをデータ移行のターゲットとして使用できます。
- AWS Schema Conversion Tool（ AWS SCT) を使用して、ソーススキーマと SQL コードを同等のターゲットスキーマと SQL コードに変換できます。
- Amazon S3 をデータのストレージサイトとして使用することも、大量のデータを移行する際の中間ステップとして使用することもできます。
- を使用して AWS CloudFormation 、インフラストラクチャの管理またはデプロイ用に AWS リソースを設定できます。例えば、レプリケーションインスタンス、タスク、証明書、エンドポイントなどの AWS DMS リソースをプロビジョニングできます。必要なすべての AWS リソースを記述するテンプレートを作成し、それらのリソースを AWS CloudFormation プロビジョニングして設定します。

AWS DMS のサポート AWS CloudFormation

を使用して AWS DMS リソースをプロビジョニングできます AWS CloudFormation。 AWS CloudFormation は、インフラストラクチャの管理またはデプロイのために AWS リソースをモデル化およびセットアップするのに役立つサービスです。例えば、レプリケーションインスタンス、タスク、証明書、エンドポイントなどの AWS DMS リソースをプロビジョニングできます。必要なす

べての AWS リソースを記述するテンプレートを作成し、それらのリソースを AWS CloudFormation プロビジョニングして設定します。

開発者またはシステム管理者は、これらのリソースのコレクションを作成および管理し、繰り返し行われる移行タスクや、組織へのリソースのデプロイに使用できます。の詳細については AWS CloudFormation、「 ユーザーガイド」の「 のAWS CloudFormation 概念AWS CloudFormation 」を参照してください。

AWS DMS では、 を使用した次の AWS DMS リソースの作成がサポートされています AWS CloudFormation。

- AWS::DMS::Certificate
- AWS::DMS::Endpoint
- AWS::DMS::EventSubscription
- AWS::DMS::ReplicationInstance
- AWS::DMS::ReplicationSubnetグループ
- AWS::DMS::ReplicationTask

AWS Database Migration Service の開始方法

このチュートリアルでは、AWS Database Migration Service (AWS DMS)を使用してデータベース移行を実行する方法について説明します。

データベース移行を実行するには、次の手順を実行します。

1. 「のセットアップ AWS Database Migration Service」の手順に従って AWS アカウントをセットアップします。
2. サンプルデータベースと Amazon EC2 クライアントを作成して、ソースデータベースにデータを入力し、レプリケーションをテストします。また、Amazon Virtual Private Cloud (Amazon VPC) サービスを基盤とする仮想プライベートクラウド (VPC) を作成して、チュートリアルのリソースコンテナとします。上記のリソースを作成するには、「の前提条件 AWS Database Migration Service」の手順に従ってください。
3. データベース作成スクリプトのサンプル を使用してソースデータベースにデータを入力します。
4. DMS Schema Conversion または AWS Schema Conversion Tool (AWS SCT) を使用して、ソースデータベースからターゲットデータベースにスキーマを変換します。DMS Schema Conversion の使用方法については、「DMS Schema Conversion の開始方法」の手順に従ってください。AWS SCT を使用してスキーマを変換するには、「スキーマを移行する」の手順に従ってください。
5. 移行のすべてのプロセスを実行するレプリケーションインスタンスを作成します。このタスクとその後のタスクを実行するには、「レプリケーション」の手順に従ってください。
6. ソースデータベースとターゲットデータベースエンドポイントを指定します。エンドポイントの作成については、「ソースおよびターゲットエンドポイントの作成」を参照してください。
7. 使用するテーブルとレプリケーションプロセスを定義するタスクを作成して、レプリケーションを開始します。データベース移行タスクの作成については、「[Creating a task] (タスクの作成)」を参照してください。
8. ターゲットデータベースでクエリを実行して、レプリケーションが機能していることを検証します。

のセットアップ AWS Database Migration Service

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

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。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。https://aws.amazon.com/ の [マイアカウント] を選んで、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理できます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、 を保護し AWS アカウントのルートユーザー、 を有効にして AWS IAM Identity Center、日常的なタスクにルートユーザーを使用しないように管理ユーザーを作成します。

のセキュリティ保護 AWS アカウントのルートユーザー

1. ルートユーザーを選択し、 AWS アカウント E メールアドレスを入力して、アカウント所有者AWS Management Consoleとして にサインインします。次のページでパスワードを入力します。

   ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「ルートユーザーとしてサインインする」を参照してください。
2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、「IAM ユーザーガイド」の AWS アカウント 「ルートユーザーの仮想 MFA デバイスを有効にする (コンソール)」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

手順については、「AWS IAM Identity Center ユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。

2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

   を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、「 ユーザーガイド」の「デフォルト でユーザーアクセス IAM アイデンティティセンターディレクトリを設定するAWS IAM Identity Center 」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

  IAM Identity Center ユーザーを使用してサインインする方法については、「 AWS サインインユーザーガイド」の AWS 「 アクセスポータルへのサインイン」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「グループの参加」を参照してください。

の前提条件 AWS Database Migration Service

このセクションでは、ソースデータベースとターゲットデータベースの設定など AWS DMS、 の前提条件タスクについて説明します。これらのタスクの一環として、Amazon VPC サービスに基づいて Virtual Private Cloud (VPC) も設定します。さらに、ソースデータベースの入力とターゲットデータベースでのレプリケーションの検証に使用する Amazon EC2 インスタンスをセットアップします。

> **Note**
>
> ソースデータベースへの入力には最大 45 分かかります。

このチュートリアルでは、ソースとして MariaDB データベースを作成し、ターゲットとして PostgreSQL データベースを作成します。このシナリオでは、一般的に使用される低コストのデータベースエンジンを使用して、レプリケーションを実証します。異なるデータベースエンジンを使用すると、異種プラットフォーム間でデータを移行する AWS DMS 機能を示しています。

このチュートリアルのリソースでは、米国西部 (オレゴン) リージョンを使用します。別の AWS リージョンを使用する場合は、米国西部 (オレゴン) が表示される場所ではなく、選択したリージョンを指定します。

> **Note**
>
> 簡単にするために、このチュートリアル用に作成するデータベースでは、暗号化やその他のアドバンストセキュリティ機能を使用しません。本番データベースを安全に保つには、セキュリティ機能を使用する必要があります。詳細については「Amazon RDS でのセキュリティ」をご参照ください。

前提条件のステップについては、以下のトピックをご参照ください。

トピック

- 「VPC を作成する」
- Amazon RDS パラメータグループの作成
- ソース Amazon RDS データベースを作成する
- ターゲットの Amazon RDS データベースを作成します。
- Amazon EC2 クライアントを作成する
- ソースデータベースに挿入する

「VPC を作成する」

このセクションでは、 AWS リソースを含む VPC を作成します。VPC を使用することは、データベース、Amazon EC2 インスタンス、セキュリティグループなどが論理的に整理され、安全になるように、 AWS リソースを使用する際のベストプラクティスです。

チュートリアルリソースに VPC を使用すると、チュートリアルの終了時に使用するすべてのリソースも削除するようにしてください。VPC に含まれるすべてのリソースを削除してからVPC を削除できます。

で使用する VPC を作成するには AWS DMS

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/vpc/ で Amazon VPC コンソールを開きます。
2. ナビゲーションペインで、[VPC ダッシュボード] を選択して、[VPC の作成] をクリックします。
3. [VPC の作成] ページで、次のオプションを入力します。

   - 作成するリソース: VPC など
   - 名前タグの自動生成: [自動生成] を選択して、**DMSVPC** と入力する
   - IPv4 ブロック: **10.0.1.0/24**
   - IPv6 CIDR ブロック: IPv6 CIDR ブロックなし
   - テナンシー: デフォルト
   - アベイラビリティーゾーンの数: 2
   - パブリックサブネットの数: 2
   - プライベートサブネットの数: 2
   - NAT ゲートウェイ ($): なし
   - VPC エンドポイント: なし

   [Create VPC] (VPC作成) を選択します。
4. ナビゲーションペインで VPC を選択します。DMSVPC の VPC ID を書き留めます。
5. ナビゲーションペインで[Security Groups] (セキュリティグループ) を選択します。
6. DMSVPCについて書き留めた ID と一致するVPC IDの[default] (デフォルト) という名前のグループを選択します。
7. [Inbound rules] (インバウンドルール) タブを選択し、[Edit inbound rules] (インバウンドルールの編集) を選択します。
8. [Add rule] (ルール追加) を選択します。[MySQL/Aurora] のルールタイプを追加して、[ソース] には Anywhere-IPv4 を選択します。
9. [Add rule] (ルールの追加) を選択しなおします。[PostgreSQL] のルールタイプを追加して、[ソース] には Anywhere-IPv4 を選択します。

10. [Save Rules] (ルールの保存) を選択します。

Amazon RDS パラメータグループの作成

のソースデータベースとターゲットデータベースの設定を指定するには AWS DMS、Amazon RDS パラメータグループを使用します。データベース間の初期および継続的なレプリケーションを許可するには、次の構成を必ず行ってください：

- ソースデータベースのバイナリログ。これにより、 AWS DMS はレプリケートする必要がある増分更新を決定できます。
- ターゲットデータベースのレプリケーションロール。これにより、 は最初のデータ転送中に外部キーの制約 AWS DMS を無視します。この設定では、 はデータを順序どおりに移行 AWS DMS できます。

で使用するパラメータグループを作成するには AWS DMS

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで [Parameter groups] (パラメータグループ) を選択します。
3. [Parameter groups] (パラメータグループ) ページで[Create parameter group] (パラメータグループの作成) を選択します。
4. [Create parameter group] (パラメータグループ作成) ページで、次の設定を入力します：

   - パラメータグループファミリー: mariadb10.6
   - [Group name] (グループ名): **`dms-mariadb-parameters`**
   - [Description] (説明): **`Group for specifying binary log settings for replication`**

   [Create] (作成) を選択します。
5. [パラメータグループ] ページで、[dms-mariadb-parameters] を選択して、[dms-mariadb-parameters] ページで、[編集] をクリックします。
6. 次のパラメータを次の値に設定します：

   - binlog_checksum:[NONE] (無し)
   - binlog_format:[ROW] (行)

[Save changes] (変更を保存) を選択します。

7. [Parameter groups] (パラメータグループ) ページで[Create parameter group] (パラメータグループの作成) を選択します。
8. [Create parameter group] (パラメータグループ作成) ページで、次の設定を入力します：

   - パラメータグループファミリー: postgres13
   - [Group name] (グループ名): **dms-postgresql-parameters**
   - [Description] (説明): **Group for specifying role setting for replication**

   [Create] (作成) を選択します。
9. [Parameter groups] (パラメータグループ) ページで dms-postgresql-parameters を選択します。
10. [dms-postgresql-parameters] ページで、[編集] をクリックして、[session_replication_role parameter] を [レプリカ] と設定します。session_replication_role のパラメータが、パラメータの最初のページにないことに注意します。このパラメータを見つけるには、ページ割りコントロールまたは検索フィールドを使用します。
11. [変更の保存] を選択します。

ソース Amazon RDS データベースを作成する

次の手順に従って、ソース Amazon RDS データベースを作成します。

ソースの Amazon RDS for MariaDB データベースを作成するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. [Dashboard] (ダッシュボード) ページで[Database] (データベース) セクションの[Create Database] (データベースの作成) を選択します。ページ上部の [MySQL と PostgreSQL 用の新しい Amazon RDS マルチ AZ 配置オプションを試す] セクションにある [データベースの作成] はクリックしないでください。
3. [Create Database] (データベースの作成) ページで、次の詳細を設定します：

   - データベース作成方法を選択: [標準作成] を選択する。
   - エンジンのオプション: [エンジンタイプ] では、[MariaDB] を選択する。[バージョン] は、[MariaDB 10.6.14] が選択されたままにする。
   - [Templates] (テンプレート): [Dev/Test] (開発/テスト) を選択します。

- [Settings] (設定) :
  - [DB instance identifier] (DB インスタンス識別子) : **dms-mariadb** を入力します。
  - [認証情報の設定] セクションで、次を入力する。
    - [Master username] (マスター ユーザーネーム): **admin** のままにしておきます。
    - AWS Secrets Manager のマスター認証情報の管理はチェックしないでください。
    - [Auto generate a password] (パスワードの自動生成): 選択を解除したままにします。
    - [Master password] (マスターパスワード): **changeit** と入力します。
    - [Confirm password] (パスワードを確認): 再び **changeit** と入力します。
- インスタンスの設定:
  - [DB instance class] (DB インスタンスクラス): [Standard classes] (標準クラス) を選んだままにします。
  - [DB instance class] (DB インスタンスクラス) でdb.m5.largeを選択します。
- [Storage] (ストレージ) :
  - [Enable storage autoscaling] (ストレージのオートスケーリングの有効化) ボックスをクリアします。
  - 残りの設定はそのままにします。
- アベイラビリティーおよび耐久性: [スタンバイインスタンスを作成しない] が選択されたままにする。
- 接続:
  - コンピューティングリソース [EC2 コンピューティングリソースに接続しない] のままにする。
  - ネットワークタイプ: [IPv4] が選択されたままにする。
  - 仮想プライベートクラウド: DMSVPC-vpc
  - [Public access] (公開アクセス) : [Yes] (はい)。 AWS Schema Conversion Toolを使用するには、公開アクセスを有効にする必要があります。
  - [Availability zone] (アベイラビリティーゾーン) : us-west-2a
  - 残りの設定はそのままにします。
- データベース認証: [パスワード認証] が選択されたままにする。
- [モニタリング] の [Performance Insights を有効にする] のチェックボックスはオフにする。[その他の設定] セクションを展開して、[拡張モニタリングの有効化] チェックボックスをオフにする。

- [Additional configuration] (追加設定) を展開します：
  - [Database options] (データベースオプション) で[Initial database name] (初期データベース名) として **dms_sample** と入力します。
  - [DB パラメータグループ] では [dms-mariadb-parameters] を選択する。
  - [オプショングループ] では、[default:mariadb-10-6] が選択されたままにする。
  - 「バックアップ」 で、次の作業を行います。
    - [自動バックアップの有効化] はオンのままにする。継続的なレプリケーションをサポートするには、ソースデータベースで自動バックアップが有効になっている必要があります。
    - [バックアップ保持期間] では、[1 日] を選択する。
    - [バックアップウィンドウ] では、[指定なし] が選択されたままにする。
    - [スナップショットにタグをコピー] チェックボックスをオフにする。
    - 別の AWS リージョンでレプリケーションを有効にする はチェックしないでください。
  - [Encryption] (暗号化) で[Enable encryption] (暗号化有効) ボックスをクリアします。
  - [ログのエクスポート] セクションはそのままにする。
  - [メンテナンス] の [マイナーバージョン自動アップグレードの有効化] チェックボックスをオフにして、[メンテナンスウィンドウ] 設定を [指定なし] のままにする。
  - [削除保護の有効化] をオフのままにする。
4. [Create database] (データベースの作成) を選択します。

ターゲットの Amazon RDS データベースを作成します。

上記の手順を繰り返して、以下の変更を加えターゲットの Amazon RDS データベースを作成します。

PostgreSQL データベース用のターゲット RDS を作成するには

1. 前述の手順のステップ 1 と 2 を繰り返します。
2. [Create database] (データベースの作成) ページで、次の場合を除き、同じオプションを設定します：

   a. [Engine options] (エンジンオプション) で PostgreSQL を選択します。

   b. [バージョン] では [PostgreSQL 13.7-R1] を選択する

   c. [DB instance identifier] (DB インスタンス識別子) に**dms-postgresql** と入力します。

d. [管理ユーザー名] では **postgres** を選択したままにする

e. [DB parameter group] (DB パラメータグループ) で、dms-postgresql-parameters を選択します。

f. [Enable automatic backups] (自動バックアップ有効) をクリアします。

3. [Create database] (データベースの作成) を選択します。

Amazon EC2 クライアントを作成する

このセクションでは、Amazon EC2を作成します。このクライアントを使用して、レプリケートするデータをソースデータベースに入力します。また、このクライアントを使用して、ターゲットデータベースでクエリを実行してレプリケーションを検証します。

Amazon EC2 クライアントを使用してデータベースにアクセスすると、インターネットからデータベースにアクセスするよりも、次のような利点があります：

- データベースへのアクセスを同じ VPC 内のクライアントに制限できます。
- このチュートリアルで使用するツールは、このチュートリアルで推奨される Amazon Linux 2023 で動作し、簡単にインストールできることが確認されている。
- VPC 内のコンポーネント間のデータオペレーションは、通常、インターネット経由のコンポーネントよりも優れたパフォーマンスを発揮します。

Amazon EC2 クライアントを作成して構成し、ソースデータベースに取り込むには

1. Amazon EC2 コンソール https://console.aws.amazon.com/ec2/を開きます。
2. [Dashboard] (ダッシュボード) で、[Launch instance] (インスタンスの起動) を選択します。
3. [インスタンスを起動] ページで、次の値を入力します。

   a. [名前とタグ] セクションの [名前] には、**DMSClient** と入力する。

   b. アプリケーションと OS のイメージ (Amazon マシンイメージ) セクションの設定はそのままにする。

   c. [インスタンスタイプ] セクションで [t2.xlarge] を選択する。

   d. [キーペア (ログイン)] セクションで、[新しいキーペアを作成する] を選択する。

   e. [キーペアの作成] ページで、次のとおり設定する。

      - キーペア名: **DMSKeyPair**

- キーペアタイプ: RSA のままにする
- プライベートキーファイル形式: MacOS または Linux の OpenSSH の場合は [pem]、Windows の PuTTY の場合は [ppk] を選択する。

プロンプトが表示されたら、キーファイルを保存します。

> **Note**
>
> 新しいキーペアを作成するのではなく、既存の Amazon EC2 キーペアを使用することもできます。

f. [ネットワーク設定] セクションで、[編集] をクリックする。以下の設定を選択します。

- VPC - 必須: DMSVPC-vpc VPC のために記録しておいた ID を持つ VPC を選択する。
- [サブネット]: 最初のパブリックサブネットを選択する。
- パブリック IP の自動割り当て: [有効] を選択する。

残りの設定はそのままにして、[インスタンスを起動] をクリックします。

ソースデータベースに挿入する

このセクションでは、後で使用するためのソースデータベースとターゲットデータベースのエンドポイントを検索し、次のツールを使用してソースデータベースに挿入します。

- Git、ソースデータベースに入力させるスクリプトをダウンロードします。
- MariaDB クライアント。このスクリプトを実行するクライアントです。

エンドポイントを取得する

RDS for MariaDB および RDS for PostgreSQL DB インスタンスのエンドポイントを検索してメモし、後で使用します。

DB インスタンスエンドポイントの検索

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

2. ナビゲーションペインで、[Databases] (データベース) を選択します。
3. [dms-mariadb] データベースを選択して、データベースの [エンドポイント] の値をメモしておきます。
4. 前のステップをdms-postgresql データベースに対して繰り返します。

ソースデータベースに挿入する

次に、クライアントインスタンスに接続し、必要なソフトウェアをインストールし、Git から AWS サンプルデータベーススクリプトをダウンロードして、スクリプトを実行してソースデータベースにデータを入力します。

ソースデータベースに挿入するには

1. 前の手順で保存したホスト名と公開キーを使用して、クライアントインスタンスに接続します。

   Amazon EC2 インスタンスへの接続の詳細については、Amazon EC2 ユーザーガイド」の「インスタンスへのアクセス」を参照してください。

   > **Note**
   >
   > PuTTY を使用している場合は、[Connection] (接続) 設定で TCP キープアライブを有効にし、接続が非アクティブ状態からタイムアウトしないようにします。

2. Git、MariaDB、PostgreSQL をインストールします。必要に応じてインストールを確認します。

   ```
   $ sudo yum install git
   $ sudo dnf install mariadb105
   $ sudo dnf install postgresql15
   ```

3. 次のコマンドを実行して、 からデータベース作成スクリプトをダウンロードします GitHub。

   ```
   git clone https://github.com/aws-samples/aws-database-migration-samples.git
   ```

4. `aws-database-migration-samples/mysql/sampledb/v1/` ディレクトリを変更します。
5. 以下のコマンドを実行します。以前にメモしたソース RDS インスタンスのエンドポイントを指定します。たとえば、`dms-mariadb.cdv5fbeyiy4e.us-east-1.rds.amazonaws.com`。

```
mysql -h dms-mariadb.abcdefghij01.us-east-1.rds.amazonaws.com -P 3306 -u admin -p
 dms_sample < ~/aws-database-migration-samples/mysql/sampledb/v1/install-rds.sql
```

6. データベース作成スクリプトを実行します。スクリプトは、スキーマを作成してデータを代入するのに最大 45 分かかります。スクリプトが表示するエラーや警告は無視して構いません。

AWS SCT を使用したソーススキーマのターゲットデータベースへの移行

このセクションでは、AWS Schema Conversion Tool を使用して、ソーススキーマをターゲットデータベースに移行します。または、DMS Schema Conversion を使用してソースデータベーススキーマを変換することもできます。詳細については、「DMS Schema Conversion の開始方法」を参照してください。

AWS SCT を使用してソーススキーマをターゲットデータベースに移行するには

1. AWS Schema Conversion Tool をインストールします。詳細については、「AWS Schema Conversion Tool ユーザーガイド」の「AWS SCT のインストール、確認、更新」を参照してください。

   MySQL 用と PostgreSQL 用の JDBC ドライバーをダウンロードする際は、ツールに場所の入力を求められる場合に備えて、ドライバーの保存場所をメモしておきます。

2. AWS Schema Conversion Tool を開きます。[ファイル] をクリックして、[新しいプロジェクト] を選択します。
3. [新しいプロジェクト] ウィンドウで、次の値を設定します。

   - [プロジェクト名] は **DMSProject** と設定する。
   - AWS SCT プロジェクトをデフォルトのフォルダに保存する [場所] はそのままにする。

   [OK] をクリックします。
4. [ソースを追加] をクリックし、ソースの MySQL データベースをプロジェクトに追加して、[MySQL] を選択し、[次へ] をクリックします。
5. [ソースを追加] ページで、次の値を設定します。

   - 接続名: **source**

- サーバー名: 前にメモしておいた MySQL データベースのエンドポイントを入力する。
- サーバーポート: **3306**
- ユーザー名: **admin**
- パスワード: **changeit**

6. [ターゲットの追加] をクリックして、ターゲットの Amazon RDS for PostgreSQL データベースをプロジェクトに追加して、[Amazon RDS for PostgreSQL] を選択します。[次へ] をクリックします。
7. [ターゲットの追加] ページで、次の値を設定します。

   - 接続名: **target**
   - サーバー名: 前にメモしておいた PostgreSQL データベースのエンドポイントを入力する。
   - サーバーポート: **5432**
   - データベース: 作業中の PostgreSQL データベース名を入力する。
   - ユーザー名: **postgres**
   - パスワード: **changeit**

8. 左側のペインで、[スキーマ] の下にある [dms_sample] を選択します。右側のペインで、ターゲットの Amazon RDS for PostgreSQL データベースを選択します。[マッピングを作成] をクリックします。単一の AWS SCT プロジェクト内に複数のマッピングルールを追加できます。マッピングルールの詳細については、「Creating mapping rules」を参照してください。
9. [メインビュー] をクリックします。
10. 左側のペインで、[スキーマ] の下にある [dms_sample] を選択します。コンテキスト (右クリック) メニューを開いて、[スキーマを変換] を選択します。アクションを確定します。

    ツールがスキーマを変換したら、右側のペインに [dms_sample] スキーマが表示されます。
11. 右側のペインの [スキーマ] の下で、[dms_sample] のコンテキストメニュー (右クリック) を開き、[Apply to database] を選択します。アクションを確定します。

スキーマの移行が完了していることを確認します。次のステップを実行します。

スキーマの移行を確認するには

1. Amazon EC2 クライアントに接続します。
2. 次のコマンドを使用して、PSQL クライアントを起動します。PostgreSQL データベースエンドポイントを指定して、プロンプトが表示されたらデータベースのパスワードを入力します。

```
psql \
   --host=dms-postgresql.abcdefg12345.us-west-2.rds.amazonaws.com \
   --port=5432 \
   --username=postgres \
   --password \
   --dbname=dms_sample
```

3. (空の) テーブルのいずれかをクエリして、AWS SCT がスキーマを正しく適用したことを検証します。

```
dms_sample=> SELECT * from dms_sample.player;
 id | sport_team_id | last_name | first_name | full_name
----+---------------+-----------+------------+-----------
(0 rows)
```

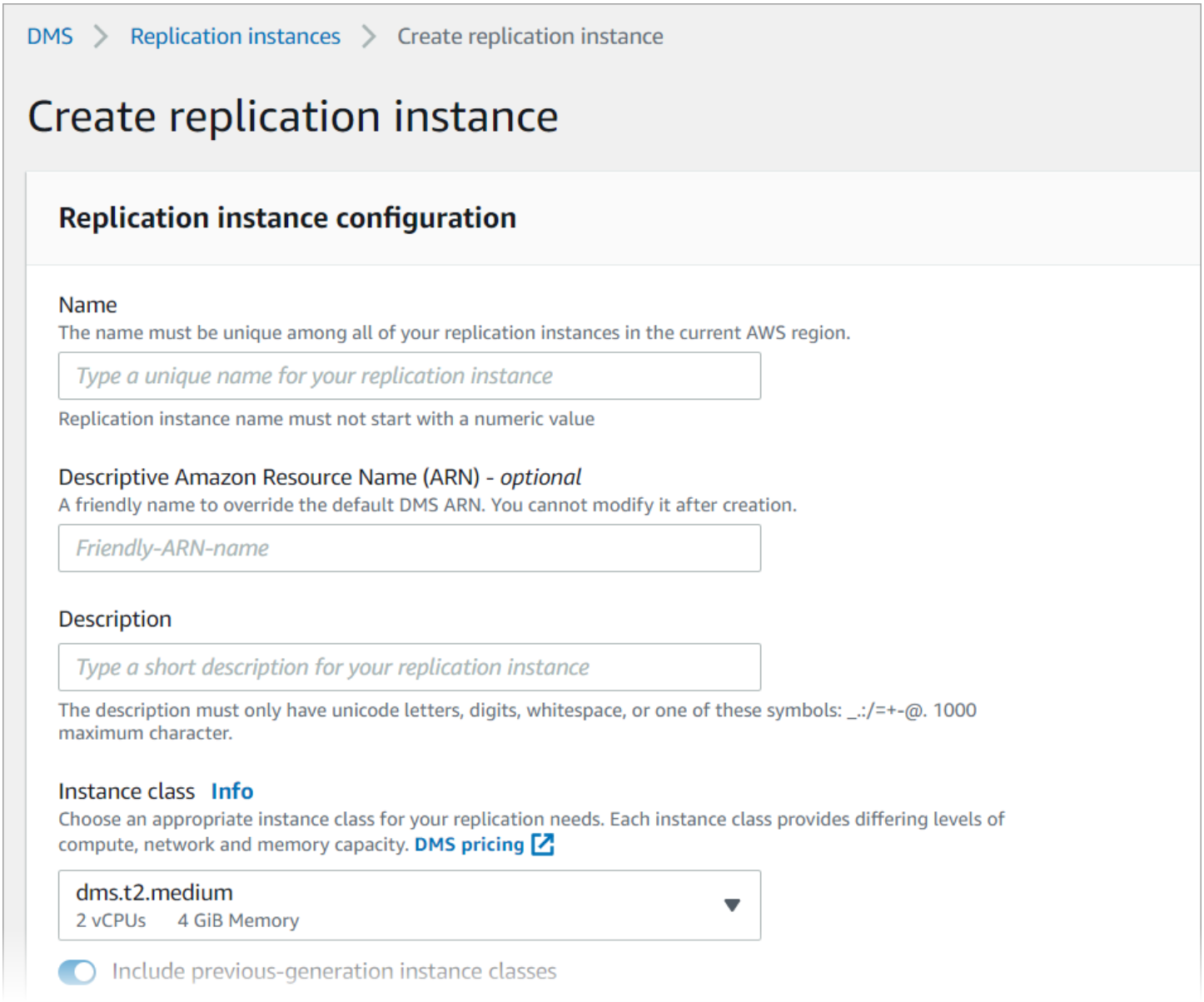
AWS Database Migration Service のレプリケーションの設定

このトピックでは、ソースデータベースとターゲットデータベース間のレプリケーションを設定します。

ステップ 1: AWS DMS コンソールを使用してレプリケーションインスタンスを作成する

AWS DMS の使用を開始するには、レプリケーションインスタンスを作成します。

レプリケーションインスタンスはソースエンドポイントとターゲットエンドポイント間での実際のデータ移行を実行します。インスタンスには、ソースデータベースからターゲットデータベースにデータを移行するタスクを実行するのに十分なストレージと処理能力が必要です。レプリケーションインスタンスのサイズは、移行するデータの量と、インスタンスが実行する必要があるタスクにより異なります。レプリケーションインスタンスの詳細については、「AWS DMS レプリケーションインスタンスの使用」を参照してください。



コンソールを使用してレプリケーションインスタンスを作成するには

1. AWS Management Console にサインインして、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. ナビゲーションペインで、[レプリケーションインスタンス] を選択して、[レプリケーションインスタンスの作成] をクリックします。
3. [レプリケーションインスタンスの作成] ページで、レプリケーションインスタンスの設定を次のとおり指定します。

   a. [名前] には **`DMS-instance`** と入力する。

b. [説明] には、このレプリケーションインスタンスの短い説明を入力する (オプション)。

c. [インスタンスクラス] には、dms.t3.medium を選択されたままにする。

インスタンスには移行に十分なストレージ、ネットワーク、処理能力が必要です。インスタンスクラスの選択方法の詳細については、「移行に適した AWS DMS レプリケーションインスタンスの選択」を参照。

d. [エンジンバージョン] はデフォルト値のままにする。

e. [マルチ AZ] には [開発またはテストワークロード (シングル AZ)] を選択する。

f. [割り当てられたストレージ (GiB)] (割り当てられたストレージ)では、デフォルトの 50 ギビバイト (GiB) のままにする。

AWS DMS の場合、ストレージは主にログファイルとキャッシュされたトランザクションが使用します。キャッシュされたトランザクションについては、キャッシュされたトランザクションをディスクに書き込む必要がある場合にのみストレージが使用されます。つまり、AWS DMS では大量のストレージは使用しません。

g. [ネットワークタイプ] では [IPv4] を選択する。

h. [VPC] では [DMSVPC] を選択する。

i. [レプリケーションサブネットグループ] では、現在選択されているレプリケーションサブネットグループのままにする。

j. [パブリックアクセス可能] はオフにする。

4. 必要に応じて、[セキュリティとネットワークの詳細設定] タブをクリックして、ネットワークと暗号化の値を次のとおり設定します。

a. [アベイラビリティーゾーン] では [us-west-2a] を選択する。

b. [VPC セキュリティグループ] では、まだ選択されていない場合は、[デフォルト] を選択する。

c. [AWS KMS key] では、[(デフォルト) aws/dms] が選択されたままにする。

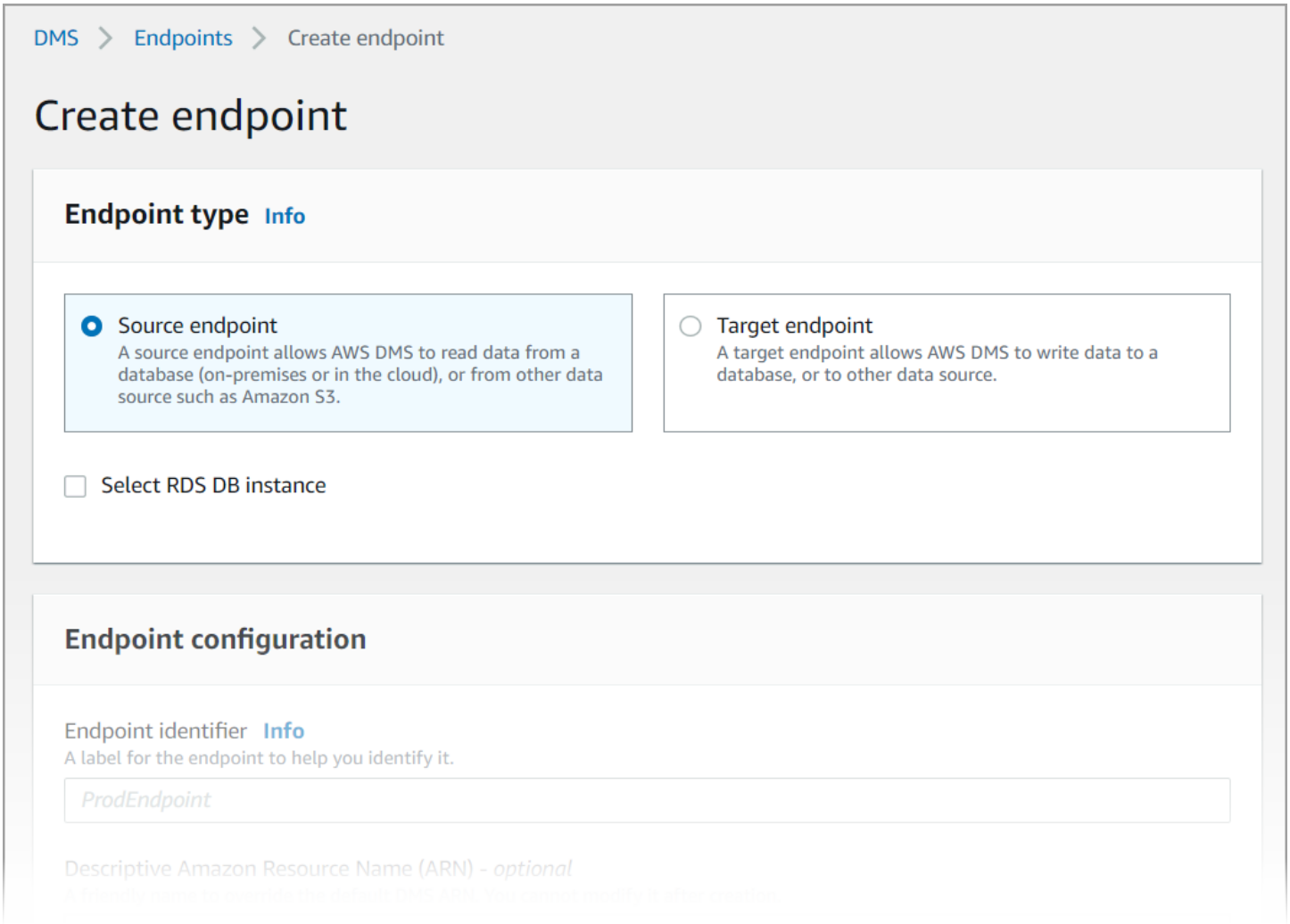
5. [メンテナンス] タブの設定はそのままにします。各 AWS リージョンの 8 時間の時間ブロックからランダムに選択された 30 分の時間枠で、ランダムな曜日に発生するというのがデフォルトです。

6. [作成] をクリックします。

AWS DMS は移行を実行するためのレプリケーションインスタンスを作成します。

ステップ 2: ソースエンドポイントとターゲットエンドポイントを指定する

レプリケーションインスタンスの作成時に、以前に作成した Amazon RDS データベースのソースデータ ストアエンドポイントとターゲットデータストアエンドポイントを指定できます。各エンドポイントは個別に作成します。



AWS DMS コンソールを使用して、ソースエンドポイントとターゲットデータベースエンドポイントを指定するには

1. コンソールで、ナビゲーションペインから [エンドポイント] を選択して、[エンドポイントの作成] をクリックします。
2. [エンドポイントの作成] ページで、[ソース] のエンドポイントタイプを選択します。[RDS DB インスタンスの選択] チェックボックスをオンにして、[dms-mariadb] インスタンスを選択します。

3. [エンドポイント設定] セクションの [エンドポイント識別子] には **dms-mysql-source** をに入力します。
4. [ソースエンジン] では [MySQL] を選択されたままにします。
5. [エンドポイントデータベースへのアクセス] では、[アクセス情報を手動で提供する] を選択します。[ポート]、[Secure Socket Layer (SSL) モード]、[ユーザー名]、[パスワード] が正しいことを検証します。
6. [エンドポイント接続のテスト (オプション)] タブをクリックします。[VPC] では [DMSVPC] を選択します。
7. [レプリケーションインスタンス] では、[dms-instance] が選択されたままにします。
8. [テストの実行] をクリックします。

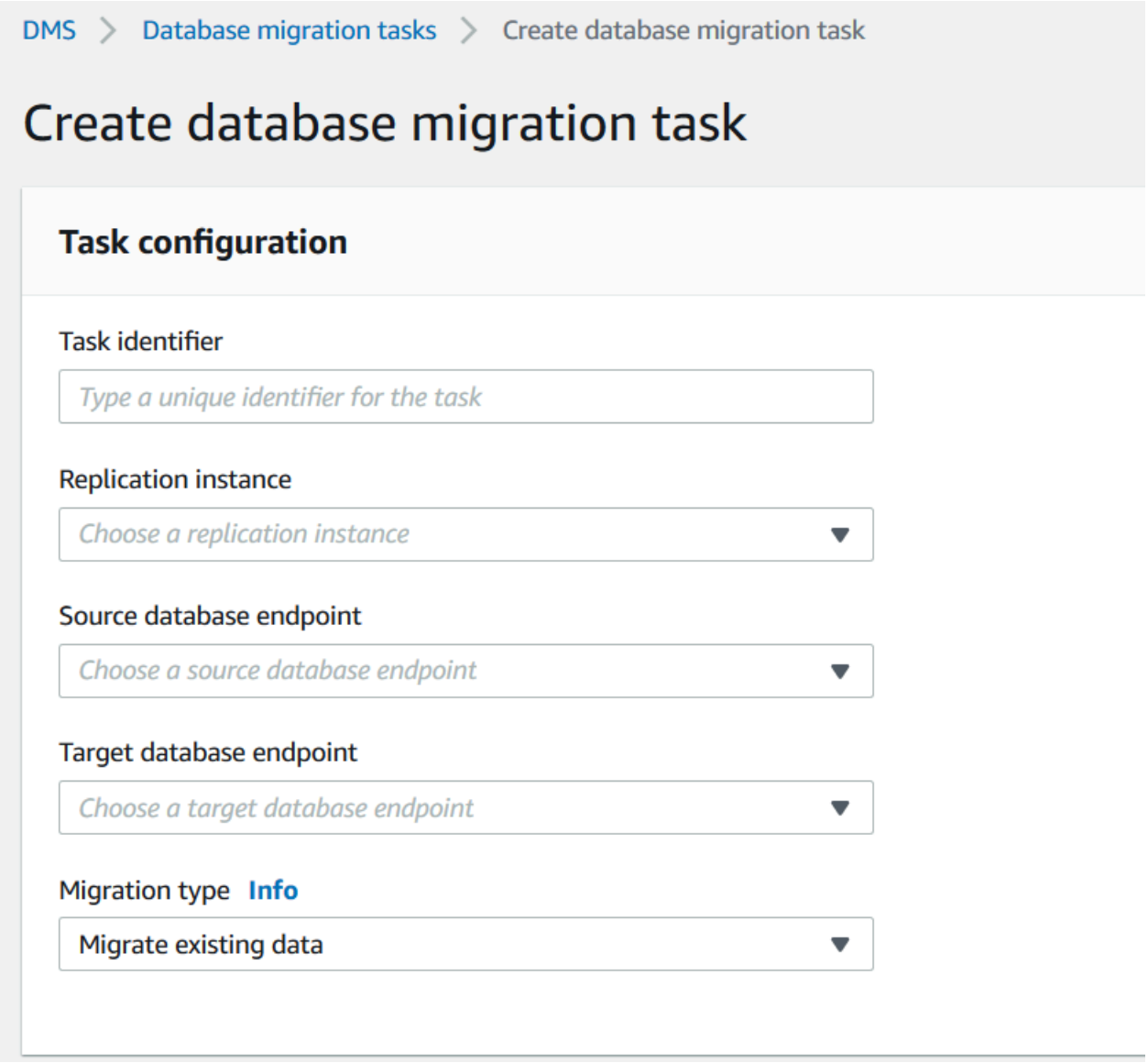
   [テストの実行] をクリックすると、AWS DMS は指定した詳細を使用してエンドポイントを作成し、このエンドポイントに接続します。接続が失敗した場合は、エンドポイントの定義を編集して、もう一度接続をテストします。エンドポイントは手動で削除することもできます。
9. テストが正常に完了したら、[エンドポイントの作成] をクリックします。
10. AWS DMS コンソールを使用してターゲットデータベースエンドポイントを指定します。これを実行するには、次の設定を使用して、前の手順を繰り返します。

    - エンドポイントタイプ: ターゲットエンドポイント
    - RDS インスタンス: dms-postgresql
    - エンドポイント識別子: **dms-postgresql-target**
    - ターゲットエンジン: **PostgreSQL** が選択されたままにする

エンドポイントについてのすべての情報の入力が完了すると、AWS DMS はデータベース移行中に使用するソースエンドポイントとターゲットエンドポイントを作成します。

ステップ 3: タスクを作成してデータを移行する

このステップでは、作成したデータベース間でデータを移行するタスクを作成します。



移行タスクを作成してデータベースの移行を開始するには

1. コンソールのナビゲーションペインで、[データベース移行タスク] を選択し、[タスクの作成] をクリックします。[データベース移行タスクの作成] ページが開きます。
2. [タスクの設定] セクションで、次のタスクのオプションを指定します。

   - タスク識別子: **dms-task** と入力する。
   - [レプリケーションインスタンス]: レプリケーションインスタンスを選択する (dms-instance-vpc-***<vpc id>***)。
   - [ソースデータベースエンドポイント]: [dms-mysql-source] を選択する。
   - [ターゲットデータベースエンドポイント]: [dms-postgresql-target] を選択する。

- [移行タイプ]: [既存データ移行と実行中変更のレプリケーション] を選択する。

3. [タスク設定] タブをクリックします。次のとおり設定します。

   - ターゲットテーブル作成モード: 何もしない
   - フルロードの完了後にタスクを停止する: 停止しない

4. [テーブルマッピング] タブをクリックして、[選択ルール] を展開します。[新しい選択ルールの追加] をクリックします。次のとおり設定します。

   - スキーマ: スキーマの入力
   - スキーマ名: **dms_sample**

5. [移行タスクのスタートアップ設定] タブをクリックして、[作成時に自動的に行う] を選択します。
6. [タスクの作成] をクリックします。

AWS DMS は次に、この移行タスクを作成して開始します。最初のデータベースレプリケーションには約 10 分かかります。AWS DMS がデータ移行を完了する前に、必ずチュートリアルの次のステップを実行します。

ステップ 4: レプリケーションをテストする

このセクションでは、最初のレプリケーション中とレプリケーション後にデータをソースデータベースに挿入し、挿入したデータをターゲットデータベースでクエリします。

レプリケーションをテストするには

1. データベース移行タスクのステータスが [実行中)] と表示され、前のステップで開始した最初のデータベースレプリケーションは完了していないことを確認します。
2. Amazon EC2 クライアントに接続して、次のコマンドを使用して MySQL クライアントを起動します。MySQL データベースのエンドポイントを指定します。

```
mysql -h dms-mysql.abcdefg12345.us-west-2.rds.amazonaws.com -P 3306 -u admin -
pchangeit dms_sample
```

3. 次のコマンドを実行して、ソースデータベースにレコードを挿入します。

```
MySQL [dms_sample]> insert person (full_name, last_name, first_name) VALUES ('Test
 User1', 'User1', 'Test');
```

```
Query OK, 1 row affected (0.00 sec)
```

4. MySQL クライアントを終了します。

```
MySQL [dms_sample]> exit
Bye
```

5. レプリケーションが完了する前に、ターゲットデータベースで新しいレコードをクエリします。

   Amazon EC2 インスタンスから、次のコマンドを使用して、ターゲットデータベースエンドポイントを指定し、ターゲットデータベースに接続します。

```
psql \
   --host=dms-postgresql.abcdefg12345.us-west-2.rds.amazonaws.com \
   --port=5432 \
   --username=postgres \
   --password \
   --dbname=dms_sample
```

   プロンプトが表示されたらパスワード (**changeit**) を入力します。

6. レプリケーションが完了する前に、ターゲットデータベースで新しいレコードをクエリします。

```
dms_sample=> select * from dms_sample.person where first_name = 'Test';
 id | full_name | last_name | first_name
----+-----------+-----------+------------
(0 rows)
```

7. 移行タスクの実行中は、データベース移行の進捗を次のとおりモニタリングできます。

   - DMS コンソールのナビゲーションペインで [データベース移行タスク] を選択する。
   - [dms-task] を選択する。
   - [テーブル統計] を選択する。

   モニタリングの詳細については、「AWS DMS タスクのモニタリング」を参照してください。

8. レプリケーションが完了したら、ターゲットデータベースでもう一度新しいレコードをクエリします。最初のレプリケーションが完了した後、AWS DMS は新しいレコードを移行します。

```
dms_sample=> select * from dms_sample.person where first_name = 'Test';
   id    | full_name  | last_name | first_name
---------+------------+-----------+------------
```

```
 7077784 | Test User1 | User1     | Test
(1 row)
```

9. psql クライアントを終了します。

```
dms_sample=> quit
```

10. ステップ 1 を繰り返して、ソースデータベースにもう一度接続します。
11. 別のレコードを person テーブルに挿入します。

```
MySQL [dms_sample]> insert person (full_name, last_name, first_name) VALUES ('Test
 User2', 'User2', 'Test');
Query OK, 1 row affected (0.00 sec)
```

12. ステップ 3 と 4 を繰り返して、ソースデータベースの接続を切断し、ターゲットデータベースに接続します。
13. レプリケートされたデータをターゲットデータベースでもう一度クエリします。

```
dms_sample=> select * from dms_sample.person where first_name = 'Test';
   id    | full_name  | last_name | first_name
---------+------------+-----------+------------
 7077784 | Test User1 | User1     | Test
 7077785 | Test User2 | User2     | Test
(2 rows)
```

ステップ 5: AWS DMS リソースをクリーンアップする

使用開始のチュートリアルを完了したら、作成したリソースを削除できます。削除するには、AWS コンソールを使用できます。レプリケーションインスタンスとエンドポイントを削除する前に、必ず移行タスクを削除します。

コンソールを使用して移行タスクを削除するには

1. AWS DMS コンソール ナビゲーションペインで、[データベース移行タスク] を選択します。
2. [dms-task] を選択します。
3. [アクション]、[削除] の順に選択します。

コンソールを使用してレプリケーションインスタンスを削除するには

1. AWS DMS コンソールのナビゲーションペインで、[レプリケーションインスタンス] を選択します。
2. [DMS インスタンス] を選択します。
3. [アクション]、[削除] の順に選択します。

AWS DMS は、このレプリケーションインスタンスを削除して、[レプリケーションインスタンス] ページから削除します。

コンソールを使用してエンドポイントを削除するには

1. AWS DMS コンソールのナビゲーションペインで [エンドポイント] を選択します。
2. [dms-mysql-source] を選択します。
3. [アクション]、[削除] の順に選択します。

AWS DMS のリソースの削除後、次のリソースも必ず削除します。その他のサービスのリソースの削除についてのサポートは、各サービスのドキュメントを参照してください。

- RDS データベース
- RDS データベースのパラメータグループ
- RDS サブネットグループ
- データベースとレプリケーションインスタンスとともに作成されたすべての Amazon CloudWatch ログ
- Amazon VPC と Amazon EC2 クライアントのために作成されたセキュリティグループ。[launch-wizard-1] のセキュリティグループのインバウンドルールは、必ず [デフォルト] から削除します。セキュリティグループを削除するために必要です。
- Amazon EC2 クライアント
- Amazon VPC
- Amazon EC2 クライアントのための Amazon EC2 キーペア

AWS Database Migration Service を使用するためのその他のリソース

このガイドの後半では、AWS DMS を使用して、最も広く使用されている商用データベースやオープンソースデータベースとの間でデータを移行する方法について説明しています。

データベース移行プロジェクトを準備して実行する際には、次のリソースも確認しておくことをお勧めします。

- AWS DMS Step-by-Step Migration Guide – このガイドでは、AWS へのデータ移行プロセスのステップバイステップのチュートリアルを提供しています。
- AWS DMS API リファレンス – このリファレンスでは、AWS Database Migration Service のすべての API オペレーションを詳細に説明しています。
- AWS CLI for AWS DMS – このリファレンスでは、AWS DMS での AWS Command Line Interface (AWS CLI) の使用についての情報を提供しています。

AWS DMS Fleet Advisor を使用した移行用データベースの検出と評価

DMS Fleet Advisor を使用して、複数のデータベース環境からメタデータとパフォーマンスメトリクスを収集できます。このように収集したメトリクスは、データインフラストラクチャに関するインサイトを提供します。DMS Fleet Advisor は、すべてのコンピュータにインストールする必要なく、単一または複数の中央ロケーションからオンプレミスのデータベースと分析サーバーのメタデータとメトリクスを収集します。現在、DMS Fleet Advisor は、Microsoft SQL Server、MySQL、Oracle、PostgreSQL データベースサーバーの検出とメトリクスの収集をサポートしています。

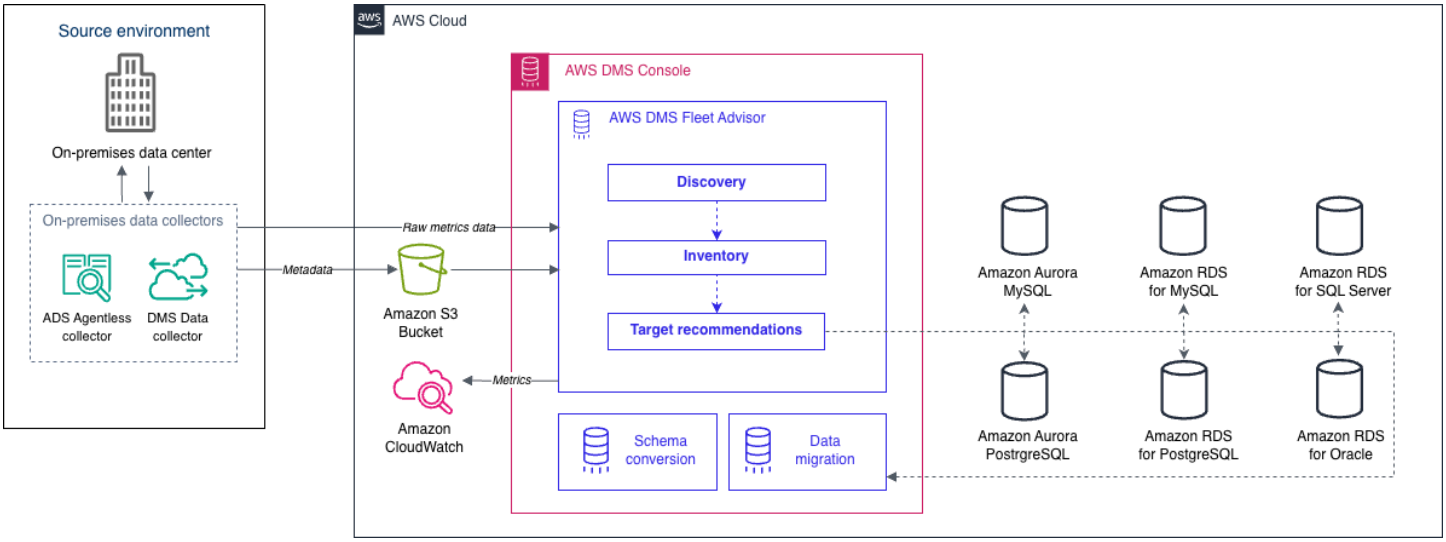
ネットワークから検出されたデータに基づいて、インベントリを構築し、さらにデータを収集するためのデータベースサーバーのリストを定義できます。AWS DMS がサーバー、データベース、スキーマの情報を収集した後、ユーザーは意図するデータベースの移行の実現可能性を分析できます。

AWS クラウド への移行を計画しているインベントリ内のデータベースについては、DMS Fleet Advisor が適切なサイズのターゲットレコメンデーションを生成します。ターゲットレコメンデーションを生成するうえで、DMS Fleet Advisor はデータコレクターからのメトリクスと優先設定を考慮に入れます。DMS Fleet Advisor がレコメンデーションを生成した後、各ターゲットデータベースの設定の詳細情報を確認できます。組織のデータベースエンジニアと管理者は、DMS Fleet Advisor のターゲットレコメンデーションを使用して、オンプレミスのデータベースの AWS への移行を計画できます。利用可能なさまざまな移行オプションを調べ、これらの推奨事項を にエクスポートAWS Pricing Calculatorして、コストをさらに最適化できます。

サポート対象のソースデータベースの一覧については、「DMS Fleet Advisor のソース」を参照してください。

DMS Fleet Advisor がターゲットレコメンデーションの生成に使用するデータベース一覧については、「DMS Fleet Advisor のターゲット」を参照してください。DMS Fleet Advisor は、例えばソース Oracle からターゲット Oracle データベースへのレコメンデーションのように を生成します。DMS Fleet Advisor は、ソース Oracle または Microsoft SQL Server からターゲット RDS for PostgreSQL または Aurora PostgreSQL データベースへの移行など、異種のレコメンデーションも生成します。

次の図は、AWS DMS Fleet Advisor のターゲットレコメンデーションのプロセスを説明しています。



AWS DMS Fleet Advisor の使用方法の理解を深めるには、次のトピックを利用できます。

トピック

- サポートされている AWS リージョン
- DMS Fleet Advisor の開始方法
- AWS DMS Fleet Advisor の設定
- データコレクターを使用した移行用データベースの検出
- AWS DMS Fleet Advisor での分析のためのインベントリの使用
- AWS DMS Fleet Advisor ターゲットレコメンデーション機能の使用
- DMS Fleet Advisor の制限事項

サポートされている AWS リージョン

DMS Fleet Advisor は、次の AWS リージョン で利用できます。

リージョン名	リージョン
米国東部 (バージニア北部)	us-east-1
米国東部 (オハイオ)	us-east-2

リージョン名	リージョン
米国西部 (北カリフォルニア)	us-west-1
米国西部 (オレゴン)	us-west-2
アジアパシフィック (香港)	ap-east-1
アジアパシフィック (東京)	ap-northeast-1
アジアパシフィック (ソウル)	ap-northeast-2
アジアパシフィック (大阪)	ap-northeast-3
アジアパシフィック (ムンバイ)	ap-south-1
アジアパシフィック (シンガポール)	ap-southeast-1
アジアパシフィック (シドニー)	ap-southeast-2
アジアパシフィック (ジャカルタ)	ap-southeast-3
カナダ (中部)	ca-central-1
欧州 (フランクフルト)	eu-central-1
欧州 (ストックホルム)	eu-north-1
欧州 (アイルランド)	eu-west-1
欧州 (ロンドン)	eu-west-2
欧州 (パリ)	eu-west-3

リージョン名	リージョン
欧州 (ミラノ)	eu-south-3
カナダ (中部)	ca-central-1
南米 (サンパウロ)	sa-east-1
中東 (バーレーン)	me-south-1
アフリカ (ケープタウン)	af-south-1

DMS Fleet Advisor の開始方法

DMS Fleet Advisor を使用すると、AWS クラウド に移行するソースのオンプレミスのデータベースを検出できます。その後、オンプレミスのデータベースそれぞれに適切な AWS クラウド の移行ターゲットを決定できます。次のワークフローを使用して、ソースデータベースのインベントリを作成し、ターゲットレコメンデーションを生成します。

1. Amazon S3 バケット、IAM ポリシー、ロール、ユーザーを作成します。詳細については、「必要なリソースの作成」を参照してください。
2. DMS データコレクターに必要な最小限のアクセス許可を持つデータベースユーザーを作成します。詳細については、「データベースユーザーの作成」を参照してください。
3. データコレクターを作成してダウンロードします。詳細については、「データコレクターの作成」を参照してください。
4. データコレクターをローカル環境にインストールします。その後、収集したデータを DMS Fleet Advisor に送信できるようにデータコレクターを設定します。詳細については、「データコレクターのインストール」を参照してください。
5. データ環境内の OS とデータベースサーバーを検出します。詳細については、「OS サーバーとデータベースサーバーの検出」を参照してください。
6. データベースのメタデータとリソース使用率のメトリクスを収集します。詳細については、「データ収集」を参照してください。
7. ソースデータベースとスキーマを分析します。DMS Fleet Advisor はデータベースの大規模な評価を行い、類似のスキーマを特定します。詳細については、「AWS DMS Fleet Advisor での分析のためのインベントリの使用」を参照してください。

8. ソースデータベースのターゲットレコメンデーションのローカルコピーを生成、確認、保存します。詳細については、「ターゲットレコメンデーション」を参照してください。

各ソースデータベースの移行ターゲットを決定したら、DMS Schema Conversion を使用してデータベーススキーマを新しいプラットフォームに変換できます。その後、AWS DMS を使用してデータを移行します。詳細については、「DMS Schema Conversion を使用したデータベーススキーマの変換」と「AWS Database Migration Service とは」を参照してください。

この動画では、DMS Schema Conversion ユーザーインターフェイスと、このサービスのコアコンポーネントを概説しています。

AWS DMS Fleet Advisor の設定

AWS DMS Fleet Advisor をセットアップするには、次の前提タスクを完了してください。

トピック

- AWS DMS Fleet Advisor に必要な AWS リソースの作成
- AWS DMS Fleet Advisor のデータベースユーザーの作成

AWS DMS Fleet Advisor に必要な AWS リソースの作成

DMS Fleet Advisor では、インベントリ情報の転送とインポート、DMS データコレクターのステータスの更新を行うために、AWS アカウントにリソースセットが必要となります。

初めてデータを収集し、データベースとスキーマのインベントリを作成する前に、次の前提条件を満たす必要があります。

Amazon S3 バケットおよび IAM リソースを設定するには、次のいずれかを実行します。

- AWS CloudFormation を使用して Amazon S3 および IAM リソースを設定する (推奨)
- AWS Management Console で Amazon S3 および IAM リソースを設定する

AWS CloudFormation を使用して Amazon S3 および IAM リソースを設定する

CloudFormation スタックは、単一のユニットとして管理できる AWS リソースのコレクションです。DMS Fleet Advisor に必要なリソースを簡単に作成するために、AWS CloudFormationテンプレートファイルを使用して CloudFormation スタックを作成できます。詳細については、AWS

CloudFormation ユーザーガイドの「AWS CloudFormation コンソールでのスタックの作成」を参照してください。

> **Note**
>
> このセクションは、スタンドアロンの DMS Fleet Advisor コレクターを使用する場合にのみ該当します。単一のオンプレミスコレクターを使用してデータベースとサーバーの両方に関する情報を収集する方法については、「AWS Application Discovery Service ユーザーガイド」の「Application Discovery Service Agentless Collector」を参照してください。

によって作成された Amazon S3 および IAM リソース CloudFormation

CloudFormation テンプレートを使用すると、 に次のリソースを含むスタックが作成されますAWS アカウント。

- `dms-fleetadvisor-data-accountId-region` という名前の Amazon S3 バケット
- `FleetAdvisorCollectorUser-region` という名前の IAM ユーザー
- `FleetAdvisorS3Role-region` という名前の IAM サービスロール
- `FleetAdvisorS3Role-region-Policy` という名前のアクセスポリシー
- `FleetAdvisorCollectorUser-region-Policy` という名前のアクセスポリシー
- `AWSServiceRoleForDMSFleetAdvisor` という名前の IAM サービスリンクロール (SLR)

以下の手順に従って、 でリソースを設定します CloudFormation。

- ステップ 1: テンプレートファイルをダウンロードする CloudFormation
- ステップ 2: を使用して Amazon S3 と IAM を設定する CloudFormation

ステップ 1: テンプレートファイルをダウンロードする CloudFormation

CloudFormation テンプレートは、スタックを構成するAWSリソースの宣言です。テンプレートは JSON ファイルとして保存されます。

CloudFormation テンプレートファイルをダウンロードするには

1. 次のリンクのいずれかでコンテキスト (右クリック) メニューを開き、[名前を付けてリンクを保存] を選択します。

- DMS Fleet Advisor を使用する場合は、dms-fleetadvisor-iam-slr-s3.zip を選択します。DMS Fleet Advisor の SLR を既に作成している場合は、dms-fleetadvisor-iam-s3.zip を選択します。
- AWS Application Discovery Service (ADS) エージェントレスコレクターを使用する予定で、SLR を作成していない場合は、 dms-fleetadvisor-ads-iam-slr-s3.zip を選択します。以前に ADS で DMS Fleet Advisor の SLR を作成したことがある場合は、 dms-fleetadvisor-ads-iam-s3.zip を選択します。

2. ファイルをコンピュータに保存します。

ステップ 2: を使用して Amazon S3 と IAM を設定する CloudFormation

IAM の CloudFormation テンプレートを使用すると、前述の Amazon S3 と IAM リソースが作成されます。

を使用して Amazon S3 と IAM を設定するには CloudFormation

1. https://console.aws.amazon.com/cloudformation で CloudFormation コンソールを開きます。
2. [スタックを作成] を選択し、ドロップダウンリストで [新しいリソースを使用] を選択して、スタックの作成ウィザードを起動します。
3. [Create stack] (スタックの作成) ページで、次の手順を実行します。
   a. [Prepare template] (テンプレートの準備) の [Template is ready] (テンプレートの準備が完了しました) を選択します。
   b. [Template source] (テンプレートのソース) で、[Upload a template file] (テンプレートファイルのアップロード) を選択します。
   c. ファイルの選択 で、-dms-fleetadvisor-iam-slr-S3.json 、-dms-fleetadvisor-iam-S3.json.、dms-fleetadvisor-ads-iam-slr-s3.zip 、または -s3.zip を選択します。 dms-fleetadvisor-ads-iam
   d. [次へ] をクリックします。
4. [Specify stack details] (プロジェクト詳細の指定) ページで、以下を実行します。
   a. [スタック名] に「**dms-fleetadvisor-iam-slr-s3**」、「**dms-fleetadvisor-iam-s3**」、「**dms-fleetadvisor-ads-iam-slr-s3**」、または「**dms-fleetadvisor-ads-iam-s3**」と入力します。
   b. [次へ] をクリックします。

5. [Configure stack options] (スタックオプションの設定) ページで、[Next] (次へ) を選択します。
6. 「レビュー dms-fleetadvisor-iam-slr-s3」、「レビュー dms-fleetadvisor-iam-s3」、「レビュー dms-fleetadvisor-ads-iam-slr-s3」、または「レビュー dms-fleetadvisor-ads-iam-s3」ページで、次の操作を行います。

   a. [AWS CloudFormation がカスタム名で IAM リソースを作成する可能性があることに同意する] チェックボックスを選択します。

   b. [送信] を選択します。

   CloudFormation は、DMS Fleet Advisor に必要な S3 バケット、IAM ロール、ユーザーを作成します。左側のパネルで、dms-fleetadvisor-iam-slr-s3、dms-fleetadvisor-iam-s3、dms-fleetadvisor-ads-iam-slr-s3、または dms-fleetadvisor-ads-iam-s3 が CREATE_COMPLETE と表示されている場合は、次のステップに進みます。
7. 左側のパネルで、dms-fleetadvisor-iam-slr-s3、dms-fleetadvisor-iam-s3、dms-fleetadvisor-ads-iam-slr-s3、または dms-fleetadvisor-ads-iam-s3 を選択します。右のパネルで、以下の操作を行います。

   a. [スタック情報] を選択します。スタックの ID は、arn:aws:cloudformation:***region*** :***account-no*** :stack/dms-fleetadvisor-iam-slr-s3/***identifier*** 、arn:aws:cloudformation:***region*** :***account-no*** :stack/dms-fleetadvisor-iam-s3/***identifier*** 、arn:aws:cloudformation:***region*** :***account-no*** :stack/dms-fleetadvisor-ads-iam-slr-s3/***identifier*** 、または arn:aws:cloudformation:***region*** :***account-no*** :stack/dms-fleetadvisor-ads-iam-s3/***identifier*** です。

   b. [Resources] (リソース) を選択します。次のように表示されます。

      - dms-fleetadvisor-data-*accountId*-*region* という名前の Amazon S3 バケット
      - FleetAdvisorS3Role-*region* という名前のサービスロール
      - FleetAdvisorCollectorUser-*region* という名前の IAM ユーザー
      - AWSServiceRoleForDMSFleetAdvisor という名前の IAM SLR (dms-fleet-advisor-iam-slr-s3.zip または dms-fleet-advisor-ads-iam-slr-s3.zip をダウンロードした場合)
      - FleetAdvisorS3Role-*region*-Policy という名前のアクセスポリシー
      - FleetAdvisorCollectorUser-*region*-Policy という名前のアクセスポリシー

AWS Management Console で Amazon S3 および IAM リソースを設定する

Amazon S3 バケットを作成する

インベントリのメタデータを保存する Amazon S3 バケットを作成します。DMS Fleet Advisor を使用する前に、この S3 バケットを事前に設定しておくことをお勧めします。AWS DMS は、DMS Fleet Advisor のインベントリメタデータをこの S3 バケットに保存します。

S3 バケットを作成する詳細については、Amazon S3 ユーザーガイド の「最初のS3バケットを作成する」をご参照ください。

> **Note**
>
> DMS Fleet Advisor は SSE-S3 暗号化バケットのみをサポートします。

Amazon S3 バケットを作成して、ローカルデータ環境情報を保存するには

1. AWS Management Console にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. [バケットの作成] を選択します。
3. [バケットの作成] ページで、「fa-bucket-***yoursignin***」などのバケットのサインイン名を含むグローバルに一意の名前を入力します。
4. DMS Fleet Advisor を使用する AWS リージョン を選択します。
5. 残りの設定はそのままにして [バケットを作成] をクリックします。

IAM リソースを作成する

このセクションでは、データコレクター、IAM ユーザー、DMS Fleet Advisor のための IAM リソースを作成します。

トピック

- データコレクターのための IAM リソースを作成する
- DMS Fleet Advisor サービスにリンクされたロールを作成する

データコレクターのための IAM リソースを作成する

データコレクターが適切に動作し、収集したメタデータを Amazon S3 バケットにアップロードするには、次のポリシーを作成します。その後、次のアクセス許可を持つ IAM ロールを作成します。DMS データコレクターの詳細については、「データコレクターを使用した移行用データベースの検出」を参照してください。

DMS Fleet Advisor とデータコレクターが Amazon S3 にアクセスするための IAM ポリシーを作成するには

1. AWS Management Console にサインインして、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、[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. [ポリシーの作成] ページで、[JSON] タブをクリックします。
5. 次の JSON をエディタに貼り付けて、サンプルコードを置き換えます。*fa_bucket* は、前のセクションで作成した Amazon S3 バケット名と置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*",
                "s3:DeleteObject*",
                "s3:PutObject*"
            ],
            "Resource": [
                "arn:aws:s3:::fa_bucket",
                "arn:aws:s3:::fa_bucket/*"
            ]
        }
    ]
}
```

6. [次へ: タグ]、[次へ: 確認] の順に選択します。
7. [名前*] には **FleetAdvisorS3Policy** と入力して、[ポリシーを作成] をクリックします。

DMS データコレクターが DMS Fleet Advisor にアクセスするための IAM ポリシーを作成するには

1. AWS Management Console にサインインして、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、[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. [ポリシーの作成] ページで、[JSON] タブをクリックします。
5. 次の JSON コードをエディタに貼り付けて、サンプルコードを置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dms:DescribeFleetAdvisorCollectors",
                "dms:ModifyFleetAdvisorCollectorStatuses",
                "dms:UploadFileMetadataList"
            ],
            "Resource": "*"
        }
    ]
}
```

6. [次へ: タグ]、[次へ: 確認] の順に選択します。
7. [名前*] には **DMSCollectorPolicy** と入力して、[ポリシーを作成] をクリックします。

DMS データコレクターを使用するための最小限のアクセス許可を持つ IAM ユーザーを作成するには

1. AWS Management Consoleにサインインして、IAM コンソールを開きます https://console.aws.amazon.com/iam/。
2. ナビゲーションペインで [Users (ユーザー)] を選択します。
3. [ユーザーの追加] を選択します。
4. [ユーザーの追加] ページの [ユーザー名*] に **FleetAdvisorCollectorUser** と入力します。[Select AWS Access Type] には [アクセスキー - プログラムによるアクセス] を選択します。[次へ: アクセス許可] を選択します。
5. [許可を設定] セクションで、[既存のポリシーを直接アタッチする] を選択します。

6. 検索コントロールを使用して、以前に作成した DMSCollectorPolicy ポリシーと FleetAdvisorS3Policy ポリシーを検索して選択します。[次へ: タグ] を選択します。
7. [Tags (タグ)] ページで、[Next: Review (次へ: レビュー)] を選択します。
8. [確認] ページで、[ユーザーの作成] を選択します。次のページで Download.csv をクリックして、新しいユーザー認証情報を保存します。必要最小限のアクセス許可については、DMS Fleet Advisor でこれらの認証情報を使用します。

DMS Fleet Advisor とデータコレクターが Amazon S3 にアクセスするための IAM ロールを作成するには

1. AWS Management Console にサインインして、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. ナビゲーションペインで [Roles] (ロール) を選択します。
3. ロールの作成 を選択します。
4. [信頼されたエンティティを選択] ページの [信頼されたエンティティタイプ] では、AWS[サービス] を選択します。[その他の AWS サービスのユースケース] では、[DMS] を選択します。
5. [DMS] チェックボックスをオンにして、[次へ] をクリックします。
6. アクセス許可の追加ページで、FleetAdvisorS3Policy を選択します。[次へ] をクリックします。
7. [名前、確認、および作成] ページで、[ロール名] に **`FleetAdvisorS3Role`** と入力して、[ロールの作成] をクリックします。
8. [ロール] ページの [ロール名] には、**`FleetAdvisorS3Role`** と入力します。FleetAdvisorS3Role を選択します。
9. FleetAdvisorS3Role ページで、信頼関係タブを選択します。[Edit trust policy] (信頼ポリシーを編集) を選択します。
10. [信頼ポリシーを編集] ページで、次の JSON コードをエディタに貼り付けて、既存のテキストを置き換えます。

```
{
 "Version": "2012-10-17",
 "Statement": [
    {
     "Sid": "",
     "Effect": "Allow",
     "Principal": {
       "Service": [
         "dms.amazonaws.com",
```

```
          "dms-fleet-advisor.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

上記のポリシーは、AWS DMS が Amazon S3 バケットから収集したデータをインポートするために使用するサービスへの `sts:AssumeRole` アクセス許可を付与します。

11. [ポリシーの更新] を選択します。

DMS Fleet Advisor サービスにリンクされたロールを作成する

DMS Fleet Advisor は、サービスにリンクされたロールを使用して、 の Amazon CloudWatch メトリクスを管理しますAWS アカウント。DMS Fleet Advisor は、このサービスにリンクされたロールを使用して、収集したデータベースパフォーマンスメトリクスをユーザー CloudWatch に代わって に公開します。

DMS Fleet Advisor のサービスにリンクされたロールを作成するには

1. AWS Management Console にサインインして、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. ナビゲーションペインで [Roles] (ロール) を選択します。続いて、[Create role] (ロールの作成) を選択します。
3. [信頼できるエンティティタイプ] で、[AWS サービス] を選択します。
4. [その他の AWS サービスのユースケース] では、[DMS – Fleet Advisor] を選択します。
5. [DMS – Fleet Advisor] チェックボックスをオンにして、[次へ] をクリックします。
6. [アクセス許可を追加] ページで [次へ] を選択します。
7. [名前、確認、および作成] ページで、[ロールの作成] をクリックします。

このようなサービスにリンクされたロールは、AWS API または AWS CLI から作成することもできます。詳細については、「AWS DMS Fleet Advisor のサービスにリンクされたロールの作成」を参照してください。

DMS Fleet Advisor のサービスにリンクされたロールを作成すると、ターゲットレコメンデーションにソースデータベースのパフォーマンスメトリクスが表示されます。また、 CloudWatch アカウント

でこれらのメトリクス および を確認することもできます。詳細については、「ターゲットレコメンデーション」を参照してください。

DMS Fleet Advisor のサービスにリンクされたロールに必要な IAM ポリシーを作成するには

サービスリンクロールを作成するために最低限必要なアクセス許可は `DMSFleetAdvisorCreateServiceLinkedRolePolicy` ポリシーで指定します。サービスリンクロールを作成できない場合は、この IAM ポリシーをアカウントで作成します。

1. AWS Management Console にサインインして、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、[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. [ポリシーの作成] ページで、[JSON] タブをクリックします。
5. 次の JSON コードをエディタに貼り付けて、サンプルコードを置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/dms-fleet-
advisor.amazonaws.com/AWSServiceRoleForDMSFleetAdvisor*",
            "Condition": {"StringLike": {"iam:AWSServiceName": "dms-fleet-
advisor.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/dms-fleet-
advisor.amazonaws.com/AWSServiceRoleForDMSFleetAdvisor*"
        }
    ]
}
```

6. [次へ: タグ]、[次へ: 確認] の順に選択します。

7. [名前*] には **DMSFleetAdvisorCreateServiceLinkedRolePolicy** と入力して、[ポリシーを作成] をクリックします。

これで、このポリシーを使用して DMS Fleet Advisor のサービスにリンクされたロールを作成できるようになりました。

AWS DMS Fleet Advisor のデータベースユーザーの作成

このセクションでは、DMS データコレクターに必要な最小限の権限を持つソースデータベースのユーザーを作成する方法について説明します。

このセクションは、以下のトピックで構成されます。

- AWS DMS Fleet Advisor でのデータベースユーザーの使用
- MySQL でのデータベースユーザーの作成
- Oracle でのデータベースユーザーの作成
- PostgreSQL でのデータベースユーザーの作成
- Microsoft SQL Server でのデータベースユーザーの作成
- データベースユーザーの削除

AWS DMS Fleet Advisor でのデータベースユーザーの使用

DMS データコレクターでは、`root` 以外のデータベースユーザーを使用できます。データベースをインベントリに追加した後、データコレクターを実行する前に、ユーザー名とパスワードを指定します。インベントリへのデータベースの追加の詳細については、「モニタリング対象オブジェクトの管理」を参照してください。

DMS データコレクターの使用を終えたら、作成したデータベースユーザーは削除できます。詳細については、「データベースユーザーの削除」を参照してください。

> **Important**
>
> 次の例では、*{your_user_name}* を、データベース用に作成したデータベースユーザー名に置き換えます。その後、*{your_password}* を安全なパスワードに置き換えます。

MySQL でのデータベースユーザーの作成

MySQL ソースデータベースでデータベースユーザーを作成するには、次のスクリプトを使用します。MySQL データベースのバージョンに応じて、GRANT ステートメントのいずれかのバージョンを必ず残します。

```
CREATE USER {your_user_name} identified BY '{your_password}';

GRANT PROCESS ON *.* TO {your_user_name};
GRANT REFERENCES ON *.* TO {your_user_name};
GRANT TRIGGER ON *.* TO {your_user_name};
GRANT EXECUTE ON *.* TO {your_user_name};

# For MySQL versions lower than 8.0, use the following statement.
GRANT SELECT, CREATE TEMPORARY TABLES ON `temp`.* TO {your_user_name};

# For MySQL versions 8.0 and higher, use the following statement.
GRANT SELECT, CREATE TEMPORARY TABLES ON `mysql`.* TO {your_user_name};

GRANT SELECT ON performance_schema.* TO {your_user_name};

SELECT
    IF(round(Value1 + Value2 / 100 + Value3 / 10000, 4) > 5.0129, 'GRANT EVENT ON *.*
 TO {your_user_name};', 'SELECT ''Events are not applicable'';') sql_statement
INTO @stringStatement
FROM (
       SELECT
         substring_index(ver, '.', 1)                              value1,
         substring_index(substring_index(ver, '.', 2), '.', - 1) value2,
         substring_index(ver, '.', - 1)                            value3
       FROM  (
         SELECT
           IF((@@version regexp '[^0-9\.]+') != 0, @@innodb_version, @@version) AS ver
         FROM dual
       ) vercase
     ) v;

PREPARE sqlStatement FROM @stringStatement;
SET @stringStatement := NULL;
EXECUTE sqlStatement;
DEALLOCATE PREPARE sqlStatement;
```

Oracle でのデータベースユーザーの作成

Oracle ソースデータベースでデータベースユーザーを作成するには、次のスクリプトを使用します。

この SQL スクリプトを実行するには、SYSDBA 権限を使用して Oracle データベースに接続します。この SQL スクリプトを実行した後、このスクリプトで作成したユーザーの認証情報を使用してデータベースに接続します。DMS データコレクターの実行にもこのユーザーの認証情報を使用します。

次のスクリプトは、Oracle マルチテナントコンテナデータベース (CDB) のユーザー名に C## プレフィックスを追加します。

```
CREATE USER {your_user_name} IDENTIFIED BY "{your_password}";
GRANT CREATE SESSION TO {your_user_name};
GRANT SELECT ANY DICTIONARY TO {your_user_name};
GRANT SELECT ON DBA_WM_SYS_PRIVS TO {your_user_name};
BEGIN
    DBMS_NETWORK_ACL_ADMIN.CREATE_ACL(
        acl => UPPER('{your_user_name}') || '_Connect_Access.xml',
        description => 'Connect Network',
        principal => UPPER('{your_user_name}'),
        is_grant => TRUE,
        privilege => 'resolve',
        start_date => NULL,
        end_date => NULL);

    DBMS_NETWORK_ACL_ADMIN.ASSIGN_ACL(
        acl => UPPER('{your_user_name}') || '_Connect_Access.xml',
        host => '*',
        lower_port => NULL,
        upper_port => NULL);
END;
```

PostgreSQL でのデータベースユーザーの作成

PostgreSQL ソースデータベースでデータベースユーザーを作成するには、次のスクリプトを使用します。

```
CREATE USER "{your_user_name}" WITH LOGIN PASSWORD '{your_password}';
GRANT pg_read_all_settings TO "{your_user_name}";

-- For PostgreSQL versions 10 and higher, add the following statement.
```

```
GRANT EXECUTE ON FUNCTION pg_ls_waldir() TO "{your_user_name}";
```

Microsoft SQL Server でのデータベースユーザーの作成

Microsoft SQL Server ソースデータベースでデータベースユーザーを作成するには、次のスクリプトを使用します。

```
USE master
GO

IF NOT EXISTS (SELECT * FROM sys.sql_logins WHERE name = N'{your_user_name}')

 CREATE LOGIN [{your_user_name}] WITH PASSWORD=N'{your_password}',
 DEFAULT_DATABASE=[master], DEFAULT_LANGUAGE=[us_english], CHECK_EXPIRATION=OFF,
 CHECK_POLICY=OFF

GO

GRANT VIEW SERVER STATE TO [{your_user_name}]

GRANT VIEW ANY DEFINITION TO [{your_user_name}]

GRANT VIEW ANY DATABASE TO [{your_user_name}]

IF LEFT(CONVERT(SYSNAME,SERVERPROPERTY('ProductVersion')), CHARINDEX('.',
 CONVERT(SYSNAME,SERVERPROPERTY('ProductVersion')), 0)-1) >= 12
 EXECUTE('GRANT CONNECT ANY DATABASE TO [{your_user_name}]')

DECLARE @dbname VARCHAR(100)
DECLARE @statement NVARCHAR(max)

DECLARE db_cursor CURSOR
LOCAL FAST_FORWARD
FOR
  SELECT
    name
  FROM MASTER.sys.databases
  WHERE state = 0
    AND is_read_only = 0
        OPEN db_cursor
FETCH NEXT FROM db_cursor INTO @dbname
    WHILE @@FETCH_STATUS = 0
BEGIN
```

```
  SELECT @statement = 'USE '+ quotename(@dbname) +';'+ '
    IF NOT EXISTS (SELECT * FROM sys.syslogins WHERE name = ''{your_user_name}'') OR
 NOT EXISTS (SELECT * FROM sys.sysusers WHERE name = ''{your_user_name}'')
   CREATE USER [{your_user_name}] FOR LOGIN [{your_user_name}];

  EXECUTE sp_addrolemember N''db_datareader'', [{your_user_name}]'

  BEGIN TRY
    EXECUTE sp_executesql @statement
  END TRY
  BEGIN CATCH
    DECLARE @err NVARCHAR(255)

    SET @err = error_message()

    PRINT @dbname
    PRINT @err
  END CATCH

  FETCH NEXT FROM db_cursor INTO @dbname
END
CLOSE db_cursor
  DEALLOCATE db_cursor

USE msdb
GO

GRANT EXECUTE ON dbo.agent_datetime TO [{your_user_name}]
```

データベースユーザーの削除

データ収集タスクをすべて完了したら、DMS データコレクター用に作成したデータベースユーザーを削除できます。次のスクリプトを使用して、最小限の権限を持つユーザーをデータベースから削除できます。

MySQL データベースからユーザーを削除するには、次のスクリプトを実行します。

```
DROP USER IF EXISTS "{your_user_name}";
```

Oracle データベースからユーザーを削除するには、次のスクリプトを実行します。

```
DECLARE
```

```
  -- Input parameters, please set correct value
  cnst$user_name CONSTANT VARCHAR2(255) DEFAULT '{your_user_name}';

  -- System variables, please, don't change
  var$is_exists INTEGER DEFAULT 0;
BEGIN
  SELECT COUNT(hal.acl) INTO var$is_exists
  FROM dba_host_acls hal
  WHERE hal.acl LIKE '%' || UPPER(cnst$user_name) || '_Connect_Access.xml';
  IF var$is_exists > 0 THEN
    DBMS_NETWORK_ACL_ADMIN.DROP_ACL(
      acl => UPPER(cnst$user_name) || '_Connect_Access.xml');
  END IF;
  SELECT COUNT(usr.username) INTO var$is_exists
  FROM all_users usr
  WHERE usr.username = UPPER(cnst$user_name);
  IF var$is_exists > 0 THEN
    EXECUTE IMMEDIATE 'DROP USER ' || cnst$user_name || ' CASCADE';
  END IF;
END;
```

PostgreSQL データベースからユーザーを削除するには、次のスクリプトを実行します。

```
DROP USER IF EXISTS "{your_user_name}";
```

SQL Server データベースからユーザーを削除するには、次のスクリプトを実行します。

```
USE msdb
GO

REVOKE EXECUTE ON dbo.agent_datetime TO [{your_user_name}]

USE master
GO

DECLARE @dbname VARCHAR(100)
DECLARE @statement NVARCHAR(max)

DECLARE db_cursor CURSOR
LOCAL FAST_FORWARD
FOR
SELECT
  name
```

```
FROM MASTER.sys.databases
WHERE state = 0
  AND is_read_only = 0
    OPEN db_cursor
FETCH NEXT FROM db_cursor INTO @dbname
  WHILE @@FETCH_STATUS = 0
BEGIN

SELECT @statement = 'USE '+ quotename(@dbname) +';'+ '
  EXECUTE sp_droprolemember N''db_datareader'', [{your_user_name}]

  IF EXISTS (SELECT * FROM sys.syslogins WHERE name = ''{your_user_name}'')
    OR EXISTS (SELECT * FROM sys.sysusers WHERE name = ''{your_user_name}'')
    DROP USER [{your_user_name}];'

BEGIN TRY
EXECUTE sp_executesql @statement
END TRY
BEGIN CATCH
  DECLARE @err NVARCHAR(255)

  SET @err = error_message()

  PRINT @dbname
  PRINT @err
END CATCH

FETCH NEXT FROM db_cursor INTO @dbname
END
CLOSE db_cursor
  DEALLOCATE db_cursor

GO

IF EXISTS (SELECT * FROM sys.sql_logins WHERE name = N'{your_user_name}')
  DROP LOGIN [{your_user_name}]  -- Use for SQL login

GO
```

データコレクターを使用した移行用データベースの検出

ソースデータインフラストラクチャを検出するには、AWS Application Discovery Service Agentless Collector または AWS DMS データコレクターを使用できます。ADS Agentless Collector は、サー

バープロファイル情報 (例: OS、CPU 数、RAM 量)、データベースメタデータ、使用率メトリクスなど、オンプレミス環境に関する情報をエージェントレス方式で収集するオンプレミスアプリケーションです。Agentless Collector は、Open Virtualization Archive (OVA) ファイルを使用して VMware vCenter Server 環境に仮想マシン (VM) としてインストールします。AWS DMS データコレクターは、ローカル環境にインストールする Windows アプリケーションです。このアプリケーションは、データ環境に接続してオンプレミスのデータベースと分析サーバーからメタデータとパフォーマンスメトリクスを収集します。ADS Agentless Collector または DMS データコレクターを通じてデータベースメタデータとパフォーマンスメトリクスが収集されると、DMS Fleet Advisor は、AWS クラウド に移行できるサーバー、データベース、スキーマのインベントリを構築します。

DMS データコレクターは、.NET ライブラリ、コネクタ、およびデータプロバイダーを使用して、データベースの検出とデータ収集のためにソースデータベースに接続する Windows アプリケーションです。

DMS データコレクターは Windows で実行されます。ただし、DMS データコレクターは、実行されている OS サーバーの種類を問わず、サポート対象のすべてのデータベースベンダーからデータを収集できます。

DMS データコレクターは TLS 暗号化で保護された RTPS プロトコルを使用して DMS Fleet Advisor との安全な接続を確立します。そのため、送信中のデータはデフォルトで暗号化されます。

AWS DMS では、AWS アカウント のために作成できるデータコレクター数の上限が定められています。AWS DMS サービスのクォータについては、以降のセクション「AWS Database Migration Service のクォータ」を参照してください。

トピック

- DMS データコレクターのアクセス許可
- AWS DMS Fleet Advisor のためのデータコレクターの作成
- データコレクターのインストールと設定
- モニタリングする OS サーバーとデータベースサーバーの検出
- モニタリング対象オブジェクトの管理
- AWS DMS Fleet Advisor での SSL の使用
- AWS DMS Fleet Advisor のデータ収集
- DMS データコレクターのトラブルシューティング

DMS データコレクターのアクセス許可

DMS データコレクターのために作成するデータベースユーザーには、読み取りアクセス許可が必要です。ただし、場合によっては、データベースユーザーに EXECUTE 権限が必要となることがあります。詳細については、「AWS DMS Fleet Advisor のデータベースユーザーの作成」を参照してください。

DMS データコレクターには、検出スクリプトを実行するために、追加のアクセス許可が必要です。

- OS を検出する場合、DMS データコレクターには、ドメインサーバーが LDAP プロトコルを使用してリクエストを実行するための認証情報が必要です。
- Linux でデータベースを検出する場合、DMS データコレクターには sudo SSH 権限が付与された認証情報が必要です。また、リモート SSH スクリプトの実行を許可するように Linux サーバーを設定する必要があります。
- Windows でデータベースを検出する場合は、DMS データコレクターが Windows Management Instrumentation (WMI) と WMI Query Language (WQL) クエリを実行してレジストリを読み取るための権限が付与された認証情報が必要です。また、リモート WMI、WQL、スクリプト PowerShell の実行を許可するように Windows サーバーを設定する必要があります。

AWS DMS Fleet Advisor のためのデータコレクターの作成

DMS データコレクターを作成してダウンロードする方法を説明します。

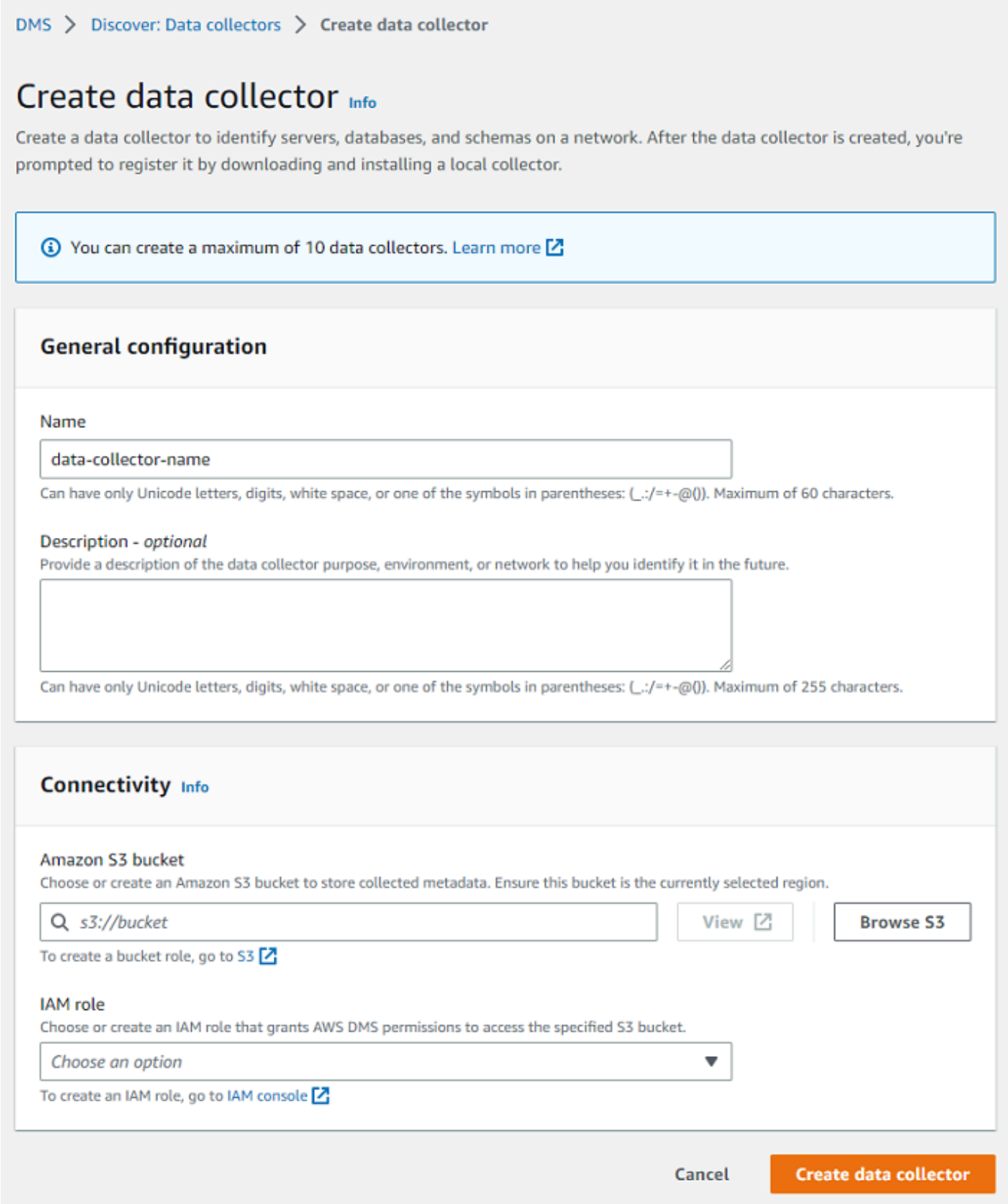
データコレクターを作成する前に、IAM コンソールで DMS Fleet Advisor のためにサービスにリンクされたロールを作成します。このロールにより、プリンシパルはメトリクスデータポイントを Amazon に発行できます CloudWatch。DMS Fleet Advisor はこのロールを使用してデータベースメトリクスを含むグラフを表示します。詳細については、「AWS DMS Fleet Advisor のサービスにリンクされたロールの作成」を参照してください。

データコレクターを作成してダウンロードするには

1. にサインインAWS Management Consoleし、https://console.aws.amazon.com/dms/v2/ で AWS DMSコンソールを開きます。

   DMS Fleet Advisor を使用するリージョンを選択します。
2. ナビゲーションペインで、[検出] の下にある [データコレクター] を選択します。[Data collectors] (データコレクター) ページが開きます。

3. [Create data collector] (データコレクターの作成) を選択します。[Create data collector] (データコレクターの作成) ページが開きます。



4. [一般的な設定] セクションの [名前] にデータコレクター名を入力します。
5. [Connectivity] (接続) セクションで、[Browse S3] (S3 を参照) を選択します。表示されたリストから事前設定した Amazon S3 バケットを選択します。

   AWS DMS は DMS Fleet Advisor インベントリのメタデータをこの S3 バケットに保存します。この Amazon S3 バケットが、AWS DMS Fleet Advisor が現在実行しているのと同じ AWS リージョン に配置されていることを確認します。

   > **Note**
   >
   > DMS Fleet Advisor は SSE-S3 暗号化バケットのみをサポートします。

6. IAM ロールのリストで表示されるリストから事前設定した IAM ロールを選択します。このロールは AWS DMS に指定した Amazon S3 バケットへのアクセスを許可します。
7. [Create data collector] (データコレクターの作成) を選択します。[データコレクター] ページが開き、作成したデータコレクターがリストに表示されます。

   初めてデータコレクターを作成する際、AWS DMS は、DMS Fleet Advisor で使用するためにデータをフォーマットして属性を保存する環境を Amazon S3 バケットに設定します。
8. 情報バナーの [ローカルコレクターをダウンロード] をクリックして、新たに作成したデータコレクターをダウンロードします。ダウンロード中であることを知らせるメッセージが表示されます。ダウンロードが完了すると、`AWS_DMS_Collector_Installer_`*`version_number`*`.msi` ファイルにアクセスできます。

これで、DMS データコレクターをクライアントにインストールできます。詳細については、「データコレクターのインストールと設定」を参照してください。

データコレクターのインストールと設定

DMS データコレクターのインストール方法、データ転送認証情報の指定方法、プロジェクトに LDAP サーバーを追加する方法について説明します。

次の表は、DMS データコレクターをインストールするためのハードウェア要件とソフトウェア要件について説明しています。

最小	推奨
プロセッサ：CPU ベンチマークスコアが 8,000 を超える 2 コア	プロセッサ：CPU ベンチマークスコアが 11,000 を超える 4 コア
RAM: 8 GB	RAM: 16 GB
ハードドライブ容量: 256 GB	ハードドライブ容量: 512 GB
オペレーティングシステム: Microsoft Windows Server 2012以降	オペレーティングシステム: Windows Server 2016以降

ネットワーク上のクライアントにデータコレクターをインストールするには

1. .MSI インストーラを実行します。AWS DMS Fleet Advisor コレクタ セットアップ ウィザードページが表示されます。
2. [Next] (次へ) を選択します。[End-user license agreement] (エンドユーザーライセンス契約) が表示されます。
3. [End-user license agreement] (エンドユーザーライセンス契約) を読み、同意します。
4. [次へ] をクリックします。[Destination folder] (送信先フォルダ) ページが表示されます。
5. [次へ] をクリックして、データコレクターをデフォルトディレクトリにインストールします。

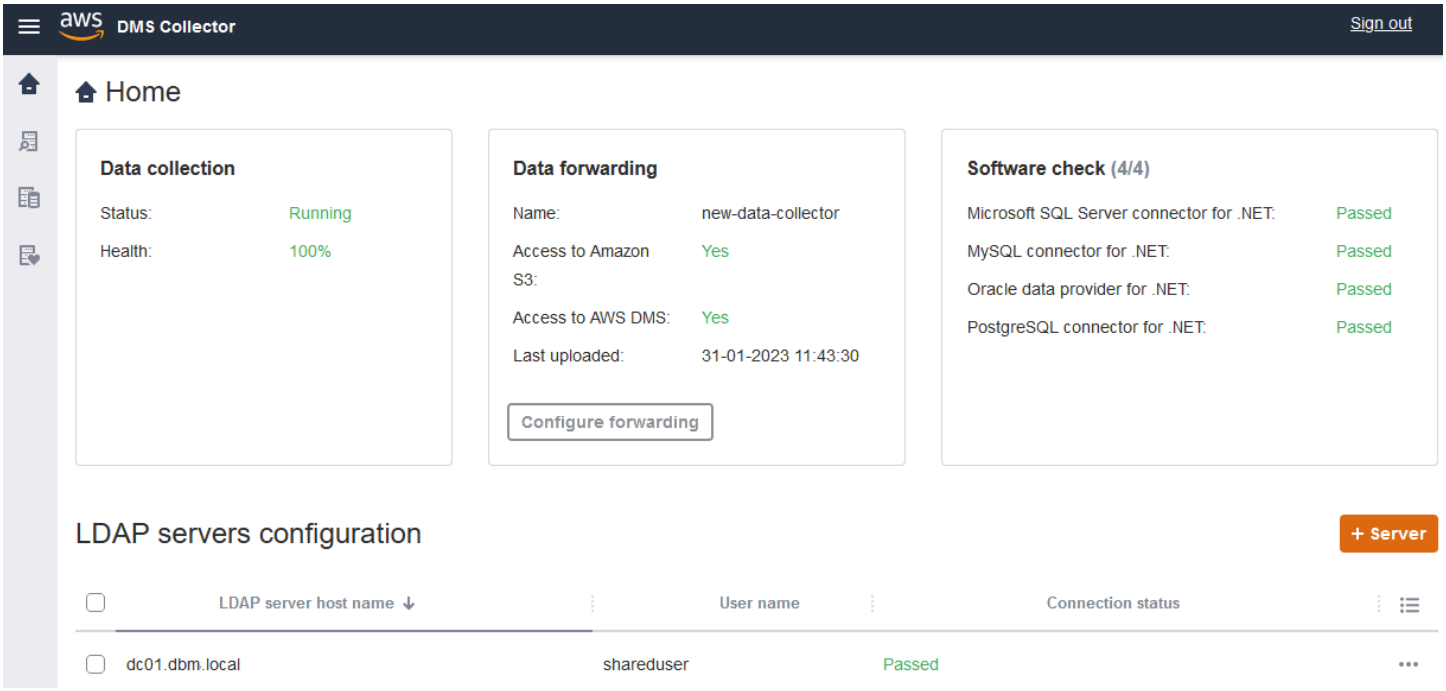
   または[Change] (変更) をクリックして、別のインストールディレクトリを入力します。次いで、[Next (次へ)] を選択します。
6. [Desktop shortcut] (デスクトップショートカット) ページで、デスクトップにアイコンをインストールするためのボックスを選択します。
7. [Install] (インストール) を選択します。データコレクターが選択したディレクトリにインストールされます。
8. [Completed DMS Collector Setup Wizard] ページで、[AWS DMS コレクターを起動] をクリックしてから、[終了] をクリックします。

DMS データコレクターは、.NET ライブラリ、コネクタ、データプロバイダーを使用してソースデータベースに接続します。DMS データコレクターインストーラは、サーバー上のサポート対象のすべてのデータベースに必要なソフトウェアを自動的にインストールします。

インストール後は、アドレスに **http://localhost:11000/** を入力してブラウザからデータコレクターを実行できます。または、Microsoft Windows の [スタート] メニューのプログラムリストで [AWS DMS コレクター] を選択して実行することもできます。DMS データコレクターを初めて実行する際は、認証情報を設定するように求められます。データコレクターにサインインするためのユーザー名とパスワードを作成します。

DMS データコレクターのホームページには、次のステータス条件など、メタデータ収集の準備と実行に関する情報が表示されます。

- データ収集のステータスと健全性。
- データコレクターがデータを AWS DMS に転送するための Amazon S3 バケットと AWS DMS へのアクセシビリティ
- インストールされているデータベース ドライバへのコネクティビティ。
- 初期ディスカバリを実行するための LDAP サーバーの認証情報。



DMS データコレクターは LDAP ディレクトリを使用して、ネットワーク内のマシンとデータベースサーバーに関する情報を収集します。Lightweight Directory Access Protocol (LDAP) は、オープン標準のアプリケーションプロトコルです。これは、IP ネットワーク経由で分散ディレクトリ情報サービスにアクセスして維持するために使用されます。システムのインフラストラクチャに関する情報を検出するために、既存の LDAP サーバーをデータコレクターのプロジェクトに追加できます。そのためには、[+サーバー] オプションをクリックして、ドメインコントローラーの完全修飾ドメイン名

(FQDN) と認証情報を指定します。サーバーを追加した後、接続チェックで検証します。検出プロセスの開始方法については、「モニタリングする OS サーバーとデータベースサーバーの検出」を参照してください。

データ転送のための認証情報の設定

データコレクターのインストール後、このアプリケーションが収集したデータを AWS DMS Fleet Advisor に送信できることを確認します。

データ転送のための認証情報を AWS DMS Fleet Advisor で設定するには

1. DMS データコレクターのホームページの [Data forwarding] セクションで、[Configure forwarding] をクリックします。[Configure credentials for data forwarding] ダイアログボックスが開きます。
2. DMS Fleet Advisor を使用する AWS リージョン を選択します。
3. 前のタスクで IAM リソースを作成した際に取得した AWS アクセスキー ID と AWS シークレットアクセスキー を入力します。詳細については、「IAM リソースを作成する」を参照してください。
4. [Browse data collectors] をクリックします。

   指定したリージョンでデータコレクターをまだ作成していない場合は、先に進む前にデータコレクターを作成します。詳細については、「データコレクターの作成」を参照してください。
5. [Choose data collector] ウィンドウのリストでデータコレクターを選択して、[選択] をクリックします。
6. [Configure credentials for data forwarding] ダイアログボックスで、[保存] をクリックします。

[DMS コレクター] ホームページの [Data forwarding] カードで、[Access to Amazon S3] と [Access to AWS DMS] のステータスが [はい] であることを確認します。

[Access to Amazon S3] または [Access to AWS DMS] のステータスが [いいえ] と表示されている場合は、Amazon S3 と DMS Fleet Advisor にアクセスする IAM リソースが作成済みであるかを確認します。必要なアクセス許可すべてを付与したこれらの IAM リソースを作成した後、もう一度データ転送を設定します。詳細については、「IAM リソースを作成する」を参照してください。

モニタリングする OS サーバーとデータベースサーバーの検出

DMS データコレクターを使用して、ネットワーク内のすべての利用可能なサーバーを検索して一覧表示できます。ネットワーク内の利用可能なデータベースサーバーをすべて検出することをお勧めし

ますが、必須ではありません。さらにデータを収集するために、必要に応じて、サーバーを手動で追加したり、サーバーのリストをアップロードしたりすることもできます。サーバーのリストを手動で追加する方法の詳細については、「モニタリング対象オブジェクトの管理」を参照してください。

これらのサーバー上のデータベースを検出する前に、すべてのオペレーティングシステム (OS) サーバーを検出することをお勧めします。OS サーバーを検出するには、リモート PowerShell、Secure Shell (SSH)、Windows Management Instrumentation (WMI) スクリプトとコマンドを実行するアクセス許可、および Windows レジストリへのアクセス許可が必要です。ネットワーク内のデータベースサーバーを検出して、データベースサーバーからメタデータを収集するには、リモートデータベース接続に対する読み取り専用の管理者権限が必要です。検出を続行する前に、LDAP サーバーの追加が完了していることを確認します。詳細については、「データ転送のための認証情報の設定」を参照してください。

DMS データコレクターの使用を開始するには、次のタスクを実行します。

- ネットワーク内のすべての OS サーバーを検出する。
- 特定の OS サーバーをモニタリング対象のオブジェクトとして追加する。
- モニタリング対象の OS サーバーの接続を確認します。
- OS サーバー上で実行されている Microsoft SQL Server、MySQL、Oracle、PostgreSQL データベースを検出する。
- データ収集するデータベースサーバーを追加する。
- モニタリング対象のデータベースへの接続を確認します。

モニタリングできるネットワーク内の OS サーバーを検出するには

1. DMS データコレクターのナビゲーションペインで、[検出] を選択します。ナビゲーション ペインを表示するには、DMS データコレクターのホームページの左上隅にあるメニューアイコンをクリックします。

   [Discovery] (検出) ページが開きます。
2. [OS サーバー] タブが選択されていることを確認して、[検出の実行] をクリ九します。[Discovery parameters] (検出パラメータ) ダイアログボックスが表示されます。
3. ネットワークのスキャンに使用する LDAP サーバーを入力します。
4. [Run discovery] (検出の実行) を選択します。このページには、データベースの実行状況を問わず、ネットワーク内で検出されたすべての OS サーバーのリストが表示されます。

オペレーティングシステム (OS) サーバー上のデータベースの検出を実行する前に、すべてのオペレーティング システム (OS) サーバーの検出を実行しておくことをお勧めします。認証情報を使用することで、まずホストサーバーの検出が可能になり、次にホストサーバーに配置されたデータベースの検出が可能になるため、OS サーバー上のデータベースの検出を実行する前に、まず OS サーバーを検出します。LDAP サーバーがネットワーク内の OS サーバーを検索するために使用する認証情報は、特定の OS サーバー上のデータベースの検出に必要な認証情報とは異なる場合があることに注意します。このため、モニタリング対象オブジェクトに OS サーバーを追加して、認証情報を確認して必要に応じて修正し、接続を確認してから次に進むことをお勧めします。

ネットワーク内で検出された OS サーバーのリストで、モニタリング対象オブジェクトに追加するサーバーを選択できるようになりました。

モニタリングするオブジェクトとして OS サーバーを選択するには

1. [Discovery] (検出) ページで、[OS servers] (OS サーバー) タブを選択します。
2. 画面の検出された OS サーバーのリストで、モニタリングする各サーバーの横にあるチェックボックスをオンにします。
3. [Add to monitored objects] (モニタリング対象オブジェクトへの追加) を選択します。

モニタリングして接続を検証する OS サーバーのリストは、[Monitor objects] ページで確認できます。

モニタリングする選択した OS サーバーの接続を確認するには

1. DMS データコレクターのナビゲーションペインで、[Monitored objects] を選択します。
2. [Monitored objects] ページで、[OS サーバー] タブをクリックします。監視対象の検出された OS サーバのリストが表示されます。
3. 列の上部にあるチェックボックスをオンにして、リストされているすべての OS サーバーを選択します。
4. [アクション] をクリックして、[Verify connection] を選択します。サーバーオブジェクトごとに、[Connections status] の結果を確認します。
5. 接続ステータスが 成功 以外のサーバーを選択します。次に、[アクション] をクリックして、[編集] を選択します。[Edit server] ダイアログボックスが開きます。

6. 情報が正しいことを確認するか、必要に応じて編集します。完了したら、[Save] (保存) を選択します。[Override credentials] (認証情報の) ダイアログボックスが開きます。
7. [Overwrite] (上書き) を選択します。DMS データコレクターが各接続のステータスを検証して、成功 として更新します。

これでモニタリング対象として選択したサーバーにあるデータベースを検出できます。

サーバー上で実行されているデータベースを検出する

1. DMS データコレクターのナビゲーションペインで、[検出] を選択します。
2. [Database servers] (データベースサーバー) タブをクリックし、[Run discovery] (検出の実行) を選択します。[検出パラメータ] ダイアログボックスが開きます。
3. [検出パラメータ] ダイアログボックスの [Discovery by] で、[Monitored objects] を選択します。[サーバー] では、データベースの検出を実行する OS サーバーを選択します。
4. [Run discovery] (検出の実行) を選択します。このページには、モニタリング対象として選択した OS サーバー上のすべてのデータベースのリストが表示されます。

監視するデータベースの選択に役立つデータベースアドレス、サーバー名、データベース エンジンなどの情報を表示します。

監視するデータベースを選択するには

1. [検出] ページで、[データベースサーバー] タブをクリックします。
2. 画面の検出されたデータベースのリストで、モニタリングするすべてのデータベースの横にあるチェックボックスをオンにします。
3. [Add to monitored objects] (モニタリング対象オブジェクトへの追加) を選択します。

これでモニタリング対象として選択したデータベースへの接続を検証できるようになります。

モニタリング対象データベースへの接続を検証するには

1. DMS データコレクターのナビゲーションペインで、[Monitored objects] を選択します。
2. [Monitored objects] ページで、[データベースサーバー] タブをクリックします。検出されたモニタリング対象のデータベースサーバーのリストが表示されます。
3. 列の上部にあるチェックボックスをオンにして、リストされているすべてのデータベースサーバーを選択します。

4. [アクション] をクリックして、[Verify connection] を選択します。データベースごとに、[Connections status] の結果を確認します。
5. ステータスが未定義 (空白) または [失敗] の接続を選択します。次に、[アクション] をクリックして、[編集] を選択します。[Edit monitored objects] (モニタリング対象オブジェクトの編集) ダイアログボックスが開きます。
6. [ログイン] と [パスワード] の認証情報を入力して、[保存] をクリックします。[Change credentials] ダイアログボックスが開きます。
7. [Overwrite] (上書き) を選択します。DMS データコレクターが各接続のステータスを検証して、成功 として更新します。

モニタリング対象 OS サーバーおよびデータベースを検出した後、モニタリング対象オブジェクトを管理するためのアクションを実行することもできます。

モニタリング対象オブジェクトの管理

OS サーバーとデータベースサーバーの検出 の説明に従って、サーバ検出プロセスを実行するときにモニタリングするオブジェクトを選択できます。オペレーティングシステム (OS) サーバーとデータベースサーバーなどのモニタリング対象オブジェクトは、手動でも管理できます。モニタリング対象オブジェクトの管理として、次のアクションを実行できます。

- 新しいモニタリング対象オブジェクトを追加する。
- 既存のオブジェクトを削除する。
- 既存のオブジェクトを編集する。
- モニタリング対象オブジェクトのリストをエクスポートまたはインポートする。
- オブジェクトへの接続をチェックする。
- データ収集を開始する。

たとえば、モニタリング対象のオブジェクトを手動で追加できます。

モニタリングするオブジェクトを手動で追加するには

1. [Monitored objects] ページで、[+サーバー] をクリックします。[Add monitored object] ダイアログボックスが開きます。
2. サーバーに関する情報を追加して、[保存] をクリックします。

また、.csv ファイルを使用して、モニタリングするオブジェクトの大きなリストをインポートすることもできます。オブジェクトのリストを DMS データコレクターにインポートするには、次のとおりの .csv ファイル形式を使用します。

```
Hostname - Hostname or IP address of Monitored Object
Port - TCP port of Monitored Object
Engine: (one of the following)
        • Microsoft SQL Server
        • Microsoft Windows
        • Oracle Database
        • Linux
        • MySQL Server
        • PostgreSQL
Connection type: (one of the following)
        • Login/Password Authentication
        • Windows Authentication
        • Key-Based Authentication
Domain name:(Windows authentication)
        • Use domain name for the account
User name
Password
```

モニタリングするオブジェクトのリストを含む .csv ファイルをインポートするには

1. [Import] (インポート) を選択します。[Import monitored objects] ページが開きます。
2. インポートする .csv ファイルを参照して、[次へ] をクリックします。

すべてのオブジェクトを表示して、メタデータの収集を開始するオブジェクトを選択できます。

OS サーバーと手動で追加したデータベースの関連付け

DMS Fleet Advisor は、MySQL データベースや PostgreSQL データベースからパフォーマンスメトリクスを直接収集できません。DMS Fleet Advisor は、ターゲットレコメンデーションに必要なメトリクスを収集するために、データベースが実行されている OS メトリクスを利用します。

MySQL データベースと PostgreSQL データベースをモニタリング対象オブジェクトのリストに手動で追加しても、DMS データコレクターはデータベースが実行されている OS サーバーを識別できません。このような問題があるため、MySQL データベースと PostgreSQL データベースは OS サーバーに関連付ける必要があります。

DMS Fleet Advisor が自動的に検出したデータベースについては、OS サーバーを手動で関連付ける必要はありません。

OS サーバーをデータベースに関連付けるには

1. DMS データコレクターのナビゲーションペインで、[Monitored objects] を選択します。
2. [Monitored objects] ページで、[データベースサーバー] タブをクリックします。データベースサーバーのリストが表示されます。
3. 手動で追加した MySQL または PostgreSQL データベースサーバーの横にあるチェックボックスをオンにします。
4. [アクション] をクリックして、[編集] を選択します。[Edit database] ダイアログボックスが開きます。
5. DMS データコレクターが、このデータベースが実行されている OS サーバーをすでに検出している場合は、[Auto detect] をクリックします。DMS データコレクターは SQL スクリプトを実行して、データベースが実行されている OS サーバーを自動的に識別します。その後、DMS データコレクターはこの OS サーバーをデータベースに関連付けます。次の手順をスキップして、編集したデータベース設定を保存します。

   DMS データコレクターがデータベースの OS サーバーを自動的に識別できない場合は、適切な認証情報を使用して、データベースへのアクセス許可を指定していることを確認します。OS サーバーは、必要に応じて手動で追加できます。
6. OS サーバーを手動で追加するには、[+Add OS server] をクリックします。[Add host OS server] ダイアログボックスが開きます。

   OS サーバーに関する情報を追加して、[保存] をクリックします。
7. [データベースを編集] ダイアログボックスで、[Verify connection] をクリックして、DMS データコレクターが OS サーバーに接続できることを確認します。
8. 接続を検証したら、[保存] をクリックします。

ソースデータベースに関連付けられている OS サーバーを変更すると、DMS Fleet Advisor は更新されたメトリクスを使用してレコメンデーションを生成します。ただし、Amazon CloudWatch チャートにはデータベースサーバーの古いデータが表示されます。 CloudWatch グラフの詳細については、「」を参照してくださいレコメンデーションの詳細。

AWS DMS Fleet Advisor での SSL の使用

データを保護するために、AWS DMS Fleet Advisor はデータベースアクセスに SSL を使用できます。

サポートされているデータベース

AWS DMS Fleet Advisor は、次のデータベースへの SSL を使用したアクセスをサポートしています。

- Microsoft SQL Server
- MySQL
- PostgreSQL

SSL の設定

SSL を使用してデータベースにアクセスするには、SSL をサポートするようにデータベースサーバーを設定します。詳細については、データベースに関する次のドキュメントを参照してください。

- SQL Server: Enable encrypted connections to the Database Engine
- MySQL: Configuring MySQL to Use Encrypted Connections
- PostgreSQL: Secure TCP/IP Connections with SSL

SSL を使用してデータベースに接続するには、手動でサーバーを追加する際に、[Trust server certificate] と [Use SSL] を選択します。MySQL データベースの場合はカスタム証明書を使用できます。カスタム証明書を使用するには、[CA を確認] チェックボックスをオンにします。サーバーの追加の詳細については、「モニタリング対象オブジェクトの管理」を参照してください。

SQL Server のサーバー認証局 (CA) 証明書の確認

SQL Server のサーバー認証局 (CA) 証明書を検証する場合は、サーバーを追加する際に [Trust server certificate] の選択を解除します。サーバーが既知の CA を使用しており、その CA がデフォルトで OS にインストールされている場合、検証は正常に機能するはずです。DMS Fleet Advisor がデータベースサーバーに接続できない場合は、データベースサーバーが使用する CA 証明書をインストールします。詳細については、「クライアントの設定」を参照してください。

AWS DMS Fleet Advisor のデータ収集

データの収集を開始するには、[Monitored objects] ページでオブジェクトを選択して、[Run data collection] をクリックします。DMS データコレクターは、一度に最大 100 のデータベースから収集できます。また、DMS データコレクターは、最大 8 つの並列スレッドを使用して環境内のデータベースに接続できます。DMS データコレクターは、このような 8 つのスレッドのうちの最大 5 つの並列スレッドを使用して、単一のデータベースインスタンスに接続できます。

> **Important**
>
> データの収集を開始する前に、DMS データコレクターのホームページの [Software check] セクションを確認します。モニタリング対象のすべてのデータベースエンジンのステータスが 合格 であることを確認します。ステータスが 失敗 のデータベースエンジンがあり、モニタリング対象オブジェクトのリストに対応するエンジンを備えたデータベースサーバーがある場合は、続行する前に問題を解決します。[Software check] セクションにリストされている 失敗 ステータスの横でヒントを確認できます。

DMS データコレクターは、単一実行モードと継続モニタリングモードの 2 つのモードで動作できます。データ収集を開始すると、[Run data collection] ダイアログボックスが開きます。次のいずれかのオプションを選択します。

メタデータとデータベースのキャパシティ

DMS データコレクターは、データベースまたは OS サーバーから情報を収集します。収集する情報には、スキーマ、バージョン、エディション、CPU、メモリ、ディスク容量などがあります。DMS データコレクターは、IOPS、I/O スループット、アクティブなデータベースサーバー接続などのメトリクスも収集して提供します。この情報に基づいて、DMS Fleet Advisor でターゲットレコメンデーションを算出できます。ソースデータベースのプロビジョニングが過剰または不足している場合、ターゲットレコメンデーションも過剰プロビジョニングまたは過小プロビジョニングとなります。

これがデフォルトのオプションです。

メタデータ、データベース容量、リソース使用率

DMS データコレクターは、メタデータとデータベースのキャパシティ情報に加えて、データベースまたは OS サーバーの CPU、メモリ、ディスク容量の実際の使用率のメトリクスを収集します。DMS データコレクターは、IOPS、I/O スループット、アクティブなデータベースサーバー接

続などのメトリクスも収集して提供します。提供されるターゲットレコメンデーションは、実際のデータベースのワークロードに基づいているため、より正確となります。

このオプションを選択した場合は、データ収集期間を設定します。データ収集の期間に [Next 7 days] を選択したり、1～60 日の範囲の [カスタム範囲] を設定したりできます。

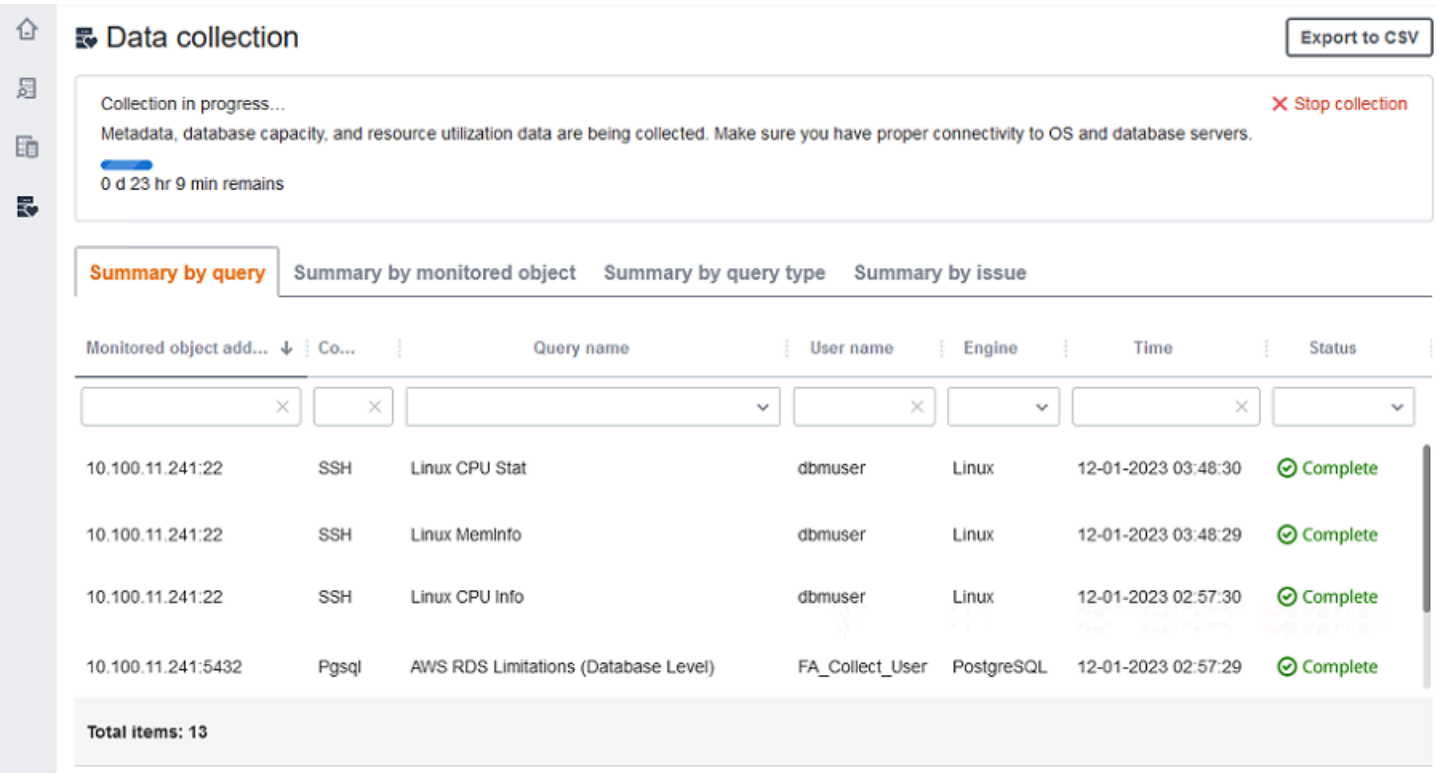
データ収集が開始されると、[データ収集] ページにリダイレクトされます。このページでは、収集クエリがどのように実行されているかを確認して、進捗をライブモニタリングできます。収集全体のヘルスは、このページまたは DMS データコレクターのホームページで確認できます。全体的なデータ収集の健全性が 100% 未満の場合は、収集関連の問題を修正する必要がある場合があります。

DMS データコレクターを [Metadata and database capacity] モードで実行する場合は、完了したクエリ数を [データ収集] ページで確認できます。

DMS データコレクターを [Metadata, database capacity, and resource utilization] モードで実行する場合は、DMS データコレクターがモニタリングを完了するまでの残り時間を確認できます。

[Data collection] (データ収集) ページでは、オブジェクトの収集ステータスを確認できます。正常に動作していない場合は、発生した問題の数を示すメッセージが表示されます。問題の修正を特定しやすくするために、詳細をチェックできます。次のタブには潜在的な問題の一覧が表示されます：

- [Summary by query] (クエリによるサマリ) — Ping テストのようなテストのステータスを表示します。結果をフィルタリングするには、[Status] (ステータス) 列でロードバランサーの ID をクリックします。[Status] (ステータス) 列にはデータ収集中に発生した障害の数を示すメッセージが表示されます。
- [Summary by a monitored object] (監視対象オブジェクト別のサマリー) — オブジェクトごとの全体的なステータスを表示します。
- [Summary by query type] – SQL、Secure Shell (SSH)、Windows Management Instrumentation (WMI) コールなどの収集クエリタイプのステータスを表示します。
- [Summary by issue] (問題別要約) - 発生した一意の課題をすべて表示し、課題名と各課題が発生した回数を表示します。



収集結果をエクスポートするには、[Export to CSV] (CSV へエクスポート)。

問題を特定して解決したら、[Start collection] (収集の開始) をクリックし、データ収集プロセスを再実行します。データ収集を実行した後、データコレクターはセキュアな接続を使用して収集したデータを DMS Fleet Advisor インベントリにアップロードします。DMS Fleet Advisor は情報を Amazon S3 バケットに保存します。データ転送の認証情報の設定については、「データ転送のための認証情報の設定」を参照してください。

AWS DMS Fleet Advisor でのキャパシティとリソース使用率のメトリクスの収集

メタデータとパフォーマンスメトリクスは、単一実行と継続的モニタリングの 2 つのモードで収集できます。選択したオプションに応じて、DMS データコレクターはデータ環境内のさまざまなメトリクスを追跡します。単一実行中、DMS データコレクターはデータベースと OS サーバーからのメタデータメトリクスのみを追跡します。継続的モニタリング中、DMS データコレクターはリソースの実際の使用率を追跡します。

AWS DMS は、DMS データコレクターの 1 回の実行中に次のメタデータとメトリクスを収集します。

- OS サーバーの使用可能なメモリ
- OS サーバーの使用可能なストレージ

- データベースのバージョンとエディション
- OS サーバー上の CPU 数
- スキーマの数
- ストアドプロシージャ数
- テーブルの数
- トリガー数
- ビュー数
- スキーマ構造

DMS Fleet Advisor は、上記のメトリクスを使用してデータベースと OS サーバーのインベントリを構築します。また、DMS Fleet Advisor はこれらのメタデータとメトリクスを使用してソースデータベーススキーマを分析します。

DMS Fleet Advisor は、データコレクターの 1 回の実行中に収集されたメトリクスを使用して、ターゲットレコメンデーションを生成できます。ただし、過剰にプロビジョニングされたソースデータベースの場合、ターゲットレコメンデーションも過剰にプロビジョニングされます。したがって、のリソースのメンテナンスには追加コストが発生しますAWS クラウド。ソースデータベースがプロビジョニング不足の場合、ターゲットレコメンデーションもプロビジョニング不足となり、パフォーマンス上の問題が発生する可能性があります。DMS データコレクターのメタデータ、データベース容量、およびリソース使用率モードを選択して、継続的なモニタリングを使用してデータを収集することをお勧めします。

継続的モニタリング中、AWS DMS は次のメトリクスを収集します。DMS データコレクターは 1 ~ 60 日間の範囲で実行できます。

- データベースサーバーの I/O スループット
- データベースサーバーの 1 秒あたりの入出力オペレーション (IOPS)
- OS サーバーが使用する CPU の数
- OS サーバーのメモリ使用状況
- アクティブなデータベースと OS サーバー接続の数

ターゲットデータベースがパフォーマンスニーズを満たすように、DMS Fleet Advisor は上記のメトリクスを使用して、正確なターゲットレコメンデーションを生成します。これにより、 内のリソースのメンテナンスで発生する追加コストを防ぐことができますAWS クラウド。

AWS DMS Fleet Advisor がキャパシティとリソース使用率のメトリクスを収集するには

DMS Fleet Advisor は 1 分ごとにパフォーマンスメトリクスを収集します。

Oracle と SQL Server の場合、DMS Fleet Advisor は SQL クエリを実行して、各データベースメトリクスの値をキャプチャします。

MySQL と PostgreSQL の場合、DMS Fleet Advisor は、データベースが実行されている OS サーバーからパフォーマンスメトリクスを収集します。Windows では、DMS Fleet Advisor は WMI Query Language (WQL) スクリプトを実行して、WMI データを受信します。Linux では、DMS Fleet Advisor は OS サーバーのメトリクスをキャプチャするコマンドを実行します。

> **Important**
>
> リモート SQL スクリプトを実行すると、本番稼働用データベースのパフォーマンスに影響が及ぶ可能性があります。ただし、データ収集クエリには計算ロジックは含まれていないため、データ収集プロセスでデータベースリソースの 1% 以上が使用されることはほとんどありません。

データコレクターがメトリクスを収集する際に実行するクエリはすべて確認できます。確認するには、`DMSCollector.Collections.json` ファイルを開きます。このファイルは、データコレクターをインストールしたのと同じフォルダ内の `etc` フォルダにあります。デフォルトのパスは `C:\ProgramData\Amazon\AWS DMS Collector\etc\DMSCollector.Collections.json` です。

DMS データコレクターは、収集したすべてのデータの一時ストレージとしてローカルファイルシステムを使用します。DMS データコレクターは、収集したデータを JSON 形式で保存します。データ転送を設定する前に、ローカルコレクターをオフラインモードで使用して、収集したファイルを手動で確認または検証できます。収集したすべてのファイルは、DMS データコレクターをインストールしたのと同じフォルダ内の `out` フォルダで確認できます。デフォルトのパスは `C:\ProgramData\Amazon\AWS DMS Collector\out` です。

> **Important**
>
> DMS データコレクターをオフラインモードで実行し、収集したデータをサーバーに 14 日以上保存する場合、Amazon を使用してこれらのメトリクス CloudWatch を表示することはできません。ただし、DMS Fleet Advisor は引き続きこのデータを使用してレコメンデーショ

ンを生成します。 CloudWatch グラフの詳細については、「」を参照してくださいレコメンデーションの詳細。

収集したデータファイルは、オンラインモードでも確認または検証できます。DMS データコレクターは、DMS データコレクター設定で指定した Amazon S3 バケットにすべてのデータを転送します。

DMS データコレクターを使用して、オンプレミスのデータベースからデータを収集できます。Amazon RDS データベースと Aurora データベースからもデータを収集することができます。ただし、Amazon RDS または Aurora とオンプレミスの DB インスタンスの違いにより、すべての DMS データコレクタークエリがクラウドで正常に実行できるとは限りません。DMS データコレクターは MySQL データベースと PostgreSQL データベースの使用状況メトリクスをホスト OS から収集するため、この方法は Amazon RDS と Aurora では機能しません。

DMS データコレクターのトラブルシューティング

次のリストは、データコレクターでのデータ収集中に特定の問題が発生した場合に実行できるアクションを提供しています。

トピック

- ネットワークおよびサーバー接続に関連するデータ収集の問題
- Windows 管理インストルメンテーションに関連するデータ収集の問題
- Windows のウェブページコンポーザに関連するデータ収集の問題
- SSL に関連するデータ収集の問題

ネットワークおよびサーバー接続に関連するデータ収集の問題

NET: ping 要求中に例外が発生しました。

コンピュータの名前を調べて、IP アドレスに解決できない状態にあるかどうかを確認します。

たとえば、コンピュータの電源がオフになっているか、ネットワークから切断されているか、または使用停止されているかどうかを確認します。

NET: タイムアウト

受信ファイアウォールルール [File and Printer Sharing (Echo Request - ICMPv4-In)] (ファイルとプリンターの共有 (Echo Request-ICMPv4-In))をオンにします。例:

* Inbound ICMPv4

NET: DestinationHostUnreachable

コンピュータの IP アドレスをチェックします。具体的には、DMS データコレクターを実行しているコンピュータと同じサブネット上にあるか、アドレス解決プロトコル (ARP) リクエストに応答するかを確認します。

コンピュータが別のサブネット上にある場合、ゲートウェイの IP アドレスをメディア アクセス コントロール (MAC) アドレスに対して解決できません。

また、コンピュータの電源が切れているか、ネットワークから切断されているか、または使用停止されているかどうかをチェックします。

Windows 管理インストルメンテーションに関連するデータ収集の問題

WMI: RPC サーバーは使用できません。(HRESULTからの例外:0x800706ba)

受信ファイアウォール ルール「Windows 管理インストルメンテーション (DCOM—In)」をオンにします。例:

* Inbound TCP/IP at local port 135.

また、受信ファイアウォール ルール「Windows 管理インストルメンテーション (WMI-In)」をオンにします。例:

Windows Server 2008 以降のバージョンについては* Inbound TCP/IP at local port 49152 – 65535。

* Inbound TCP/IP at local port 1025 – 5000 (Windows Server 2003 以前のバージョンの場合)

WMI: アクセスが拒否されました。(HRESULTからの例外:0x80070005 (E_ACCESSDENED))

次の操作を試してください：

- DMS データコレクターのユーザーを Windows グループ、Distributed COM Users、または Administrators に追加する。
- Windows 管理インストルメンテーションサービスを開始し、スタートアップの種類を [自動] に設定します。
- DMS dデータコレクターのユーザー名が \ 形式であることを確認する。

WMI: アクセス拒否

ルート WMI 名前空間の DMS データコレクターユーザーに「リモートの有効化」アクセス許可を追加します。

詳細設定を使用し、権限が[This namespace and subnamespaces] (この名前空間とサブ名前空間)に適用されていることを確認します。

WMI: コールはメッセージフィルタによってキャンセルされました。(HRESULTからの例外:0x80010002...)

Windows 管理インストルメンテーションサービスを再起動します。

Windows のウェブページコンポーザに関連するデータ収集の問題

WPC: ネットワークパスが見つかりませんでした

受信ファイアウォールルール「ファイルとプリンタの共有 (SMB—In)」をオンにします。例:

```
* Inbound TCP/IP at local port 445.
```

また、リモート レジストリサービスを起動し、起動タイプを [自動] に設定します。

アクセスが拒否されました

DMS データコレクターのユーザーを Performance Monitor Users または管理者グループに追加します。

WPC: カテゴリが存在しません

`loader /r` を実行して、パフォーマンス カウンター キャッシュを再構築し、コンピュータを再起動します。

> **Note**
>
> AWS Database Migration Service (AWS DMS) を使用してデータを移行する際の問題のトラブルシューティングについては、「Troubleshooting and diagnostic support」を参照してください。

SSL に関連するデータ収集の問題

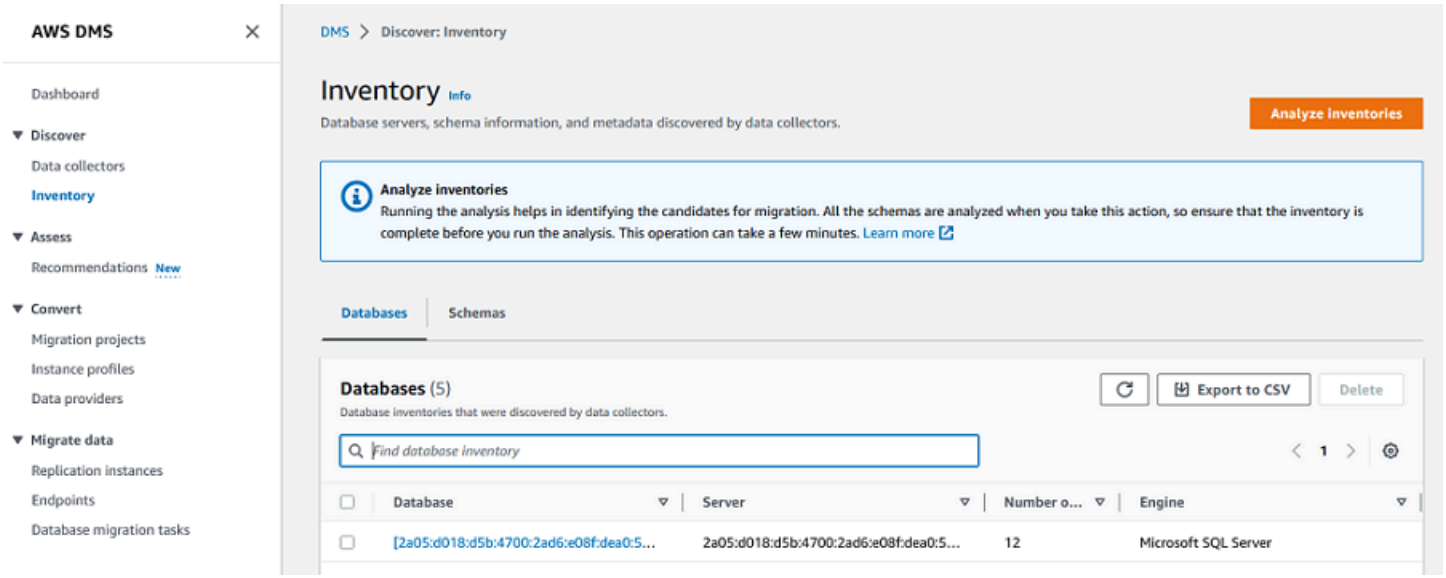
SSL エラー

> データベースには安全な SSL 接続が必要です。現在、接続の [CA を確認] と [Use SSL] オプションが有効になっていません。上記のオプションを有効にして、データベースが使用する証明書がローカル OS にインストールされていることを確認します。詳細については、「SSL の設定」を参照してください。

AWS DMS Fleet Advisor での分析のためのインベントリの使用

データベースの移行の可能性をチェックするために、検出されたデータベースとスキーマのインベントリを操作できます。これらのインベントリの情報を使用して、移行に適したデータベースとスキーマを把握できます。

データベースとスキーマのインベントリにはコンソールからアクセスできます。アクセスするには、コンソールで [インベントリ] を選択します。



DMS Fleet Advisor はデータベーススキーマを分析して、さまざまなスキーマの類似性を判断します。この分析では、オブジェクトの実際のコードは比較しません。DMS Fleet Advisor は、関数やプロシージャなどのスキーマオブジェクト名のみを比較して、異なるデータベーススキーマ内の類似オブジェクトを識別します。

トピック

- 分析にデータベース インベントリを使用する

- 分析にスキーマインベントリを使用する

分析にデータベース インベントリを使用する

データの収集元であるネットワーク内で検出されたすべてのサーバー上のすべてのデータベースのリストを表示するには、次の手順に従います。

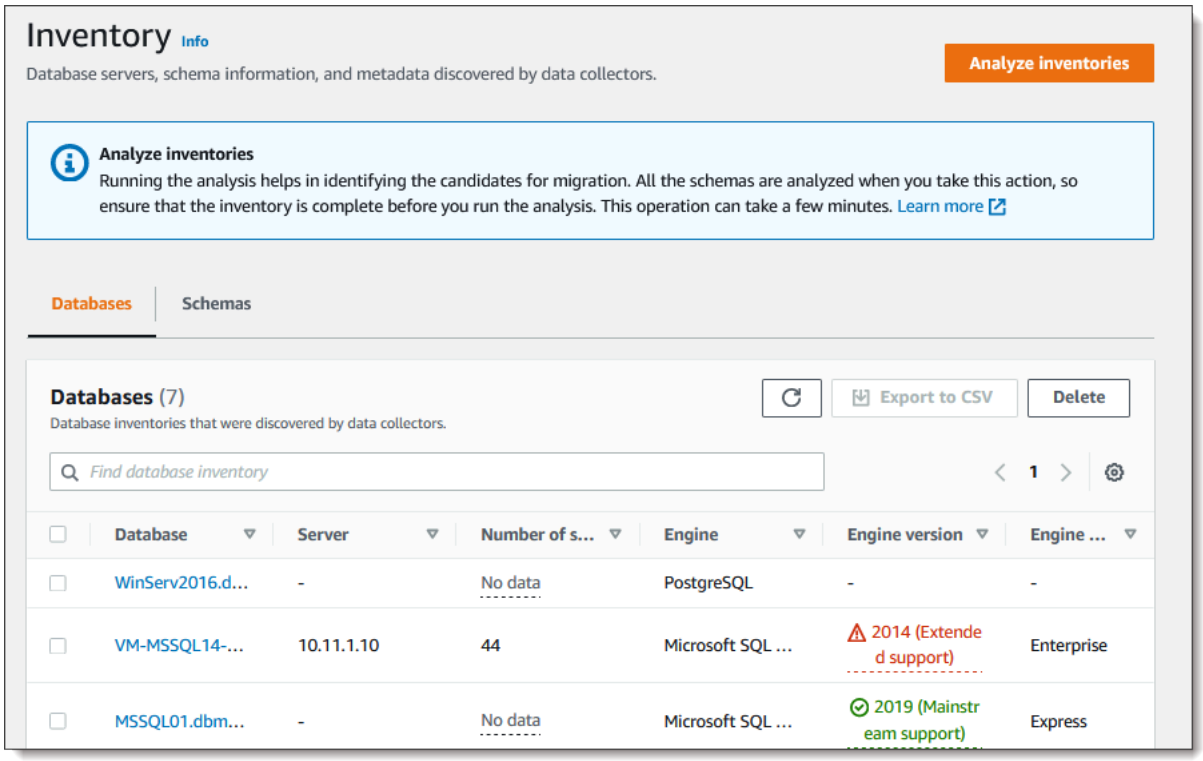
データが収集されたネットワークサーバー上のデータベースのリストを表示するには

1. コンソールで [インベントリ] を選択します。

   [Inventory] (インベントリ) ページが開きます。

2. [Databases](データベース) タブ。

   検出されたデータベースのリストが表示されます。



3. [インベントリを分析] をクリックして、類似性や複雑さなどのスキーマプロパティを指定します。このプロセスにかかる時間は分析するオブジェクトの数によって異なるとはいえ、1 時間以上かかることはありません。分析の結果は、[インベントリ] ページの [スキーマ] タブに表示されます。

DMS Fleet Advisor は、検出されたすべてのデータベースのスキーマを分析して、オブジェクトの共通部分を定義します。分析結果はパーセンテージで表されます。DMS Fleet Advisor は、共通部分が 50% 以上のスキーマを重複と見なします。元のスキーマは、重複が見つかったスキーマとして識別されます。これにより、最初に変換または移行する元のスキーマを識別できます。

インベントリ全体が一度に分析され、重複するスキーマが特定されます。

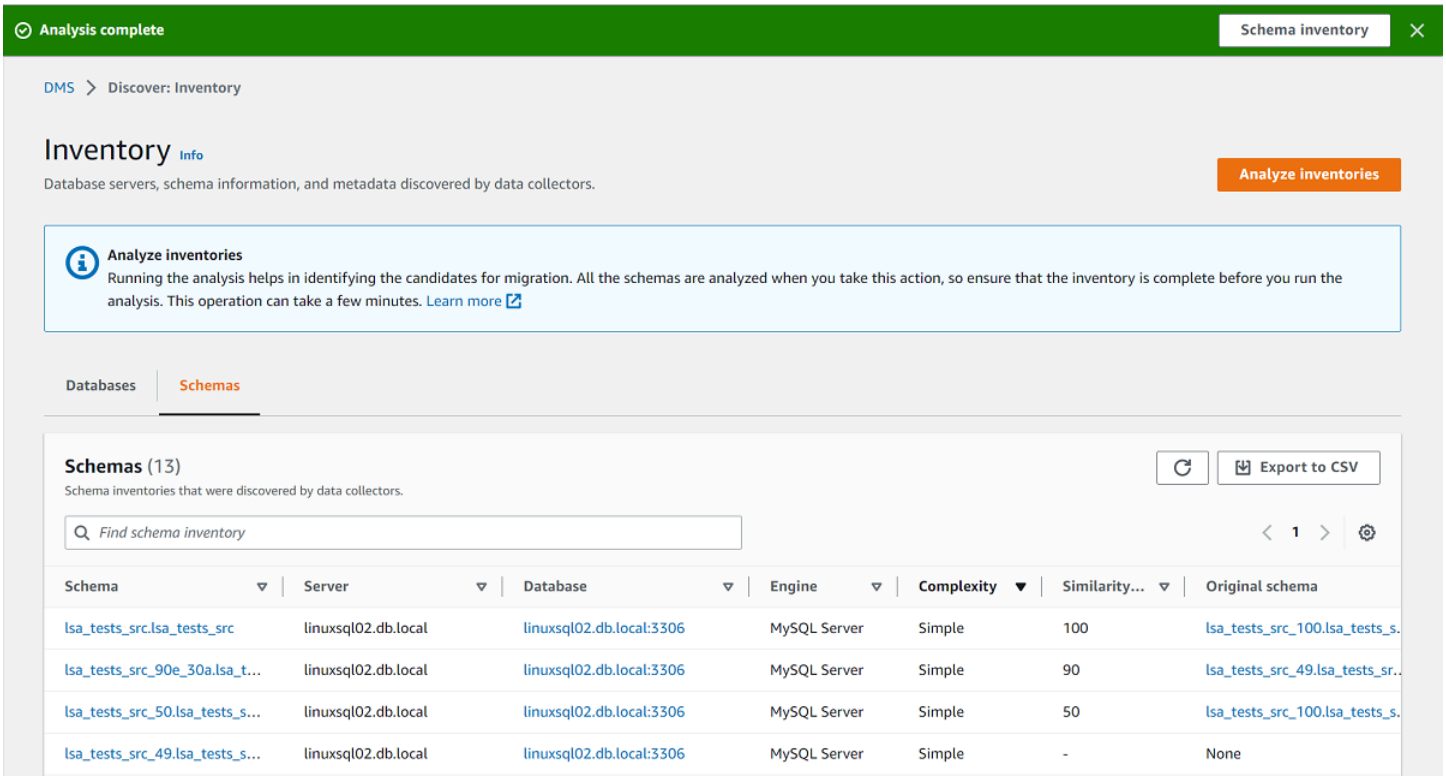
分析にスキーマインベントリを使用する

データの収集元であるネットワーク内のサーバーで検出されたデータベーススキーマのリストを確認できます。次の手順を実行します。

データが収集されたネットワークサーバー上のスキーマリストを表示するには

1. コンソールで [インベントリ] を選択します。[Inventory] (インベントリ) ページが開きます。
2. [Schemas] (スキーマ) タブを選択します。スキーマのリストが表示されます。
3. リストからスキーマを選択して、サーバー、データベース、サイズ、複雑さなどの情報を確認します。

   各スキーマについて、オブジェクトのタイプ、オブジェクトの数、オブジェクトのサイズ、コードの行数に関する情報を提供するオブジェクトの概要を表示できます。
4. (オプション) [Analyze inventories] (インベントリの分析) を選択し、重複するスキーマを識別します。DMS Fleet Advisor はデータベーススキーマを分析してオブジェクトの共通部分を定義します。
5. インベントリ情報は、後で確認するために `.csv` ファイルにエクスポートできます。



移行するスキーマを特定し、移行ターゲットを決定するには、 AWS Schema Conversion Tool (AWS SCT) または DMS Schema Conversion を使用できます。詳細については、「Using a new project wizard in AWS SCT」を参照してください。

移行するスキーマを特定した後、AWS SCT または DMS Schema Conversion を使用してスキーマを変換できます。DMS Schema Conversion の詳細については、「DMS Schema Conversion を使用したデータベーススキーマの変換」を参照してください。

AWS DMS Fleet Advisor ターゲットレコメンデーション機能の使用

最適な移行ターゲットを調べて選択するために、DMS Fleet Advisor でソースのオンプレミスのデータベースのターゲットレコメンデーションを生成できます。レコメンデーション には、オンプレミスのソースデータベースの移行に選択できる単一または複数の AWS ターゲットエンジンが含まれています。これらの可能なターゲットエンジンから、DMS Fleet Advisor は 1 つのターゲットエンジンを適切なサイズの移行先として提案し、このターゲットを DMS 推奨 として示します。DMS Fleet Advisor は、適切なサイズの移行先を決定するうえで、データコレクターが収集したインベントリのメタデータとメトリクスを使用します。

移行を開始する前にレコメンデーションを参考に、移行オプションを検出して、コストを節約し、リスクを軽減できます。レコメンデーションは、カンマ区切り値 (CSV) ファイルとしてエクスポートして、主要な関係者と共有すると。意思決定を円滑に進めることができます。推奨事項を にエクスポートAWS Pricing Calculatorして、メンテナンスコストをさらに最適化できます。詳細については、https://calculator.aws/#/ のページを参照してください。

ターゲットレコメンデーションを DMS Fleet Advisor で変更することはできません。このため、DMS Fleet Advisor を What-If 分析に使用することはできません。What-If 分析とは、ターゲットのパラメータを変更して、その変更がレコメンデーションの料金見積もりにどのような影響を及ぼすかを確認するプロセスです。レコメンデーションターゲットパラメータを AWS Pricing Calculator の出発点として使用して、AWS Pricing Calculator で What-If 分析を実行できます。詳細については、https://calculator.aws/#/ のページを参照してください。

DMS Fleet Advisor のレコメンデーションは、移行計画の出発点として検討することをお勧めします。その後、データベースワークロードのコストやパフォーマンスを最適化するために、レコメンデーションのインスタンスパラメータを変更するかを決断できます。

トピック

ターゲットインスタンスのレコメンデーション

ターゲットレコメンデーションとして、DMS Fleet Advisor は次のとおり、汎用、メモリ最適化、バーストパフォーマンスの Amazon RDS DB インスタンスを検討します。

- db.m5
- db.m6i
- db.r5
- db.r6i
- db.t3

- db.x1
- db.x1e
- db.z1d

Amazon RDS DB インスタンスクラスの詳細については、「Amazon RDS ユーザーガイド」の「DB インスタンスクラス」を参照してください。

DMS Fleet Advisor がレコメンデーションのターゲットインスタンスの仕様を決定するには

DMS Fleet Advisor は、データベースのキャパシティまたは使用率に基づいてレコメンデーションを生成できます。

- データベースのキャパシティに基づくレコメンデーションの生成を選択した場合、DMS Fleet Advisor は既存のデータベースキャパシティを最も近いインスタンスクラスの仕様にマッピングします。
- リソース使用率に基づくレコメンデーションの生成を選択した場合、DMS Fleet Advisor は CPU、メモリ、I/O スループット、IOPS などのメトリクスの 95 パーセンタイル値を決定します。95 パーセンタイルとは、収集されたデータの 95% がこの値よりも低いことを意味します。その後 DMS Fleet Advisor は、上記の値を最も近いインスタンスクラスの仕様にマッピングします。

DMS Fleet Advisor は、ターゲットデータベースのサイズを決定するうえで、ソースデータベースのサイズに関する情報を収集します。その後、DMS Fleet Advisor はターゲットストレージに同じサイズを使用するレコメンデーションを提供します。ソースデータベースストレージがオーバープロビジョンされている場合、ターゲットストレージのサイズのレコメンデーションもオーバープロビジョンされます。

AWS DMS を使用してデータを移行する場合、ターゲット DB インスタンスの IOPS プロビジョニングを増やす必要がある場合があります。ターゲットレコメンデーションを生成する際、DMS Fleet Advisor サービスはソースデータベースのメトリクスのみを考慮します。DMS Fleet Advisor は、データ移行タスクの実行に必要になる可能性のある追加の IOPS は考慮に入れません。詳細については、「移行タスクの実行が遅い」を参照してください。

IOPS コストを見積もるために、DMS Fleet Advisor はソース IOPS 使用量の one-to-one マッピングをベースラインとして使用します。DMS Fleet Advisor は、ピーク負荷時をベースライン値、つまり IOPS 料金を 100% 使用しているとみなします。

ソースデータベースが PostgreSQL や MySQL の場合、DMS Fleet Advisor は Aurora インスタンスと Amazon RDS DB インスタンスをターゲットレコメンデーションに含めることができます。Aurora の設定がソースの要件とマップできる場合、DMS Fleet Advisor はこのオプションをレコメンデーションとしてマークします。

AWS DMS Fleet Advisor でのターゲットレコメンデーションの生成

データベースと分析フリートのデータ収集とインベントリが完了したら、DMS Fleet Advisor でターゲットレコメンデーションを生成できます。生成するには、ソースデータベースを選択して、DMS Fleet Advisor ターゲットレコメンデーション機能がターゲットインスタンスのサイズを決定するために使用する設定を構成します。DMS Fleet Advisor のターゲットレコメンデーション機能では、ソースデータベースから収集したキャパシティと使用率のメトリクスが使用されます。

ターゲットレコメンデーションを生成するには

1. にサインインAWS Management Consoleし、https://console.aws.amazon.com/dms/v2/ で AWS DMSコンソールを開きます。

   DMS Fleet Advisor を使用する AWS リージョン を選択していることを確認します。
2. ナビゲーションペインで、[評価] の下の [レコメンデーション] を選択して、[レコメンデーションを生成] をクリックします。
3. [ソースデータベースを選択] パネルで、AWS クラウド に移行するデータベース名のチェックボックスをオンにします。

   [ソースデータベースを検索する] にデータベース名を入力してインベントリをフィルタリングします。

   DMS Fleet Advisor は、一度に最大 100 のデータベースについてのレコメンデーションを生成できます。
4. [アベイラビリティーおよび耐久性] では、優先するデプロイオプションを選択します。

   本番稼働用データベースのターゲットレコメンデーションを算出するには、[プロダクション (マルチ AZ)] を選択します。DMS Fleet Advisor は、ターゲットレコメンデーションに異なるアベイラビリティーゾーンに配置された 2 つの DB インスタンスを含めます。このようなマルチ AZ 配置オプションを使用すると、高可用性、データ冗長性、フェイルオーバーのサポートを提供します。

Aurora が推奨されるターゲットエンジンであり、可用性と耐久性がマルチ AZ 配置である場合、ターゲットレコメンデーションにはリーダー DB インスタンスとライター DB インスタンスが含まれます。

開発やテストに使用するデータベースのターゲットレコメンデーションを算出するには、[開発/テスト (シングル AZ)] を選択します。DMS Fleet Advisor は、ターゲットレコメンデーションに単一の DB インスタンスを含めます。このようなシングル AZ 配置オプションを使用すると、メンテナンスコストが低減します。

5. [ターゲットインスタンスのサイズ] では、DMS Fleet Advisor がターゲットレコメンデーションの算出に使用する優先オプションを選択します。

   ソースデータベースまたは OS サーバーの設定に基づいてターゲットレコメンデーションを算出するには、[キャパシティの合計] を選択します。DMS Fleet Advisor は、ソースデータベースまたは OS サーバーの合計 CPU、メモリ、ディスク容量などのメトリクスを使用してターゲットレコメンデーションを生成します。その後 DMS Fleet Advisor は、データベースのキャパシティメトリクスを最も近いインスタンスクラスの仕様にマッピングします。

   ソースデータベースまたは OS サーバーの実際の使用率に基づいてターゲットレコメンデーションを算出するには、[リソース使用率] を選択します。DMS Fleet Advisor は、ソースデータベースまたは OS サーバーの合計 CPU、メモリ、ディスク容量の使用率メトリクスを使用してターゲットレコメンデーションを生成します。DMS Fleet Advisor は、使用率メトリクスの 95 パーセンタイルを算出します。95 パーセンタイルとは、期間内のデータの 95% がこの値よりも低いことを意味します。その後 DMS Fleet Advisor は、上記の値を最も近いインスタンスクラスのインスタンスクラスにマッピングします。

   レコメンデーションの精度を向上するには、[リソース使用率] オプションを使用することをお勧めします。このオプションを使用するには、キャパシティの合計とリソース使用率のメトリクスを収集していることを確認します。

6. [Generate] (生成) を選択します。

DMS Fleet Advisor は、選択したデータベースのターゲットレコメンデーションを生成ます。レコメンデーションの生成が正常に完了すると、DMS Fleet Advisor はステータスを [Computed] に変更します。DMS Fleet Advisor は、AWS Pricing Calculator を使用してレコメンデーションのターゲット DB インスタンスの月額コストの見積りも算出します。これで、生成されたレコメンデーションを詳細に調べることができます。詳細については、「レコメンデーションの詳細」を参照してください。

データインベントリの毎月のコスト合計を見積るには、クラウドに移行する予定のデータベースのチェックボックスをオンにします。DMS Fleet Advisor は、合計月額料金の概算と、AWS クラウド内のターゲットデータベースの概要を表示します。DMS Fleet Advisor は、AWS の料金表Query APIを使用して、情報提供のみを目的として料金の詳細を提供します。実際の料金は、AWS のサービスサービスの実際の使用状況などのさまざまな要因により異なります。AWS のサービス サービス料金の詳細については、「クラウドサービスの料金」を参照してください。

AWS DMS Fleet Advisor でのターゲットレコメンデーションの詳細の調査

DMS Fleet Advisor がターゲットレコメンデーションを生成した後、[レコメンデーション] テーブルで推奨される移行ターゲットの主要なパラメータを確認できます。このような主要パラメータには、ターゲットエンジン、インスタンスクラス、仮想 CPU の数、メモリ、ストレージ、ストレージタイプなどがあります。このようなパラメータ以外にも、DMS Fleet Advisor はこの推奨される移行ターゲットの月額コストの見積りも表示します。

各レコメンデーションには、AWS ターゲットエンジン候補が 1 つまたは複数提示されている場合があります。レコメンデーションに複数のターゲットエンジンがある場合、AWS DMS はそのうちのいずれかをお勧めとしてマークします。AWS DMS は、この推奨されるオプションのパラメータと月額コストの見積りも [レコメンデーション] テーブルで提供します。

ターゲットレコメンデーションをソースデータベースの使用率やキャパシティと比較するには、レコメンデーションを詳しく調べます。また、選択したレコメンデーションの移行に関する制限を確認することもできます。このような制限には、サポートされていないデータベース機能、アクション項目、その他の移行に関する考慮事項が含まれます。

レコメンデーションを詳しく調べるには

1. DMS Fleet Advisor を使用してターゲットレコメンデーションを生成します。詳細については、「ターゲットレコメンデーションの生成」を参照してください。
2. [レコメンデーション] テーブルからレコメンデーション名を選択します。レコメンデーションページが開きます。
3. レコメンデーションに複数のターゲットオプションが含まれている場合は、[ターゲットレコメンデーション] でターゲットのオプションを選択します。
4. [ソースの使用率とキャパシティ] セクションを展開します。DMS Fleet Advisor は、次のメトリクスのリソース使用率グラフを表示します。

   - CPU の数
   - 「メモリ」

- I/O スループット
- 1 秒あたりの入出力オペレーション (IOPS)
- [Storage (ストレージ)]
- アクティブなデータベースサーバー接続の数

このようなグラフを利用して、DMS データコレクターからのソースデータベースメトリクスを、選択したターゲットエンジンのメトリクスと比較します。

ソースの使用率と容量セクションを展開してもグラフが表示されない場合は、Amazon CloudWatch ダッシュボードを表示するアクセス許可を IAM ユーザーに付与していることを確認してください。詳細については、「Amazon ユーザーガイド」の「Amazon CloudWatch ダッシュボードの使用」を参照してください。 CloudWatch

5. 選択したターゲットエンジン名のリンクをクリックします。[ターゲットの詳細] ページが開きます。
6. ターゲットレコメンデーションを CSV にエクスポートするには、「アクション」ドロップダウンから「CSV にエクスポート」オプションを選択します。
7. ターゲットレコメンデーションを にエクスポートするにはAWS Pricing Calculator、アクションドロップダウンから オプションでコストを最適化 AWS Pricing Calculator を選択します。
8. [設定] セクションで、ソースデータベースのパラメータの値とターゲットエンジンのパラメータを比較します。ターゲットエンジンについては、DMS Fleet Advisor はクラウドリソースの月額コスト見積りを提示します。DMS Fleet Advisor は、AWS の料金表Query API を使用して、情報提供のみを目的として料金の詳細を提供します。実際の料金は、AWS のサービス サービスの実際の使用状況などのさまざまな要因により異なります。AWS のサービス サービス料金の詳細については、「https://aws.amazon.com/pricing/クラウドサービスの料金」を参照してください。
9. [移行制限] セクションで、移行の制限を確認します。ソースデータベースを AWS クラウド に移行する際には、このような制限を考慮に入れることをお勧めします。

AWS DMS Fleet Advisor でのターゲットレコメンデーションのエクスポート

ターゲットレコメンデーションを生成した後、レコメンデーションのリストのコピーをカンマ区切り値 (CSV) ファイルとして保存できます。

ターゲットレコメンデーションを生成するには

1. にサインインAWS Management Consoleし、https://console.aws.amazon.com/dms/v2/ で AWS DMSコンソールを開きます。

   DMS Fleet Advisor を使用する AWS リージョン を選択していることを確認します。
2. ナビゲーションペインで、[評価] の下の [レコメンデーション] を選択して、CSV ファイルに含めるレコメンデーションを選択します。
3. [CSV にエクスポート] をクリックしてファイル名を入力し、PC のファイル保存先フォルダを選択します。
4. CSV ファイルを開きます。

レコメンデーションを記載した CSV ファイルには、次の情報が含まれています。

- CreatedDate – DMS Fleet Advisor がターゲットエンジンのレコメンデーションを作成した日付。
- DatabaseId – DMS Fleet Advisor がこのレコメンデーションを作成したソースデータベースの識別子。
- DeploymentOption – 推奨される Amazon RDS DB インスタンスのデプロイオプション。
- EngineEdition – 推奨されるターゲット Amazon RDS エンジンエディション。
- EngineName – ターゲットエンジンの名前。
- InstanceMemory – 推奨される Amazon RDS DB インスタンスのメモリ量。
- InstanceSizingType – ターゲットインスタンスのサイズ。
- InstanceType – 推奨されるターゲット Amazon RDS インスタンスタイプ。
- InstanceVcpu – 推奨される Amazon RDS DB インスタンスCPUs の数。
- Preferred – このターゲットオプションが推奨されていることを示すブール型フラグ
- Status – ターゲットエンジンのレコメンデーションのステータス
- StorageIops – 推奨 Amazon RDS DB インスタンスで 1 秒ごとに完了した I/O オペレーションの数 (IOPS)。
- StorageSize – 推奨される Amazon RDS DB インスタンスのストレージサイズ。
- StorageType – 推奨される Amazon RDS DB インスタンスのストレージタイプ。
- WorkloadType — マルチ AZ 配置やシングル AZ 配置など、ターゲットエンジンのデプロイオプション。

Fleet Advisor AWS DMS による移行制限の検出と分析

DMS データコレクターを使用して、ターゲットエンジンがサポートしていないデータベース機能を検出できます。適切な移行ターゲットを選択するには、次の制限を考慮する必要があります。

DMS データコレクターは、特定のソースデータベース機能を検出します。次に、DMS Fleet Advisor は、移行の観点から指定されたターゲットへのソース機能を分析し、制限に関する追加情報を提供し、この制限に対処または回避するための推奨アクションを含めます。また、DMS Fleet Advisor はこのような制限の影響を算出します。

制限のリストは、ターゲットエンジンの詳細ページで確認できます。左側のナビゲーションメニューの「レコメンデーション」ページからこのページに移動します。ターゲットのリストから、確認するターゲットエンジンを選択します。制限のリストはページの下部にあります。

次の表は、Amazon RDS for MySQL ではサポートされていない MySQL データベース機能を説明しています。

制限	説明	Impact
認証プラグイン	Amazon RDS は、MySQL 認証プラグインをサポートしていません。	低
システムログへのエラーログ記録	Amazon RDS は、エラーログのシステムログへの書き込みをサポートしていません。	低
グローバルトランザクション ID (GTID)	グローバルトランザクション ID は、RDS for MySQL 5.7 バージョン、RDS for MySQL 8.0.26 以降の MySQL 8.0 バージョンで使用できます。	低
グループレプリケーション	Amazon RDS は、MySQL グループレプリケーションプラグインをサポートしていません。	低

制限	説明	Impact
InnoDB テーブルスペースの暗号化	Amazon RDS は、InnoDB テーブルスペースの暗号化をサポートしていません。	低
InnoDB 予約語	InnoDB は、Amazon RDS for MySQL の予約語です。My SQL データベースにこの名前を使用することはできません。	低
キーリングプラグイン	Amazon RDS は、MySQL キーリングプラグインをサポートしていません。	低
パスワード検証プラグイン	Amazon RDS は、MySQL `validate_password` プラグインをサポートしていません。	低
永続的システム可変	Amazon RDS は、MySQL の永続システム変数をサポートしていません。	低
制限付きアクセス	Amazon RDS は、高度なアクセス許可を必要とする特定のシステムプロシージャやテーブルへのアクセスを制限しています。Amazon RDS は、Telnet、Secure Shell (SSH)、または Windows のリモートデスクトップ接続を使用した DB インスタンスへの直接ホストアクセスを許可していません。	低

制限	説明	Impact
Rewriter クエリ書き換えプラグイン	Amazon RDS は、MySQL Rewriter クエリ書き換えプラグインをサポートしていません。	低
準同期レプリケーション	Amazon RDS は、MySQL の準同期レプリケーションをサポートしていません。	低
トランスポータブルテーブルスペース	Amazon RDS は、MySQL のトランスポータブルテーブルスペースをサポートしていません。	低
X プラグイン	Amazon RDS は、MySQL X プラグインをサポートしていません。	低

次の表は、Amazon RDS for Oracle ではサポートされていない Oracle データベース機能を説明しています。

制限	説明	Impact
Active Data Guard	Active Data Guard は Oracle マルチテナントコンテナデータベース (CDB) では使用できません。	中程度
Automatic Storage Management (ASM)	Amazon RDS は、Oracle Automatic Storage Management (Oracle ASM) をサポートしていません。	中程度
データベースアクティビティストリーミング	Amazon RDS は、シングルテナントアーキテクチャ向け	高い

制限	説明	Impact
	の Oracle Database Activity Streams をサポートしていません。	
ファイルサイズの制限	RDS for Oracle DB インスタンス上の単一のファイルの最大サイズは 16 TiB (テビバイト) です。	中程度
FTP および SFTP	Amazon RDS は、FTP と SFTP をサポートしていません。	中程度
パーティション分割されたハイブリッドテーブル	Amazon RDS は、Oracle ハイブリッドパーティションテーブルをサポートしていません。	高い
Oracle Data Guard	Amazon RDS は、シングルテナントアーキテクチャ向けの Oracle Data Guard をサポートしていません。	高い
Oracle Database Vault	Amazon RDS は、Oracle Database Vault をサポートしていません。	高い
Oracle DBA 権限 Vault	Amazon RDS には Oracle DBA 権限に制限があります。詳細については、「Oracle DBA 権限の制限事項」を参照してください。	高い
Oracle Enterprise Manager	Amazon RDS は、シングルテナントアーキテクチャ向けの Oracle Enterprise Manager をサポートしていません。	高い

制限	説明	Impact
Oracle Enterprise Manager Agent	Amazon RDS は、シングルテナントアーキテクチャ向けの Oracle Enterprise Manager Agent をサポートしていません。	中程度
Oracle Enterprise Manager Cloud Control Management Repository	Amazon RDS for Oracle DB インスタンスを Oracle Enterprise Manager Cloud Control 管理リポジトリに使用することはできません。	高い
Oracle Flashback Database	Amazon RDS は、Oracle Flashback Database 機能をサポートしていません。	高い
Oracle Label Security	Amazon RDS は、シングルテナントアーキテクチャ向けの Oracle Label Security をサポートしていません。Oracle Label Security はマルチテナントコンテナデータベース (Oracle CDB) でのみ使用できます。	高い
Oracle Messaging Gateway	Amazon RDS は、Oracle Messaging Gateway をサポートしていません。	高い
Oracle Real Application Clusters	Amazon RDS は、Oracle Real Application Clusters (Oracle RAC) をサポートしていません。	高い

制限	説明	Impact
Oracle Real Application Testing	Amazon RDS は、Oracle Real Application Testing をサポートしていません。	高い
Oracle スナップショットスタンバイデータベース	Amazon RDS は、Oracle スナップショットスタンバイデータベースをサポートしていません。	高い
パブリックシノニム	Amazon RDS は、Oracle が提供するスキーマのパブリックシノニムをサポートしていません。	中程度
サポートされていない機能のスキーマ	Amazon RDS は、システム権限を必要とする Oracle の機能やコンポーネントのスキーマをサポートしていません。	高い
純粋な統合監査	Amazon RDS は、純粋な統合監査をサポートしていません。統合監査は混合モードで使用できます。	中程度
Workspace Manager	Amazon RDS は、Oracle Database Workspace Manager の `WMSYS` スキーマをサポートしていません。	高い

次の表は、Amazon RDS for PostgreSQL ではサポートされていない PostgreSQL データベース機能を説明しています。

制限	説明	Impact
同時接続	RDS for PostgreSQL インスタンスへの同時接続数の上限は、`max_connections` パラメータにより制限されます。	低
最新バージョン	Amazon RDS は、メジャーバージョンのアップグレードを自動的に適用しません。メジャーバージョンのアップグレードを実行するには、DB インスタンスを手動で変更します。詳細については、「PostgreSQL のメジャーバージョンアップグレードの選択」を参照してください。	低
予約済み接続	Amazon RDS では、システムメンテナンス用に最大 3 接続が予約されます。ユーザー接続パラメータの値を指定する場合は、使用予定の接続数に 3 を加えます。	低
サポートされる拡張機能	RDS for PostgreSQL でサポートされる PostgreSQL データベースエンジンの拡張機能には制限があります。サポートされている拡張機能のリストは、PostgreSQL バージョンのデフォルトの DB パラメータグループで確認できます。`psql` を使用して `rds.extensions` パラメータを表示すると、現在の拡張	低

制限	説明	Impact
	機能のリストを確認することもできます。	
テーブルスペースの分割または分離	I/O の分割や分離にテーブルスペースを使用することはできません。RDS for PostgreSQL では、すべてのストレージが単一の論理ボリューム上に配置されます。	低

次の表は、Amazon RDS for SQL Server ではサポートされていない SQL Server データベース機能を説明しています。

制限	説明	Impact
Microsoft Azure Blob ストレージへのバックアップ	RDS for SQL Server は、Microsoft Azure Blob ストレージへのバックアップをサポートしていません。	中程度
バッファプールの拡張	RDS for SQL Server は、バッファープール拡張機能をサポートしていません。	高い
カスタムパスワードポリシー	RDS for SQL Server は、カスタムパスワードポリシーをサポートしていません。	中程度
データクオリティ・サービス	RDS for SQL Server は、SQL Server の Data Quality Services (DQS) をサポートしていません。	高い

制限	説明	Impact
データベースのログ配布	RDS for SQL Server は、データベースのログ配布をサポートしていません。	高い
データベース名	データベース名には次の制限があります。先頭に rdsadmin は使用できない、先頭と末尾は空白文字やタブは使用できない、改行文字は使用できない、一重引用符 (') は使用できない。	中程度
データベーススナップショット	RDS for SQL Server は、データベースのスナップショットをサポートしていません。Amazon RDS では DB インスタンスのスナップショットのみを使用できます。	中程度
拡張ストアドプロシージャ	RDS for SQL Server は、`xp_cmdshell` などの拡張ストアドプロシージャをサポートしていません。	高い
ファイルテーブル	RDS for SQL Server は、ファイルテーブルをサポートしていません。	中程度
FILESTREAM のサポート	RDS for SQL Server は FILESTREAM サポートを提供しません。	中程度
リンクサーバー	RDS for SQL Server のリンクサーバーのサポートは限定的です。	高い

制限	説明	Impact
Machine Learning と R Services	Machine Learning と R Services のインストールには OS へのアクセスが必要であるため、RDS for SQL Server は、Machine Learning と R Services をサポートしていません。	高い
メンテナンスプラン	RDS for SQL Server は、メンテナンスプランをサポートしていません。	高い
パフォーマンスデータコレクター	RDS for SQL Server は、パフォーマンスデータコレクターをサポートしていません。	高い
ポリシーベースの管理	RDS for SQL Server は、ポリシーベースの管理をサポートしていません。	中程度
PolyBase	RDS for SQL Server は をサポートしていません PolyBase。	高い
レプリケーション	RDS for SQL Server は、レプリケーションをサポートしていません。	中程度
リソースガバナー	RDS for SQL Server は、リソースガバナーをサポートしていません。	高い
サーバーレベルのトリガー	RDS for SQL Server は、サーバースコープのトリガーをサポートしていません。	中程度

制限	説明	Impact
サービスブローカーエンドポイント	RDS for SQL Server は、Service Broker エンドポイントをサポートしていません。	高い
SSAS	RDS for SQL Server での SQL Server Analysis Services (SSAS) の実行に適用される制限を考慮に入れる必要があります。詳細については、「制限」を参照してください。	低
SSIS	RDS for SQL Server での SQL Server Integration Services (SSIS) の実行に適用される制限を考慮に入れる必要があります。詳細については、「制限」を参照してください。	低
SSRS	RDS for SQL Server での SQL Server Reporting Services (SSRS) の実行に適用される制限を考慮に入れる必要があります。詳細については、「制限」を参照してください。	低
SQL Server DB インスタンスのストレージ サイズ	SQL Server の汎用 (SSD) ストレージとプロビジョンド IOPS ストレージインスタンスの最大ストレージサイズは 16 TiB (テビバイト) です。 SQL Server の磁気ストレージインスタンスの最大ストレージサイズは 1 TiB (テビバイト) です。	高い

制限	説明	Impact
Stretch Database	RDS for SQL Server は、SQL Server の Stretch Database 機能をサポートしていません。	中程度
T-SQL エンドポイント	RDS for SQL Server は、`CREATE ENDPOINT` を使用するすべての操作をサポートしているわけではありません。	高い
TRUSTWORTHY データベースプロパティ	`sysadmin` ロールが必要となるため、RDS for SQL Server は、`TRUSTWORTHY` データベースプロパティをサポートしていません。	中程度

次の表に、レコメンデーションの問題のリストを示します。DMS Fleet Advisor は、ソースデータベースとターゲットデータベースの機能を分析し、これらの移行制限を提供します。ブロッカーの影響に関する制限は、DMS Fleet Advisor がソースデータベースのターゲットレコメンデーションを生成できないことを意味します。

制限	説明	Impact
適切なインスタンスが見つからない	AWS DMS は、ソースデータベースメトリクスの組み合わせに適したサイズの移行先として動作するターゲットインスタンスを見つけることができません。	ブロッカー
適切なインスタンスが IOPS で見つからない	ソースデータベースは、可能なターゲット DB インスタンスの最大 IOPS 数を超える IOPS を多数使用します。	ブロッカー

制限	説明	Impact
適切なインスタンスが RAM で見つからない	ソースデータベースは、可能なターゲット DB インスタンスの RAM の最大サイズを超える GB の RAM を使用します。	ブロッカー
適切なインスタンスがストレージサイズで見つからない	ソースデータベースは、可能なターゲット DB インスタンスの最大ストレージサイズを超える多数の TB のストレージを使用します。	ブロッカー
適切なインスタンスがエディションによって見つからない	ソースデータベースにはエディションがあり、Amazon RDS ではサポートされていません。	ブロッカー
適切なインスタンスが CPU コアによって見つからない	ソースデータベースには多数の CPU コアがあり、可能なターゲット DB インスタンスの最大 CPU コア数を超えています。	ブロッカー
適切なインスタンスがバージョンによって見つからない	ソースデータベースには バージョンがあり、 AWS DMSでは認識されません。	ブロッカー

制限	説明	Impact
CPU パラメータが未定義	DMS データコレクターは、ソースデータベースが使用する CPU に関する情報を収集しませんでした。データコレクターでデータ転送に必要なメトリクスと設定済みの認証情報を収集したことを確認します。データ転送のための認証情報の設定 を参照してください。	ブロッカー
メモリパラメータが未定義	DMS データコレクターは、ソースデータベースが使用するメモリに関する情報を収集しませんでした。データコレクターでデータ転送に必要なメトリクスと設定済みの認証情報を収集したことを確認します。データ転送のための認証情報の設定 を参照してください。	ブロッカー
ストレージサイズパラメータが未定義	DMS データコレクターは、ソースデータベースが使用するストレージサイズに関する情報を収集しませんでした。データコレクターでデータ転送に必要なメトリクスと設定済みの認証情報を収集したことを確認します。データ転送のための認証情報の設定 を参照してください。	ブロッカー

制限	説明	Impact
ストレージ IOPS パラメータが未定義	DMS データコレクターは、ソースデータベースが使用するストレージ IOPS メトリクスを収集しませんでした。データコレクターでデータ転送に必要なメトリクスと設定済みの認証情報を収集したことを確認します。	ブロッカー
十分なデータがない	DMS データコレクターは、ターゲットレコメンデーションを生成するのに十分なデータを収集しませんでした。データコレクターでデータを転送するための認証情報が設定されていることを確認します。データ転送のための認証情報の設定 を参照してください。	ブロッカー
データベースエディションが未定義	DMS データコレクターはソースデータベースのエディションに関する情報を収集しませんでした。データコレクターでデータ転送に必要なメトリクスと設定済みの認証情報を収集したことを確認します。データ転送のための認証情報の設定 を参照してください。	ブロッカー
未知のエラー	DMS Fleet Advisor は、ソースデータベースのターゲットレコメンデーションを生成できません。	ブロッカー

制限	説明	Impact
データベースバージョンが未定義	DMS Fleet Advisor は、ソースデータベースのバージョンに関する情報を収集しませんでした。DMS Fleet Advisor では、ソースデータベースに最新のデータベースバージョンを使用することをお勧めします。このレコメンデーションを選択した場合は、データベースのバージョンをアップグレードする必要があります。ソースデータベースに対して生成されたターゲットレコメンデーションを確認し、これらのレコメンデーションが要件を満たしていることを確認します。	高い
RDS 設定のデータベース接続数を増やす	ソースデータベースには一定数の接続が必要です。デフォルトでは、Amazon RDS データベースインスタンスで使用できる接続の数は異なります。RDS データベースインスタンスを作成するときに、このデフォルト値を必ず変更してください。そのためには、`max_connections` パラメータグループのパラメータ値を更新します。	中程度

制限	説明	Impact
ターゲットエディションには互換性があります	ソースデータベースのターゲットレコメンデーションでは、異なるデータベースエディションを使用します。ソースデータベースエディションは、推奨されるターゲットエディションと同じ機能をサポートしています。ただし、この新しいデータベースエディションを選択すると、費用が増加する可能性があります。	中程度
ストレージスループットパラメータが未定義	DMS データコレクターは、ソースデータベースが使用するストレージスループットメトリクスを収集しませんでした。ソースデータベースに対して生成されたターゲットレコメンデーションを確認し、これらのレコメンデーションが要件を満たしていることを確認します。	中程度
データベース接続番号パラメータが未定義	DMS データコレクターは、ソースデータベースが使用する接続数に関する情報を収集しませんでした。ソースデータベースに対して生成されたターゲットレコメンデーションを確認し、これらのレコメンデーションが要件を満たしていることを確認します。または、クォータの引き上げをリクエストします。	中程度

制限	説明	Impact
データベースダウングレードバージョン	ソースデータベースは、Amazon RDS データベースよりも高いバージョンで実行されます。データベースのバージョンをダウングレードするには、下位バージョンに実装されていない機能を使用しないようにしてください。または、Amazon EC2 を移行ターゲットとして使用します。	中程度
ターゲットエディションが異なる	ソースデータベースのターゲットレコメンデーションでは、異なるデータベースエディションを使用します。ソースデータベースの工ディションは、推奨されるターゲット工ディションと互換性があります。ただし、推奨されるターゲットデータベースエディションは、ソースデータベース工ディションの一部の機能をサポートしていません。この新しいデータベースエディションを選択すると、費用が増加する可能性があります。	中程度

制限	説明	Impact
サポートされていないバージョンからのアップグレード	ソースデータベースがサポート終了段階に達しました。最新の DB エンジンバージョンをターゲットとして使用するには、移行前にデータベースをアップグレードします。または、Amazon EC2 を移行ターゲットとして使用します。 データベースエンジンに応じて、詳細について次のいずれかのリンクを使用します。 MySQL のアップグレード SQL Server のアップグレード OracleDB のアップグレード PostgreSQL のアップグレード	中程度

ターゲットレコメンデーションのトラブルシューティング

次のリストは、DMS Fleet Advisor のターゲットレコメンデーション機能で問題が発生した場合に実行できるアクションを提供しています。

トピック

- ターゲットレコメンデーションの料金見積りが表示されない
- リソース使用率グラフが表示されない
- メトリクス収集ステータスが表示されない

ターゲットレコメンデーションの料金見積りが表示されない

ステータスが [成功] であるレコメンデーションの [月額料金の概算] に [データがありません] と表示されている場合、AWS の料金表 Service API にアクセスする権限を IAM ユーザーに付与しているかを確認します。これを実行するには、「IAM リソースを作成する」に記載されるとおり `pricing:GetProducts` アクセス許可を含むポリシーを作成して、IAM ユーザーに付与する必要があります。

DMS Fleet Advisor は、ステータスが [失敗] のレコメンデーションの月額コスト見積りは算出しません。

リソース使用率グラフが表示されない

ソースの使用率と容量セクションを展開した後にメトリクスのロードに失敗したというメッセージが表示された場合は、Amazon CloudWatch ダッシュボードを表示するアクセス許可を IAM ユーザーに付与していることを確認してください。これを実行するには、「IAM リソースを作成する」に記載されるとおり、必要なポリシーを IAM ユーザーに付与する必要があります。

また
は、`cloudwatch:GetDashboard`、`cloudwatch:ListDashboards`、`cloudwatch:PutDashboard`、`c`
のアクセス許可を持つカスタムポリシーを作成することもできます。詳細については、「Amazon
ユーザーガイド」の「Amazon CloudWatch ダッシュボードの使用」を参照してください。
CloudWatch

メトリクス収集ステータスが表示されない

[レコメンデーションを生成] をクリックした際、[メトリクスコレクション] に [利用可能なデータがありません] と表示された場合は、データが収集されたかを確認します。詳細については、「AWS DMS Fleet Advisor のデータ収集」を参照してください。

データを収集した後にこの問題が発生した場合は、Amazon へのアクセス`cloudwatch:Get*`許可を IAM ユーザーに付与していることを確認してください CloudWatch。DMS Fleet Advisor は、サービスにリンクされたロールを使用して、収集したデータベースパフォーマンスメトリクスをユーザー CloudWatch に代わって に公開します。DMS Fleet Advisor で使用するサービスにリンクされたロールを必ず作成してください。詳細については、「IAM リソースを作成する」を参照してください。

DMS Fleet Advisor の制限事項

DMS Fleet Advisor を使用する場合の制限は次のとおりです。

- DMS Fleet Advisor は one-to-one レコメンデーションを生成します。DMS Fleet Advisor は、ソースデータベースごとに単一のターゲットエンジンを決定します。DMS Fleet Advisor はマルチテナントサーバーを取扱いません。単一のターゲット DB インスタンスで複数のデータベースを実行するためのレコメンデーションは提供しません。
- DMS Fleet Advisor は、利用可能なデータベースバージョンアップグレードに関するレコメンデーションを提供しません。
- DMS Fleet Advisor は、一度に最大 100 のデータベースに関するレコメンデーションを生成します。
- Windows アプリケーションである DMS データコレクターをインストールする場合は、.NET Framework 4.8 および PowerShell 6.0 以降もインストールしてください。ハードウェア要件については、「データコレクターのインストール」を参照してください。
- DMS データコレクターには、ドメインサーバーで LDAP プロトコルを使用してリクエストを実行するアクセス許可が必要です。
- DMS データコレクターには Linux で実行される sudo SSH スクリプトが必要です。
- DMS データコレクターには、リモート PowerShell、Windows Management Instrumentation (WMI)、WMI クエリ言語 (WQL)、および Windows のレジストリスクリプトを実行するアクセス許可が必要です。
- MySQL と PostgreSQL については、DMS Fleet Advisor はデータベースからパフォーマンスメトリクスを収集できません。DMS Fleet Advisor は、その代わりに OS サーバーのメトリクスを収集します。そのため、Amazon RDS と Aurora で実行される MySQL データベースと PostgreSQL データベースの使用率メトリクスに基づくレコメンデーションは生成できません。

DMS Schema Conversion を使用したデータベーススキーマの変換

AWS Database Migration Service（ AWS DMS) の DMS Schema Conversion を使用すると、さまざまなタイプのデータベース間のデータベース移行をより予測しやすくなります。DMS Schema Conversion を使用して、ソースデータプロバイダーの移行の複雑さの評価、データベーススキーマとコードオブジェクトの変換ができます。その後、変換したコードをターゲットデータベースに適用できます。

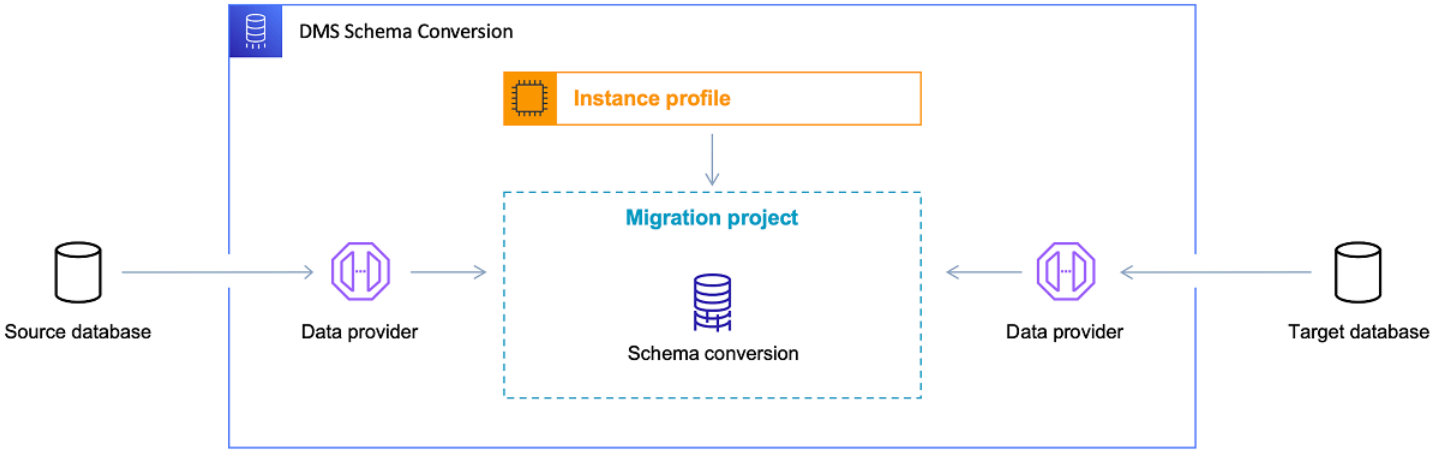
DMS Schema Conversion は、ソースデータベーススキーマとほとんどのデータベースコードオブジェクトをターゲットデータベースと互換性のある形式に自動的に変換します。この変換は、テーブル、ビュー、ストアドプロシージャ、関数、データ型、シノニムなどを対象としています。DMS Schema Conversion が自動的に変換できないオブジェクトには、判別しやすいマークが付けられます。移行を完了するには、このようなオブジェクトは手動で変換します。

DMS Schema Conversion は概括すると、インスタンスプロファイル、データプロバイダー、移行プロジェクトの 3 つのコンポーネントで動作します。[インスタンスプロファイル] では、ネットワークとセキュリティの設定を指定します。[データプロバイダー] には、データベース接続の認証情報を保存します。移行プロジェクトには、データプロバイダー、インスタンスプロファイル、移行ルールが含まれています。 AWS DMS は、データプロバイダーとインスタンスプロファイルを使用して、データベーススキーマとコードオブジェクトを変換するプロセスを設計します。

サポート対象のソースデータベースの一覧については、「DMS Schema Conversion のソース」を参照してください。

サポート対象のターゲットデータベースの一覧については、「DMS Schema Conversion のターゲット」を参照してください。

次の図は、DMS Schema Conversion のプロセスを説明しています。



DMS Schema Conversion の使用方法の理解を深めるには、次のトピックを利用できます。

トピック

- サポートされる AWS リージョン
- スキーマ変換の機能
- スキーマ変換の制限
- DMS Schema Conversion の開始方法
- DMS Schema Conversion のためのネットワークのセットアップ
- DMS Schema Conversion でのソースデータプロバイダーの作成
- DMS Schema Conversion でのターゲットデータプロバイダーの作成
- DMS スキーマ変換での移行プロジェクトの管理
- DMS Schema Conversion でのデータベース移行評価レポートの作成
- DMS Schema Conversion の使用
- DMS Schema Conversion での拡張パックの使用

サポートされる AWS リージョン

DMS Schema Conversion 移行プロジェクトは、次の で作成できます AWS リージョン。これ以外の リージョンでは、 AWS Schema Conversion Tool を利用できます。の詳細については AWS SCT、AWS 「 Schema Conversion Tool ユーザーガイド」を参照してください。

リージョン名	リージョン
米国東部 (バージニア北部)	us-east-1
米国東部 (オハイオ)	us-east-2
米国西部 (オレゴン)	us-west-2
アジアパシフィック (東京)	ap-northeast-1
アジアパシフィック (シンガポール)	ap-southeast-1
アジアパシフィック (シドニー)	ap-southeast-2
欧州 (フランクフルト)	eu-central-1
欧州 (ストックホルム)	eu-north-1
欧州 (アイルランド)	eu-west-1

スキーマ変換の機能

DMS Schema Conversion は次の機能を提供します。

- DMS Schema Conversion は、データベース移行プロジェクトに必要な AWS クラウド リソースを自動的に管理します。これらのリソースには、インスタンスプロファイル、データプロバイダー、シー AWS Secrets Manager クレットが含まれます。また、 AWS Identity and Access Management (IAM) ロール、Amazon S3 バケット、移行プロジェクトも含まれます。
- DMS Schema Conversion を使用すると、ソースデータベースへの接続、メタデータの読み取り、データベース移行評価レポートの作成ができます。このレポートは、Amazon S3 バケットに保存できます。このようなレポートを使用すると、スキーマ変換タスクの概要、DMS Schema Conversion がターゲットデータベースに自動的に変換できない項目の詳細を取得できます。データベース移行評価レポートは、DMS Schema Conversion が移行プロジェクトをどの程度自動化できるかを評価するのに役立ちます。また、移行評価レポートは、変換を完了するのに必要な手動の

作業量の見積りにも役立ちます。詳細については、「DMS Schema Conversion でのデータベース移行評価レポートの作成」を参照してください。

- ソースデータプロバイダーとターゲットデータプロバイダーに接続すると、DMS Schema Conversion は既存のソースデータベーススキーマをターゲットデータベースエンジンに変換できます。ソースデータベースから任意のスキーマ項目を選択して変換できます。DMS Schema Conversion でデータベースコードを変換した後、ソースコードと変換したコードを確認できます。変換済みの SQL コードは、Amazon S3 バケットに保存することもできます。
- ソースデータベーススキーマを変換する前に、変換ルールを設定できます。変換ルールを使用すると、列のデータ型の変更、オブジェクトのスキーマの移動、オブジェクト名の変更ができます。変換ルールは、データベース、スキーマ、テーブル、列に適用できます。詳細については、「 変換ルールの設定」を参照してください。
- コンバージョン設定を変更して、変換済みのコードのパフォーマンスを向上させることができます。コンバージョン設定は各変換ペアに固有であり、コードで使用するソースデータベースの機能によって異なります。詳細については、「スキーマ変換設定の指定」を参照してください。
- 場合によって、DMS Schema Conversion ではソースデータベースの機能を同等の Amazon RDS 機能に変換できないことがあります。このような場合、DMS Schema Conversion はターゲットデータベースに拡張パックを作成して、変換されなかった機能をエミュレートします。詳細については、「拡張パックの使用」を参照してください。
- 変換済みのコードと拡張パックスキーマをターゲットデータベースに適用できます。詳細については、「変換したコードの適用」を参照してください。
- DMS Schema Conversion は、最新 AWS SCT リリースのすべての機能をサポートしています。詳細については、AWS 「SCT の最新リリースノート」を参照してください。
- DMS がターゲットデータベースに移行する前に、変換された SQL コードを編集できます。詳細については、「変換された SQL コードの編集と保存 」を参照してください。

スキーマ変換の制限

DMS Schema Conversion は () のウェブバージョンです AWS Schema Conversion Tool AWS SCT。DMS Schema Conversion は、 AWS SCT デスクトップアプリケーションと比較すると、サポート対象のデータベースプラットフォームが少なく、提供される機能も限られています。データウェアハウスのスキーマ、ビッグデータフレームワーク、アプリケーション SQL コード、ETL プロセスを変換するには、 AWS SCT を使用してください。の詳細については AWS SCT、AWS 「 Schema Conversion Tool ユーザーガイド」を参照してください。

DMS Schema Conversion を使用してデータベーススキーマを変換する場合、次の制限が適用されます。

- 移行プロジェクトの保存やオフラインモードでの使用はできません。
- DMS Schema Conversion の移行プロジェクトでソースの SQL コードを編集することはできません。ソースデータベースの SQL コードを編集するには、通常の SQL エディタを使用します。更新したコードを移行プロジェクトに追加するには、[データベースから更新] をクリックします。
- DMS Schema Conversion の移行ルールは、列の照合順序の変更をサポートしていません。移行ルールを使用してオブジェクトを新しいスキーマに移動することもできません。
- ソースデータベースとターゲットデータベースのツリーをフィルタリングして、フィルター句を満たすデータベースオブジェクトのみを表示することはできません。
- DMS Schema Conversion 拡張パックには、変換されたコードの E メール送信、ジョブスケジューリング、その他の機能をエミュレートする AWS Lambda 機能は含まれていません。
- DMS Schema Conversion は、カスタマー AWS リソースへのアクセスにカスタマーマネージド KMS キーを使用しません。例えば、DMS Schema Conversion は、カスタマーマネージド KMS キーを使用した Amazon S3 のお客様のデータへのアクセスをサポートしていません。

DMS Schema Conversion の開始方法

DMS Schema Conversion の使用を開始するには、次のチュートリアルを使用します。このチュートリアルでは、DMS Schema Conversion の設定方法、移行プロジェクトの作成方法、データプロバイダーへの接続方法について説明しています。その後、移行の複雑さを評価して、ソースデータベースをターゲットデータベースと互換性のある形式に変換する方法を説明しています。さらに、変換したコードをターゲットデータベースに適用する方法も説明しています。

次のチュートリアルでは、前提となるタスクを説明し、Amazon RDS for SQL Server データベースから Amazon RDS for MySQL への変換のデモを提供しています。サポート対象のソースデータプロバイダーとターゲットデータプロバイダーは任意のものを使用できます。詳細については、「DMS Schema Conversion のソースデータプロバイダー」を参照してください。

DMS スキーマ変換の詳細については、Oracle から PostgreSQL への移行と SQL Server から MySQL step-by-step への移行に関する移行チュートリアルをご覧ください。

この動画では、DMS Schema Conversion ユーザーインターフェイスと、このサービスのコアコンポーネントを概説しています。

トピック

DMS Schema Conversion を使用するための前提条件

DMS Schema Conversion をセットアップするには、次のタスクを実行します。その後、インスタンスプロファイルを設定し、データプロバイダーを追加して、移行プロジェクトを作成できます。

トピック

Amazon VPC を基盤に VPC を作成する

このステップでは、に仮想プライベートクラウド (VPC) を作成します。 AWS アカウントこの VPC は Amazon Virtual Private Cloud (Amazon VPC) サービスをベースとしており、 AWS お客様のリソースを含んでいます。

DMS Schema Conversion のための VPC を作成するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/vpc/ にある Amazon VPC コンソールを開きます。
2. [Create VPC (VPC の作成)] を選択します。
3. [VPC の作成] ページで、次の設定を入力します。

    - 作成するリソース – VPC など

- 名前タグの自動生成 – [自動生成] を選択して、グローバルに一意の名前を入力する。たとえば、**sc-vpc** と入力します。
- [IPv4 CIDR block] (IPv4 CIDR ブロック) – **10.0.1.0/24**
- NAT ゲートウェイ – In 1 AZ
- [VPC endpoints] (VPC エンドポイント) – なし

4. 残りの設定はそのままにして、[VPC を作成] をクリックします。
5. [サブネット] を選択して、パブリックサブネット ID とプライベートサブネット ID をメモしておきます。

   Amazon RDS データベースに接続するには、パブリックサブネットを備えたサブネットグループを作成します。

   オンプレミスのデータベースに接続するには、プライベートサブネットを備えたサブネットグループを作成します。詳細については、「ステップ 1: インスタンスプロファイルを作成する」を参照してください。
6. [NAT ゲートウェイ] を選択します。[NAT ゲートウェイ] を選択して、[Elastic IP アドレス] をメモしておきます。

   この NAT ゲートウェイのパブリック IP AWS DMS アドレスからソースのオンプレミスデータベースにアクセスできるよう、ネットワークを設定します。詳細については、「VPC へのインターネット接続の使用」を参照してください。

Amazon RDS でインスタンスプロファイルとターゲットデータベースを作成する場合は、この VPC を使用します。

Amazon S3 バケットを作成する

移行プロジェクトからの情報を保存するには、Amazon S3 バケットを作成します。DMS Schema Conversion は、この Amazon S3 バケットを使用して、評価レポート、変換済みの SQL コード、データベーススキーマオブジェクトに関する情報などのアイテムを保存します。

DMS Schema Conversion のための Amazon S3 バケットを作成するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/s3/ にある 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. [バケットの作成] ページで、S3 バケットのグローバルに一意な名前を選択します。たとえば、**sc-s3-bucket** と入力します。
4. [AWS リージョン] では、現在作業中のリージョンを作成します。
5. [バケットのバージョニング] では、[有効にする] をオンにします。
6. 残りの設定はそのままにして、[バケットを作成] をクリックします。

データベース認証情報を以下に保存します。 AWS Secrets Manager

AWS Secrets Managerソースデータベースとターゲットデータベースの認証情報をに保存します。これらのシークレットは必ずに複製してください。 AWS リージョン DMS Schema Conversion は、移行プロジェクトのデータベースへの接続に、上記のシークレットを使用します。

データベースの認証情報をに保存するには AWS Secrets Manager

1. AWS Management Console にサインインし、https://console.aws.amazon.com/secretsmanager/ AWS Secrets Manager のコンソールを開きます。
2. [新しいシークレットを保存] を選択します。
3. [シークレットのタイプを選択] ページが開きます。シークレットタイプで、保存するデータベース認証情報のタイプを選択します。

   - Amazon RDS データベースの認証情報 – Amazon RDS データベースの認証情報を保存するには、このオプションを選択する。[認証情報] には、作業中のデータベースの認証情報を入力する。[Database] (データベース) で、データベースを選択します。
   - その他のデータベースの認証情報 – Oracle または SQL Server のソースデータベースの認証情報を保存するには、このオプションを選択する。[認証情報] には、作業中のデータベースの認証情報を入力する。
   - その他のシークレットのタイプ – データベースに接続するためのユーザー名とパスワードのみを保存するには、このオプションを選択する。[行の追加] をクリックして、キーバリューのペアを 2 つ追加する。キー名には必ず **username** と **password** を使用する。これらのキーに関連する値には、データベースの認証情報を入力する。
4. [暗号化キー] では、Secrets Manager AWS KMS がシークレット値を暗号化するために使用するキーを選択します。[次へ] をクリックします。
5. [シークレットを設定] ページで、判別しやすい [シークレット名] を入力します。たとえば、**sc-source-secret** ないし **sc-target-secret** を入力します。

6. [シークレットをレプリケート] をクリックして、[AWS リージョン] では作業中のリージョンを選択します。[次へ] をクリックします。
7. [ローテーションを設定する] ページで、[次へ] をクリックします。
8. [Review] (レビュー) ページで、シークレットの詳細を確認し、[Store] (保存) を選択します。

ソースデータベースとターゲットデータベースの認証情報を保存するには、この手順を繰り返します。

IAM ロールを作成する

移行プロジェクトで使用する AWS Identity and Access Management (IAM) ロールを作成します。DMS Schema Conversion は、このような IAM ロールを使用して、Amazon S3 バケットと AWS Secrets Managerに保存されている データベース認証情報にアクセスします。

Amazon S3 バケットへのアクセスを提供する IAM ロールを作成するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/iam/ にある IAM コンソールを開きます。
2. ナビゲーションペインで Roles (ロール) を選択します。
3. [ロールの作成] を選択します。
4. [信頼されたエンティティを選択] ページで、[AWS サービス] を選択します。DMS を選択します。
5. [次へ] をクリックします。[許可を追加] ページが開きます。
6. [Filter policies] (ポリシーのフィルタリング) を使用する場合、**S3** と入力します。AmazonS3FullAccess を選択してください。
7. [次へ] をクリックします。[名前、確認、および作成] ページが開きます。
8. [ロール名] には判別しやすい名前を入力します。たとえば、**sc-s3-role** と入力します。[ロールを作成] を選択します。
9. [ロール] ページの [ロール名] には、**sc-s3-role** と入力します。[sc-s3-role] を選択します。
10. [sc-s3-role] ページの [信頼関係] タブをクリックします。[Edit trust policy] (信頼ポリシーを編集) を選択します。
11. [信頼ポリシーを編集] ページで、信頼できるエンティティとして `schema-conversion.dms.amazonaws.com` サービスプリンシパルを使用するようにロールの信頼関係を編集します。
12. [信頼ポリシーの更新] を選択します。

アクセスを提供する IAM ロールを作成するには AWS Secrets Manager

1. AWS Management Console にサインインし、https://console.aws.amazon.com/iam/ にある IAM コンソールを開きます。
2. ナビゲーションペインで Roles (ロール) を選択します。
3. [ロールの作成] を選択します。
4. [信頼されたエンティティを選択] ページで、[AWS サービス] を選択します。DMS を選択します。
5. [次へ] をクリックします。[許可を追加] ページが開きます。
6. [Filter policies] (ポリシーのフィルタリング) を使用する場合、**Secret** と入力します。を選択しますSecretsManagerReadWrite。
7. [次へ] をクリックします。[名前、確認、および作成] ページが開きます。
8. [ロール名] には判別しやすい名前を入力します。たとえば、**sc-secrets-manager-role** と入力します。[ロールを作成] を選択します。
9. [ロール] ページの [ロール名] には、**sc-secrets-manager-role** と入力します。選択sc-secrets-manager-role。
10. sc-secrets-manager-roleページで [信頼関係] タブを選択します。[Edit trust policy] (信頼ポリシーを編集) を選択します。
11. 「信頼ポリシーの編集」ページで、schema-conversion.dms.amazonaws.com使用するロールの信頼関係と、 AWS DMS 信頼されたエンティティとしての地域のサービスプリンシパルを編集します。 AWS DMS このリージョナルサービスプリンシパルの形式は次のとおりです。

    ```
    dms.region-name.amazonaws.com
    ```

    [######] は、us-east-1 などの作業中のリージョン名に置き換えます。

    次のコード例は、us-east-1 リージョンのプリンシパルを示しています。

    ```
    dms.us-east-1.amazonaws.com
    ```

    次のコード例は、 AWS DMS スキーマ変換にアクセスするための信頼ポリシーを示しています。

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "dms.us-east-1.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "schema-conversion.dms.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

12. [信頼ポリシーの更新] を選択します。

ステップ 1: インスタンスプロファイルを作成する

インスタンスプロファイルを作成する前に、インスタンスプロファイルのサブネットグループを設定します。 AWS DMS 移行プロジェクト用のサブネットグループの作成について詳しくは、を参照してくださいサブネットグループの作成。

次の手順の説明のとおり、インスタンスプロファイルを作成します。このインスタンスプロファイルでは、DMS Schema Conversion プロジェクトのネットワークとセキュリティの設定を指定します。

インスタンスプロファイルを作成するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/dms/v2/ AWS DMS のコンソールを開きます。
2. ナビゲーションペインで、[インスタンスプロファイル] を選択して、[インスタンスプロファイルの作成] をクリックします。
3. [名前] には、インスタンスプロファイルの一意の名前を入力します。たとえば、**sc-instance** と入力します。
4. [ネットワークタイプ] では、[IPv4] を選択して、IPv4 アドレス指定のみをサポートするインスタンスプロファイルを作成します。IPv4 と IPv6 のアドレス指定をサポートするインスタンスプロファイルを作成するには、[デュアルスタックモード] を選択します。

5. [仮想プライベートクラウド (VPC)] では、前提条件のステップで作成した VPC を選択します。
6. [サブネットグループ] では、インスタンスプロファイルのサブネットグループを選択します。Amazon RDS データベースに接続するには、パブリックサブネットを備えたサブネットグループを使用します。オンプレミスのデータベースに接続するには、プライベートサブネットを備えたサブネットグループを使用します。
7. [インスタンスプロファイルの作成] をクリックします。

移行プロジェクトを作成するには、このインスタンスプロファイルを使用します。

ステップ 2: データプロバイダーを設定する

次に、ソースデータベースとターゲットデータベースを記述するデータプロバイダーを作成します。データプロバイダーごとに、データストアのタイプと場所の情報を指定します。データベースの認証情報は、データプロバイダーには保存しません。

オンプレミスのソースデータベースのデータプロバイダーを作成するには

1. にサインインし AWS Management Console、 AWS DMS コンソールを開きます。
2. ナビゲーションペインで、[データプロバイダー] を選択して、[データプロバイダーの作成] をクリックします。
3. [名前] には、ソースデータプロバイダーの一意の名前を入力します。たとえば、**`sc-source`** と入力します。
4. [エンジンタイプ] では、データプロバイダーのデータベースのエンジンタイプを選択します。
5. ソースデータベースには接続情報を指定します。接続パラメータはソースデータベースエンジンによって異なります。詳細については、「データプロバイダーの作成」を参照してください。
6. [Secure Socket Layer (SSL) モード] では、SSL の強制タイプを選択します。
7. [データプロバイダーの作成] をクリックします。

ターゲットの Amazon RDS データベースのデータプロバイダーを作成するには

1. AWS Management Console にログインし、 AWS DMS コンソールを開きます。
2. ナビゲーションペインで、[データプロバイダー] を選択して、[データプロバイダーの作成] をクリックします。
3. [設定] で、[RDS データベースインスタンス] を選択します。

4. [RDS からのデータベース] では、[参照] をクリックして、データベースを選択します。エンジンタイプ、サーバー名、ポートに関する情報は、DMS Schema Conversion が自動的に取得します。
5. [名前] には、ターゲットデータプロバイダーの一意の名前を入力します。たとえば、**sc-target** と入力します。
6. [Database name (データベース名)] には、データベースの名前を入力します。
7. [Secure Socket Layer (SSL) モード] では、SSL の強制タイプを選択します。
8. [データプロバイダーの作成] をクリックします。

ステップ 3: 移行プロジェクトを作成する

これで、移行プロジェクトを作成できるようになりました。移行プロジェクトでは、ソースデータプロバイダーとターゲットデータプロバイダー、インスタンスプロファイルを指定します。

移行プロジェクトを作成するには

1. [移行プロジェクト] を選択して、[移行プロジェクトの作成] をクリックします。
2. [名前] には、移行プロジェクトの一意の名前を入力します。たとえば、**sc-project** と入力します。
3. [インスタンスプロファイル] では、[**sc-instance**] を選択します。
4. [ソース] では、[参照] をクリックして、[**sc-source**] を選択します。
5. [シークレット ID] では、[**sc-source-secret**] を選択します。
6. [IAM ロール] に **sc-secrets-manager-role** を選択します。
7. [ターゲット] では、[参照] をクリックして、[**sc-target**] を選択します。
8. [シークレット ID] では、[**sc-target-secret**] を選択します。
9. [IAM ロール] に **schema-conversion-role** を選択します。
10. [移行プロジェクトの作成] をクリックします。

ステップ 4: 評価レポートを作成する

移行の複雑さを評価するには、データベース移行評価レポートを作成します。このレポートには、DMS Schema Conversion が自動的に変換できないすべてのデータベースオブジェクトのリストが記載されています。

評価レポートを作成するには

1. [移行プロジェクト] を選択して、[**sc-project**] を選択します。
2. [スキーマ変換] を選択して、[スキーマ変換を起動] をクリックします。
3. ソースデータベースペインで、評価するデータベーススキーマを選択します。また、このスキーマ名のチェックボックスをオンにします。
4. ソースデータベースペインの [アクション] メニューで [評価] を選択します。[評価] ダイアログボックスが開きます。
5. ダイアログボックスの [評価] をクリックして、この選択を確定します。

   [概要] タブには、DMS Schema Conversion がDMS Schema Conversion がデータベーストレージオブジェクトとデータベースコードオブジェクトについて自動的に変換できる項目の数が表示されています。
6. [アクション項目] をクリックすると、DMS Schema Conversion が自動的に変換できないすべてのデータベースオブジェクトの一覧が表示されます。各項目の推奨アクションを確認します。
7. 評価レポートのコピーを保存するには、[結果をエクスポート] をクリックします。次に、[CSV] または [PDF] のいずれかの形式を選択します。[エクスポート] ダイアログボックスが開きます。
8. [エクスポート] をクリックして、選択を確定します。
9. [S3 バケット] をクリックします。Amazon S3 コンソールが開きます。
10. [ダウンロード] をクリックして、評価レポートを保存します。

ステップ 5: ソースコードを変換する

次の手順を使用して、ソースデータベーススキーマを変換できます。その後、変換したコードを SQL スクリプトとしてテキストファイルに保存できます。

データベーススキーマを変換するには

1. ソースデータベースペインで、変換するデータベーススキーマを選択します。また、このスキーマ名のチェックボックスをオンにします。
2. ソースデータベースペインの [アクション] メニューで [変換] を選択します。[変換] ダイアログボックスが開きます。
3. ダイアログボックスの [変換] をクリックして、この選択を確定します。
4. ソースデータベースペインでデータベースオブジェクトを選択します。DMS Schema Conversion には、このオブジェクトのソースコードと変換したコードが表示されます。Edit

SQL 機能を使用して、データベースオブジェクトに変換された SQL コードを編集できます。詳細については、「変換された SQL コードの編集と保存」を参照してください。

5. ターゲットデータベースペインで、変換したデータベーススキーマを選択します。また、このスキーマ名のチェックボックスをオンにします。
6. [アクション] では、[SQL として保存] を選択します。[保存] ダイアログボックスが開きます。
7. [SQL として保存] をクリックして、選択を確定します。
8. [S3 バケット] をクリックします。Amazon S3 コンソールが開きます。
9. [ダウンロード] をクリックして、SQL スクリプトを保存します。

ステップ 6: 変換したコードを適用する

DMS Schema Conversion では、変換済みのコードはターゲットデータベースにすぐには適用されません。ターゲットデータベースを更新するには、前のステップで作成した SQL スクリプトを使用できます。または、次の手順を使用して DMS Schema Conversion から変換したコードを適用します。

変換したコードを適用するには

1. ターゲットデータベースペインで、変換したデータベーススキーマを選択します。また、このスキーマ名のチェックボックスをオンにします。
2. [アクション] では、[変更を適用] を選択します。[変更の適用] ダイアログボックスが開きます。
3. [適用] をクリックして、選択を確定します。

ステップ 7: クリーンアップとトラブルシューティング

Amazon CloudWatch を使用して DMS スキーマ変換ログを確認または共有できます。

DMS Schema Conversion ログを確認するには

1. AWS Management Console にサインインし、https://console.aws.amazon.com/cloudwatch/ CloudWatch にあるコンソールを開きます。
2. [Logs] (ログ)、[Log groups] (ロググループ) を選択します。

   DMS Schema Conversion のロググループ名は `dms-tasks-sct` で始まります。ロググループは [作成時刻] 順に並べ替えて DMS Schema Conversion ロググループを検索できます。

また、ロググループ名には、移行プロジェクトの Amazon リソースネーム (ARN) が含まれています。プロジェクトの ARN は、DMS Schema Conversion の [移行プロジェクト] ページで確認できます。[ARN] は、[詳細設定] で必ず選択します。

3. ロググループ名を選択してから、ログストリーム名を選択します。
4. [アクション] では、[結果をエクスポート] を選択して、DMS Schema Conversion ログを保存します。

DMS Schema Conversion でのスキーマ変換が完了した後、リソースをクリーンアップします。

DMS Schema Conversion のリソースをクリーンアップするには

1. AWS Management Console にサインインし、 AWS DMS コンソールを開きます。
2. ナビゲーションペインで [移行プロジェクト] を選択します。
   a. [**sc-project**] を選択します。
   b. [スキーマ変換] を選択して、[スキーマ変換を閉じる] をクリックします。
   c. [削除] をクリックして、選択を確定します。
3. ナビゲーションペインで [インスタンスプロファイル] を選択します。
   a. [**sc-instance**] を選択します。
   b. [削除] をクリックして、選択を確定します。
4. ナビゲーションペインで [データプロバイダー] を選択します。
   a. [**sc-source**] と [**sc-target**] を選択します。
   b. [削除] をクリックして、選択を確定します。

また、Amazon S3 バケット、内のデータベースシークレット、IAM ロール、仮想プライベートクラウド (VPC) など AWS Secrets Manager、 AWS 作成した他のリソースも必ずクリーンアップしてください。

DMS Schema Conversion のためのネットワークのセットアップ

DMS Schema Conversion は、Amazon VPC サービスを基盤に仮想プライベートクラウド (VPC) 内にスキーマ変換インスタンスを作成します。使用する VPC は、インスタンスプロファイルを作成す

る際に指定します。アカウントと AWS リージョン リージョンのデフォルトの VPC を使用することも、新しい VPC を作成することもできます。

DMS Schema Conversion のソースデータベースとターゲットデータベースとの連携のセットアップには、さまざまなネットワーク構成を使用できます。このような設定は、ソースデータプロバイダーの場所とネットワーク設定により異なります。次のトピックでは、一般的なネットワーク設定について説明します。

トピック

ソースデータプロバイダーとターゲットデータプロバイダーでの単一の VPC の使用

DMS Schema Conversion の最もシンプルなネットワーク設定は、単一の VPC の設定です。この場合、ソースデータプロバイダー、インスタンスプロファイル、ターゲットデータプロバイダーはすべて同じ VPC に配置されます。この設定を使用して、Amazon EC2 インスタンスでソースデータベースを変換できます。

この設定を使用するには、インスタンスプロファイルで使用される VPC セキュリティグループがデータプロバイダーにアクセスできることを確認する必要があります。例えば、VPC クラスレスドメイン間ルーティング (CIDR) 範囲または Elastic IP アドレスのいずれかをネットワークアドレス変換 (NAT) ゲートウェイに許可できます。

ソースデータプロバイダーとターゲットデータプロバイダーに複数の VPC を使用する

ソースデータプロバイダーとターゲットデータプロバイダーが別々の VPC にある場合は、いずれかの VPC にインスタンスプロファイルを作成できます。その後、VPC ピアリング接続を使用して 2 つの VPC をリンクできます。この設定を使用して、Amazon EC2 インスタンスでソースデータベースを変換できます。

VPC ピアリング接続は、2 つの VPC 間のネットワーキング接続で、同じネットワーク内にあるかのように各 VPC のプライベート IP アドレスを使用してルーティングを有効にできます。VPC ピアリング接続は、アカウント内の複数 VPC 間、別の AWS アカウントの VPC との間、別の AWS リージョン の VPC との間で作成できます。VPC ピアリング接続の詳細については、「Amazon VPC ユーザーガイド」の「VPC ピアリング接続」を参照してください。

VPC ピアリング接続を実装するには、「Amazon VPC ユーザーガイド」の「VPC ピア接続を操作する」を参照してください。一方の VPC のルートテーブルに、もう一方の VPC の CIDR ブロックが含まれていることを確認します。例えば、A という VPC が送信先 10.0.0.0/16 を使用しており、B という VPC が送信先 172.31.0.0 を使用しているとします。この場合、VPC A のルートテーブルには 172.31.0.0、VPC B のルートテーブルには 10.0.0.0/16 が含まれている必要があります。詳細については、「Amazon VPC ピアリング接続ガイド」の「VPC ピアリング接続のルートテーブルを更新する」を参照してください。

AWS Direct Connect または VPN を使用した VPC へのネットワーク設定

リモートネットワークは、AWS Direct Connect やソフトウェアまたはハードウェア VPN 接続など、いくつかのオプションを使用して VPC に接続できます。このようなオプションを使用して、内部ネットワークを AWS クラウド に拡張することで、既存のオンサイトサービスを統合できます。モニタリング、認証、セキュリティ、データ、その他のシステムなどのオンサイトサービスを統合する場合があります。このようなタイプのネットワーク拡張を使用すると、オンサイトのサービスを VPC などの AWS がホストするリソースにシームレスに接続できます。この設定を使用して、オンプレミスのソースデータベースを変換できます。

この設定では VPC セキュリティグループに、VPC の CIDR 範囲または特定の IP アドレスを送信先とするトラフィックをホストに送信するルーティングルールが必要です。このホストは、VPC からのトラフィックをオンプレミスの VPN にブリッジできる必要があります。この場合、NAT ホストには独自のセキュリティグループ設定が必要です。このような設定の場合、VPC の CIDR 範囲またはセキュリティグループから NAT インスタンスへのトラフィックを許可する必要があります。詳細については、「AWS Site-to-Site VPN ユーザーガイド」の「Site-to-Site VPN 接続を作成する」を参照してください。

VPC へのインターネット接続の使用

AWS リソースへの接続に VPN または AWS Direct Connect を使用しない場合は、インターネット接続を使用できます。この設定の場合、インターネットゲートウェイを備えた VPC 内のプライベートサブネットが必要です。ゲートウェイにはターゲットデータプロバイダーとインスタンスプロファイルが必要です。この設定を使用して、オンプレミスのソースデータベースを変換できます。

VPC にインターネットゲートウェイを追加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のアタッチ」を参照してください。

VPC ルートテーブルには、VPC が送信先ではないトラフィックをデフォルトでインターネットゲートウェイに送信するルーティングルールが含まれている必要があります。この設定の場合、データプロバイダーへの接続が NATゲートウェイのパブリック IP アドレスからのように見えます。詳細については、「Amazon VPC ユーザーガイド」の「VPC ルートテーブル」を参照してください。

インターネットゲートウェイを備えていない環境の使用

インターネットゲートウェイを使用せずにスキーマ変換のための環境を作成するには、次の手順を実行します。

1. 次の変更点を使用して、「開始 チュートリアル」のステップ 1～3 を実行します。
   - パブリックサブネットの代わりにプライベートサブネットを選択する。
   - インスタンスの作成中、[パブリック IP を割り当てる] で [いいえ] を選択する。
2. Amazon VPC コンソールを開きます。
3. [エンドポイント] を選択して、[エンドポイントを作成] をクリックします。
4. [エンドポイントの作成] ページで、次のとおり設定します。
   - [サービスカテゴリ] では、[AWS サービス] を選択する。
   - [サービス] リストで、[com.amazonaws.***{region}***.secretsmanager] を選択する。
   - [VPC] セクションで、作成した VPC を選択する。
   - この VPC のサブネットを選択する。
   - この VPC のセキュリティグループを選択する。
   - [ポリシー] では、[フルアクセス] が選択されたままにする。
5. 「開始チュートリアル」の残りのステップを実行します。

DMS Schema Conversion でのソースデータプロバイダーの作成

DMS Schema Conversion の移行プロジェクトでは、Microsoft SQL Server、Oracle、または PostgreSQL データベースをソースデータプロバイダーとして使用できます。ソースデータプロバイダーとして、オンプレミスまたは Amazon EC2 インスタンスで実行されているセルフマネージド型エンジンを使用できます。

ソースデータプロバイダーと DMS Schema Conversion 間の連携を許可するようにネットワークを設定します。詳細については、「DMS Schema Conversion のためのネットワークのセットアップ」を参照してください。

トピック

- DMS Schema Conversion でのソースとしての Microsoft SQL Server データベースの使用
- DMS Schema Conversion でのソースとしての Oracle データベースの使用
- DMS スキーマ変換のソースとしての Oracle Data Warehouse データベースの使用
- DMS Schema Conversion でのソースとしての PostgreSQL データベースの使用
- DMS Schema Conversion でのソースとしての MySQL データベースの使用

DMS Schema Conversion でのソースとしての Microsoft SQL Server データベースの使用

DMS Schema Conversion では、SQL Server データベースを移行のソースとして使用できます。

SQL Server のデータベースコードオブジェクトを DMS Schema Conversion を使用して次のターゲットに変換できます。

- Aurora MySQL
- Aurora PostgreSQL
- RDS for MySQL
- RDS for PostgreSQL

サポートされている SQL Server データベースのバージョンの詳細については、「DMS Schema Conversion のソースデータプロバイダー」を参照してください。

ソース SQL Server データベースでの DMS Schema Conversion の使用の詳細については、「SQL Server から MySQL への移行 step-by-step チュートリアル」を参照してください。

Microsoft SQL Server をソースとする場合の権限

Microsoft SQL Server をソースとして使用する場合に必要となる権限は、次のリストのとおりです。

- VIEW DEFINITION
- VIEW DATABASE STATE

VIEW DEFINITION を使用すると、パブリックアクセス許可を持つユーザーはオブジェクト定義を表示できます。DMS Schema Conversion は、VIEW DATABASE STATE 権限を使用して、SQL Server Enterprise Edition の機能を確認します。

スキーマを変換する、各データベースの付与を繰り返します。

さらに、master データベースに次の権限を付与します。

- VIEW SERVER STATE
- VIEW ANY DEFINITION

DMS Schema Conversion は、VIEW SERVER STATE 権限を使用してサーバーの設定と構成を収集します。データプロバイダーを表示する VIEW ANY DEFINITION 権限は必ず付与します。

Microsoft Analysis Services に関する情報を読み取るには、master データベースで次のコマンドを実行します。

```
EXEC master..sp_addsrvrolemember @loginame = N'<user_name>', @rolename = N'sysadmin'
```

上記の例の *<user_name>* プレースホルダーを、前に必要な権限を付与したユーザー名に置き換えます。

SQL Server エージェントに関する情報を読み取るには、SQLAgentUser ロールにユーザーを追加します。msdb データベースで次のコマンドを実行します。

```
EXEC sp_addrolemember <SQLAgentRole>, <user_name>;
```

上記の例の *<SQLAgentRole>* プレースホルダーを、SQL Server エージェントのロール名に置き換えます。次に、*<user_name>* プレースホルダーを、前に必要な権限を付与したユーザー名に置き換えます。詳細については、「Amazon RDS ユーザーガイド」の「SQL AgentUser ロールへのユーザーの追加」を参照してください。

ログ配布を検出するには、SELECT on dbo.log_shipping_primary_databases データベースに対する msdb 権限を付与します。

データ定義言語 (DDL) レプリケーションの通知方法を使用するには、ソースデータベースに対する RECEIVE ON *<schema_name>*.*<queue_name>* 権限を付与します。この例では、*<schema_name>* プレースホルダーをデータベースのスキーマ名に置き換えます。次に、*<queue_name>* プレースホルダーをキューテーブル名に置き換えます。

DMS Schema Conversion でのソースとしての Oracle データベースの使用

DMS Schema Conversion では、Oracle データベースを移行のソースとして使用できます。

Oracle データベースに接続するには、Oracle System ID (SID) を使用します。Oracle SID を見つけるには、Oracle データベースに対して以下のクエリを発行します。

```
SELECT sys_context('userenv','instance_name') AS SID FROM dual;
```

Oracle Database のデータベースコードオブジェクトを DMS Schema Conversion を使用して次のターゲットに変換できます。

- Aurora MySQL
- Aurora PostgreSQL
- RDS for MySQL
- RDS for PostgreSQL

サポートされている Oracle データベースのバージョンの詳細については、「DMS Schema Conversion のソースデータプロバイダー」を参照してください。

ソース Oracle データベースで DMS Schema Conversion を使用する方法の詳細については、「Oracle から PostgreSQL への移行 step-by-step チュートリアル」を参照してください。

ソースとしての Oracle の権限

Oracle をソースとして使用するには、次の権限が必要です。

- CONNECT
- SELECT_CATALOG_ROLE
- SELECT ANY DICTIONARY
- SELECT ON SYS.ARGUMENT$

DMS スキーマ変換のソースとしての Oracle Data Warehouse データベースの使用

DMS スキーマ変換の移行ソースとして Oracle Data Warehouse データベースを使用して、データベースコードオブジェクトとアプリケーションコードを Amazon Redshift に変換できます。

サポートされる Oracle データベースのバージョンについては、「DMS Schema Conversion のソースデータプロバイダー」を参照してください。ソース Oracle データベースで DMS Schema Conversion を使用する方法の詳細については、「Oracle から PostgreSQL への移行 step-by-step チュートリアル」を参照してください。

Oracle Data Warehouse データウェアハウスをソースとして使用する場合の権限

Oracle Data Warehouse をソースとして使用するには、次の権限が必要です。

- CONNECT
- SELECT_CATALOG_ROLE
- SELECT ANY DICTIONARY

Oracle データウェアハウスから Amazon Redshift への変換設定

DMS スキーマ変換を編集する方法については、「移行プロジェクトのスキーマ変換設定の指定」を参照してください。

Oracle Data Warehouse から Amazon Redshift への変換設定には、以下の設定が含まれます。

- 選択した重大度以上のアクション項目に関するコメントを変換後のコードに追加する: この設定では、変換後のコード内のアクション項目に関するコメントの数を制限します。DMS は、選択した重大度以上のアクション項目に関するコメントを変換後のコードに追加します。

  たとえば、変換したコード内のコメントの数を最小限に抑えるには、[エラーのみ] を選択します。変換したコードのすべてのアクション項目にコメントを含めるには、[すべてのメッセージ] を選択します。
- ターゲット Amazon Redshift クラスターの最大テーブル数: この設定では、DMS がターゲット Amazon Redshift クラスターに適用できるテーブルの最大数を設定します。Amazon Redshift には、クラスターノードタイプの使用を制限するクォータがあります。この設定では、以下の値がサポートされます。
  - 自動: DMS はノードタイプに応じてターゲット Amazon Redshift クラスターに適用するテーブルの数を決定します。
  - 値を設定する: テーブル数を手動で設定します。

  DMS は、テーブルの数が Amazon Redshift クラスターが保存できる数よりも多い場合でも、すべてのソーステーブルを変換します。DMS は変換後のコードをプロジェクトに保存し、ターゲットデータベースには適用しません。変換後のコードを適用したときに Amazon Redshift クラスター

のテーブルクォータに達すると、DMS は警告メッセージを表示します。また、DMS は、テーブル数が制限に達するまで、ターゲット Amazon Redshift クラスターにテーブルを適用します。

Amazon Redshift のテーブルクォータについては、「Amazon Redshift のクォータと制限」を参照してください。

- UNION ALL ビューを使用する: この設定では、DMS が 1 つのソーステーブルに対して作成できるターゲットテーブルの最大数を設定できます。

  Amazon Redshift は、テーブルのパーティションをサポートしていません。テーブルのパーティショニングをエミュレートしてクエリの実行を迅速化するために、DMS はソーステーブルの各パーティションを Amazon Redshift の個別のテーブルに移行できます。次に、DMS は、作成したすべてのターゲットテーブルのデータを含むビューを作成します。

  DMS は、ソーステーブルのパーティションの数を自動的に決定します。ソーステーブルパーティショニングのタイプによっては、この数は Amazon Redshift クラスターに適用できるテーブルのクォータを超える場合があります。このクォータに達しないようにするには、DMS が 1 つのソーステーブルのパーティションに対して作成できるターゲットテーブルの最大数を入力します。デフォルトのオプションは 368 個のテーブルです。これは 1 年 366 日のパーティションと、NO RANGE パーティションおよび UNKNOWN パーティションの 2 つのテーブルを表します。
- Oracle コードで使用する日付型フォーマット要素は Amazon Redshift の日時フォーマット文字列に類似する: この設定を使用して、TO_CHAR、TO_DATE、TO_NUMBER などのデータ型フォーマット関数を Amazon Redshift がサポートしていない日時フォーマット要素に変換します。デフォルトでは、DMS は拡張パック関数を使用して、変換されたコード内のサポートされていないフォーマット要素をエミュレートします。

  Oracle の日時フォーマットモデルには、Amazon Redshift の日時フォーマット文字列よりも多くの要素が含まれています。ソースコードに Amazon Redshift がサポートする日時形式の要素のみが含まれている場合、変換されたコードで拡張パック関数を使用しないように、この値を設定します。拡張関数を使用しないほうが、変換されたコードの実行速度が速くなります。
- Oracle コードで使用する数値フォーマット要素は、Amazon Redshift の数値フォーマット文字列に類似する: この設定を使用して、Amazon Redshift がサポートしていない数値データ型フォーマット関数を変換します。デフォルトでは、DMS は拡張パック関数を使用して、変換されたコード内のサポートされていないフォーマット要素をエミュレートします。

  Oracle の数値フォーマットモデルには、Amazon Redshift の数値フォーマット文字列よりも多くの要素が含まれています。ソースコードに Amazon Redshift がサポートする数値フォーマット要素のみが含まれている場合、変換されたコードで拡張パック関数を使用しないように、この値を設定します。拡張関数を使用しないほうが、変換されたコードの実行が速くなります。

- NVL 関数を使用して Oracle の LEAD 関数と LAG 関数の動作をエミュレートする: ソースコードで LEAD 関数および LAG 関数のオフセットのデフォルト値を使用しない場合、DMS は NVL 関数を使用してこれらの関数をエミュレートできます。デフォルトでは、DMS は、LEAD 関数と LAG 関数を使用するたびにアクション項目を生成します。NVL を使用してこれらの関数をエミュレートすると、変換されたコードの実行速度が速くなります。
- プライマリキーとユニークキーの動作をエミュレートする: この設定により、DMS はターゲット Amazon Redshift クラスターでのプライマリキーとユニークキーの制約の動作をエミュレートします。Amazon Redshift はプライマリキーやユニークキーの制約を強制せず、これらのキーを情報提供のみを目的として使用します。ソースコードでプライマリキーまたはユニークキーの制約を使用する場合は、この設定を使用して DMS が制約の動作をエミュレートするように設定します。
- 圧縮エンコードを使用する: この設定により、Amazon Redshift テーブル列に圧縮エンコードを適用します。DMS は、Redshift のデフォルトアルゴリズムを使用して圧縮エンコードを自動的に割り当てます。圧縮エンコードの詳細については、「Amazon Redshift データベースデベロッパーガイド」の「圧縮エンコード」を参照してください。

  デフォルトでは、Amazon Redshift はソートキーと分散キーとして定義されている列に圧縮を適用しません。これらの列に圧縮を適用するには、[KEY 列で圧縮エンコードを使用する] を設定します。このオプションは、[圧縮エンコードを使用] を設定した場合にのみ選択できます。

DMS Schema Conversion でのソースとしての PostgreSQL データベースの使用

DMS Schema Conversion では、PostgreSQL データベースを移行ソースとして使用できます。

DMS Schema Conversion を使用して、データベースコードオブジェクトを PostgreSQL データベースから次のターゲットに変換できます。

- MySQL
- Aurora MySQL

ソースとして PostgreSQL に必要な権限を以下に示します。

- CONNECT ON DATABASE <database_name>
- USAGE ON SCHEMA <database_name>
- SELECT ON ALL TABLES IN SCHEMA <database_name>
- SELECT ON ALL SEQUENCES IN SCHEMA <database_name>

DMS Schema Conversion でのソースとしての MySQL データベースの使用

DMS Schema Conversion では、MySQL データベースを移行ソースとして使用できます。

DMS Schema Conversion を使用して、データベースコードオブジェクトを MySQL データベースから次のターゲットに変換できます。

- PostgreSQL
- Aurora PostgreSQL

ソースとして MySQL に必要な権限を以下に示します。

- `SELECT ON *.*`
- `SHOW VIEW ON *.*`

MySQL から PostgreSQL への変換設定

DMS スキーマ変換を編集する方法については、「移行プロジェクトのスキーマ変換設定の指定」を参照してください。

MySQL から PostgreSQL への変換設定には以下が含まれます。

- 変換された SQL コードのコメント : この設定を設定して、選択した重要度以上のアクション項目の変換されたコードにコメントを追加します。

  有効値:
  - エラーのみ
  - エラーおよび警告
  - すべてのメッセージ

DMS Schema Conversion でのターゲットデータプロバイダーの作成

DMS Schema Conversion の移行プロジェクトでは、MySQL データベースや PostgreSQL データベースをターゲットデータプロバイダーとして使用できます。ターゲットデータプロバイダーは、Amazon EC2 インスタンス、Amazon RDS インスタンス、または Amazon Aurora インスタンスにすることができます。

トピック

- DMS Schema Conversion でのターゲットとしての MySQL データベースの使用
- DMS Schema Conversion でのターゲットとしての PostgreSQL データベースの使用
- DMS スキーマ変換のターゲットとしての Amazon Redshift クラスターの使用

DMS Schema Conversion でのターゲットとしての MySQL データベースの使用

DMS Schema Conversion では、MySQL データベースを移行のターゲットとして使用できます。

サポート対象のターゲットデータベースの詳細については、「DMS Schema Conversion のターゲットデータプロバイダー」を参照してください。

MySQL をターゲットとする場合の権限

MySQL をターゲットとして使用するには、次の権限が必要です。

- `CREATE ON *.*`
- `ALTER ON *.*`
- `DROP ON *.*`
- `INDEX ON *.*`
- `REFERENCES ON *.*`
- `SELECT ON *.*`
- `CREATE VIEW ON *.*`
- `SHOW VIEW ON *.*`
- `TRIGGER ON *.*`
- `CREATE ROUTINE ON *.*`
- `ALTER ROUTINE ON *.*`
- `EXECUTE ON *.*`
- `CREATE TEMPORARY TABLES ON *.*`
- `AWS_LAMBDA_ACCESS`
- `INSERT, UPDATE ON AWS_ORACLE_EXT.*`

- INSERT, UPDATE, DELETE ON AWS_ORACLE_EXT_DATA.*
- INSERT, UPDATE ON AWS_SQLSERVER_EXT.*
- INSERT, UPDATE, DELETE ON AWS_SQLSERVER_EXT_DATA.*
- CREATE TEMPORARY TABLES ON AWS_SQLSERVER_EXT_DATA.*

次のコード例を使用すると、データベースユーザーを作成して、権限を付与できます。

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON *.* TO 'user_name';
GRANT AWS_LAMBDA_ACCESS TO 'user_name';
GRANT INSERT, UPDATE ON AWS_ORACLE_EXT.* TO 'user_name';
GRANT INSERT, UPDATE, DELETE ON AWS_ORACLE_EXT_DATA.* TO 'user_name';
GRANT INSERT, UPDATE ON AWS_SQLSERVER_EXT.* TO 'user_name';
GRANT INSERT, UPDATE, DELETE ON AWS_SQLSERVER_EXT_DATA.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON AWS_SQLSERVER_EXT_DATA.* TO 'user_name';
```

前述の例では、[*user_name*] をお客様の設定のユーザー名に置き換えます。次に、*your_password* を安全なパスワードに置き換えます。

Amazon RDS for MySQL または Aurora MySQL をターゲットとして使用するには、lower_case_table_names パラメータを 1 に設定します。この値は、MySQL サーバーがテーブル、インデックス、トリガー、データベースなどのオブジェクト名の識別子を、大文字と小文字を区別せずに処理することを意味します。ターゲットインスタンスでバイナリログを有効にしている場合は、log_bin_trust_function_creators パラメータを 1 と設定します。この場合、ストアド関数を作成するのに、DETERMINISTIC 特性、READS SQL DATA 特性、NO SQL 特性を使用する必要はありません。これらのパラメータを設定するには、新しい DB パラメータグループを作成するか、既存の DB パラメータグループを変更します。

DMS Schema Conversion でのターゲットとしての PostgreSQL データベースの使用

DMS Schema Conversion では、PostgreSQL データベースを移行のターゲットとして使用できます。

サポート対象のターゲットデータベースの詳細については、「DMS Schema Conversion のターゲットデータプロバイダー」を参照してください。

PostgreSQL をターゲットとする場合の権限

PostgreSQL をターゲットとして使用するには、DMS Schema Conversion に `CREATE ON DATABASE` 権限が必要です。DMS Schema Conversion の移行プロジェクトで使用するデータベースごとに、ユーザーを作成し、このユーザーにこの権限を付与します。

Amazon RDS for PostgreSQL をターゲットとして使用するには、DMS Schema Conversion に `rds_superuser` ロールが必要です。

変換したパブリックシノニムを使用するには、次のコマンドを使用してデータベースのデフォルト検索パスを変更します。

```
ALTER DATABASE <db_name> SET SEARCH_PATH = "$user", public_synonyms, public;
```

この例では、*<db_name>* プレースホルダーをデータベース名に置き換えます。

PostgreSQL では、スキーマを削除できるのはスキーマ所有者または `superuser` のみです。スキーマ所有者が一部のオブジェクトを所有していない場合でも、スキーマとスキーマに含まれるすべてのオブジェクトを削除できます。

別のユーザーを使用して別のスキーマを変換してターゲットデータベースに適用すると、DMS Schema Conversion がスキーマを削除できないというエラーメッセージが表示されることがあります。このエラーメッセージを回避するには、`superuser` ロールを使用します。

DMS スキーマ変換のターゲットとしての Amazon Redshift クラスターの使用

DMS スキーマ変換では、Amazon Redshift データベースを移行のターゲットとして使用できます。サポート対象のターゲットデータベースの詳細については、「DMS Schema Conversion のターゲットデータプロバイダー」を参照してください。

ターゲットとしての Amazon Redshift に関する権限

DMS スキーマ変換のターゲットとして Amazon Redshift を使用するには以下の権限が必要です。

- CREATE ON DATABASE: データベースに新しいスキーマを作成することを DMS に許可します。
- CREATE ON SCHEMA: データベーススキーマにオブジェクトを作成することを DMS に許可します。
- GRANT USAGE ON LANGUAGE: データベースに新しい関数やプロシージャを作成することを DMS に許可します。
- GRANT SELECT ON ALL TABLES IN SCHEMA pg_catalog: Amazon Redshift クラスターに関するシステム情報をユーザーに提供します。
- GRANT SELECT ON pg_class_info: テーブル分散スタイルに関する情報をユーザーに提供します。

次のコード例を使用してデータベースユーザーを作成し、アクセス許可を付与できます。例の値は、独自の値に置き換えます。

```
CREATE USER user_name PASSWORD your_password;
GRANT CREATE ON DATABASE db_name TO user_name;
GRANT CREATE ON SCHEMA schema_name TO user_name;
GRANT USAGE ON LANGUAGE plpythonu TO user_name;
GRANT USAGE ON LANGUAGE plpgsql TO user_name;
GRANT SELECT ON ALL TABLES IN SCHEMA pg_catalog TO user_name;
GRANT SELECT ON pg_class_info TO user_name;
GRANT SELECT ON sys_serverless_usage TO user_name;
GRANT SELECT ON pg_database_info TO user_name;
GRANT SELECT ON pg_statistic TO user_name;
```

変換したコードを適用するか、データを移行するターゲットスキーマごとに、GRANT CREATE ON SCHEMA の操作を繰り返します。

ターゲットの Amazon Redshift データベースに拡張パックを適用できます。拡張パックは、オブジェクトを Amazon Redshift に変換するときに必要なソース データベース機能をエミュレートするアドオンモジュールです。詳細については、「DMS Schema Conversion での拡張パックの使用」を参照してください。

DMS スキーマ変換での移行プロジェクトの管理

インスタンスプロファイルとスキーマ変換用の互換性のあるデータプロバイダーを作成した後、移行プロジェクトを作成します。詳細については、「 移行プロジェクトの作成」を参照してください。

この新しいプロジェクトを DMS Schema Conversion で使用するには、[移行プロジェクト] ページのリストからプロジェクトを選択します。次に、[スキーマ変換] タブで、[スキーマ変換を起動] をクリックします。

DMS Schema Conversion を初めて起動するには、多少のセットアップが必要です。AWS Database Migration Service (AWS DMS) がスキーマ変換インスタンスを起動します。これには最長 15 分かかります。このプロセスでは、ソースデータベースとターゲットデータベースからメタデータも読み取ります。初回起動が正常に完了した後の DMS Schema Conversion へのアクセスは迅速になります。

Amazon では、移行プロジェクトが使用したスキーマ変換インスタンスはプロジェクト完了の 3 日後に終了します。DMS Schema Conversion のために使用する Amazon S3 バケットから、変換後のスキーマと評価レポートを取得できます。

DMS Schema Conversion の移行プロジェクト設定の指定

移行プロジェクトを作成してスキーマ変換を起動した後、移行プロジェクト設定を指定できます。コンバージョン設定を変更して、変換済みのコードのパフォーマンスを向上させることができます。また、スキーマ変換ビューをカスタマイズすることもできます。

コンバージョン設定は、ソースデータベースプラットフォームとターゲットデータベースプラットフォームによって異なります。詳細については、 ソースデータプロバイダーの作成 および ターゲットデータプロバイダーの作成 を参照してください。

ソースデータベースペインとターゲットデータベースペインに表示するスキーマとデータベースを指定するには、ツリービュー設定を使用します。空のスキーマ、空のデータベース、システムデータベース、ユーザー定義のデータベースやスキーマは、非表示にできます。

ツリービューのデータベースやスキーマを非表示にするには

1. AWS Management Console にサインインし、AWS DMS コンソール (https://console.aws.amazon.com/dms/v2/) を開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. 移行プロジェクトを選択して、[スキーマ変換] タブで、[スキーマ変換を起動] をクリックします。

4. [設定] を選択します。[設定] ページが開きます。
5. [ツリービュー] セクションで、次のとおり設定します。

   - 空のスキーマを非表示にするには、[空のスキーマを非表示] を選択します。
   - 空のデータベースを非表示にするには、[空のデータベースを非表示] を選択します。
   - [システムデータベースまたはスキーマ] で、非表示にするシステムデータベースやスキーマを選択します。
   - [ユーザー定義のデータベースまたはスキーマ] には、非表示にするユーザー定義のデータベース名とスキーマ名を入力します。[Add] (追加) を選択します。名前では大文字小文字は区別されません。

     複数のデータベースやスキーマを追加するには、名前をカンマで区切ります。類似の名前のオブジェクトを複数追加するには、パーセント (%) をワイルドカードとして使用します。このワイルドカードは、データベース名またはスキーマ名に含まれる任意の数の記号を置き換えます。

   [ソース] セクションと [ターゲット] セクションについても同じ手順を繰り返します。
6. [適用]、[スキーマ変換] の順に選択します。

DMS Schema Conversion でのデータベース移行評価レポートの作成

スキーマの変換をサポートするために生成されるレポートは、DMS Schema Conversion の重要な部分です。このデータベース移行評価レポートは、すべてのスキーマ変換タスクの概要を提供しています。ターゲット DB インスタンスの DB エンジンに変換できないスキーマのアクション項目の詳細も提供しています。このレポートは、AWS DMS コンソールで確認できます。レポートのコピーを PDF またはコンマ区切り値 (CSV) ファイルとして保存することもできます。

移行評価レポートでは、次が提供されます。

- エグゼクティブサマリー
- サーバーオブジェクトの変換、バックアップの提案、リンクサーバーの変更などのレコメンデーション

DMS Schema Conversion が自動的に変換できない項目がある場合、ターゲット DB インスタンスに同等のコードを記述するために必要な作業量の見積りがレポートに表示されます。

トピック

データベース移行評価レポートの作成

移行プロジェクトを作成した後、次の手順でデータベース移行評価レポートを作成します。

データベース移行評価レポートを作成するには

1. AWS Management Console にサインインして、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. [移行プロジェクト]、[スキーマ変換] の順に選択します。
4. [スキーマ変換を起動] をクリックします。[スキーマ変換] ページが開きます。
5. ソースデータベースペインで、評価するデータベーススキーマまたはスキーマ項目を選択します。レポートに複数のオブジェクトを含めるには、すべての項目を選択したことを確認します。
6. 評価するスキーマオブジェクトすべてのチェックボックスをオンにした後、選択したオブジェクトの親ノードを選択する必要があります。ソースデータベースペインで [アクション] メニューが利用できるようになりました。
7. [アクション] メニューで、[評価] を選択します。確認のダイアログボックスが開きます。
8. ダイアログボックスの [評価] をクリックして、この選択を確定します。

データベース移行評価レポートの確認

評価レポートを作成した後、DMS Schema Conversion は次のタブに情報を追加します。

- 概要
- アクション項目

[概要] タブには、DMS Schema Conversion が自動的に変換できる項目の数が表示されます。

[アクション項目] タブには、DMS Schema Conversion が自動的に変換できない項目と、そのような項目の対処方法に関するレコメンデーションが表示されます。

評価レポートの概要

[概要] タブには、データベース移行評価レポートの概要情報が表示されます。DMS Schema Conversion がデータベースストレージオブジェクトとデータベースコードオブジェクトについて自動的に変換できる項目の数が表示されています。

ほとんどの場合、DMS Schema Conversion が自動的にターゲットデータベースエンジンに変換するのはスキーマ項目すべてではありません。[概要] タブには、ソース内のスキーマ項目と同等のスキーマ項目をターゲット DB インスタンスで作成するために必要な作業量の見積りが表示されます。

テーブル、シーケンス、制約、データ型などのデータベースストレージオブジェクトの変換の概要を確認するには、[データベースストレージオブジェクト] をクリックします。

プロシージャ、関数、ビュー、トリガーなどのデータベースコードオブジェクトの変換の概要を確認するには、[データベースコードオブジェクト] をクリックします。

評価レポートの範囲を変更するには、ソースデータベースツリーで必要なノードを選択します。選択した範囲に応じて、DMS Schema Conversion は評価レポートの概要を更新します。

評価レポートのアクション項目

[アクション項目] タブには、DMS Schema Conversion がターゲットデータベースエンジンと互換性のある形式に自動的に変換できない項目のリストが表示されています。DMS Schema Conversion は、アクション項目ごとに問題の説明と推奨アクションを提供しています。DMS Schema Conversion では、類似のアクション項目がグループ化されて、発生回数が表示されます。

関連するデータベースオブジェクトのコードを表示するには、リスト内のアクション項目を選択します。

データベース移行評価レポートの保存

データベース移行評価レポートを作成した後、データベース移行評価レポートのコピーを PDF ファイルまたはカンマ区切り値 (CSV) ファイルのいずれかとして保存できます。

データベース移行評価レポートを PDF ファイルとして保存するには

1. [エクスポート] をクリックして、[PDF] を選択します。ダイアログボックスを確認して、[PDF にエクスポート] を選択します。

2. DMS Schema Conversion は、PDF ファイルを使用してアーカイブを作成し、このアーカイブを Amazon S3 バケットに保存します。Amazon S3 バケットを変更するには、インスタンスプロファイルのスキーマ変換設定を編集します。
3. Amazon S3 バケットにある評価レポートファイルを開きます。

データベース移行評価レポートを CSV ファイルとして保存するには

1. [エクスポート] をクリックして、[CSV] を選択します。ダイアログボックスを確認して、[CSV にエクスポート] を選択します。
2. DMS Schema Conversion は、CSV ファイルを使用してアーカイブを作成し、このアーカイブを Amazon S3 バケットに保存します。Amazon S3 バケットを変更するには、インスタンスプロファイルのスキーマ変換設定を編集します。
3. Amazon S3 バケットにある評価レポートファイルを開きます。

この PDF ファイルには、概要とアクション項目の情報の両方が記載されています。

評価レポートを CSV にエクスポートすると、DMS Schema Conversion は CSV ファイルを 3 つ作成します。

最初の CSV ファイルには、アクション項目に関する次の情報が提供されています。

- カテゴリ
- 頻度
- アクション項目
- 件名
- グループ
- 説明
- ドキュメントリファレンス
- 推奨されるアクション
- 行
- ポジション
- ソース
- ターゲット

- サーバー IP アドレスとポート
- データベース
- スキーマ

2 番目の CSV ファイル名には Action_Items_Summary というサフィックスがついており、次の情報を提供します。

- スキーマ
- アクション項目
- 発生回数
- 学習曲線の労力 (各アクション項目を変換するためのアプローチを設計するために必要な作業量)
- アクション項目の単一の発生を変換するための労力 (設計されたアプローチに沿って各アクション項目を変換するのに必要な作業量)
- アクション項目の説明
- 推奨されるアクション

必要となる作業量のレベルを示すために使用される値は、低 (最小) から高 (最高) までの加重スケールに基づいています。

3 番目の CSV ファイル名には Summary が含まれており、次の情報を提供します。

- カテゴリ
- オブジェクト数
- 自動的に変換されるオブジェクト
- シンプルなアクションを持つオブジェクト
- 複雑度が中程度のアクションを持つオブジェクト
- 複雑なアクションを持つオブジェクト
- 総コード行数

DMS Schema Conversion の使用

DMS Schema Conversion は、既存のデータベーススキーマとほとんどのデータベースコードオブジェクトをターゲットデータベースと互換性のある形式に変換します。

DMS Schema Conversion を使用すると、オンライントランザクション処理 (OLTP) データベーススキーマを Amazon RDS for MySQL または RDS for PostgreSQL に変換するプロセスの多くを自動化します。ソースデータベースエンジンとターゲットデータベースエンジンにはさまざまな機能があり、DMS Schema Conversion は可能な限り同等のスキーマの作成を試みます。直接変換できないデータベースオブジェクトの場合、DMS Schema Conversion は実行できるアクションのリストを提供します。

データベーススキーマを変換するには、次のプロセスを使用します。

- データベーススキーマを変換する前に、変換中にデータベースオブジェクト名を変更する変換ルールを設定する。
- データベース移行評価レポートを作成して、移行の複雑さを推定する。このレポートは、DMS Schema Conversion が自動的に変換できないスキーマ要素に関する詳細を提供する。
- ソースデータベースストレージとコードオブジェクトを変換する。DMS Schema Conversion が変換したデータベースオブジェクトのローカルバージョンを作成する。このように変換したオブジェクトには、移行プロジェクトでアクセスできる。
- 変換済みのコードを SQL ファイルに保存して、変換アクション項目を確認、編集、または対処する。必要に応じて、変換したコードをターゲットデータベースに直接適用します。

データウェアハウススキーマを変換するには、デスクトップ を使用します AWS Schema Conversion Tool。詳細については、「AWS Schema Conversion Tool ユーザーガイド」の「データウェアハウススキーマの Amazon Redshift への変換」を参照してください。

トピック

- DMS Schema Conversion での変換ルールの設定
- DMS Schema Conversion でのデータベーススキーマの変換
- 移行プロジェクトのスキーマ変換設定の指定
- DMS Schema Conversion でのデータベーススキーマの更新
- DMS Schema Conversion での変換したコードの保存と適用

DMS Schema Conversion での変換ルールの設定

DMS Schema Conversion を使用してデータベーススキーマを変換する前に、変換ルールを設定できます。変換ルールを使用すると、オブジェクト名の小文字または大文字への変更、プレフィックスやサフィックスの追加または削除、オブジェクト名の変更ができます。例えば、`test_TABLE_NAME`

という名前のソーススキーマにテーブルがあるとします。ターゲットスキーマ内にある プレフィックス `test_` をプレフィックス `demo_` に変更するルールを設定できます。

変換ルールを作成して、次のとおりのタスクを実行できます。

- プレフィックスの追加、削除、または置換
- サフィックスの追加、削除、または置換
- 列のデータ型の変更
- オブジェクト名の小文字または大文字への変更
- オブジェクトの名前変更

次のオブジェクトの変換ルールを作成できます。

- Schema
- テーブル
- 列

変換ルールの作成

移行プロジェクトの一環として、DMS Schema Conversion は変換ルールを保存します。変換ルールは、移行プロジェクトの作成時に設定することも、後で編集することもできます。

プロジェクトには複数の変換ルールを追加できます。DMS Schema Conversion は変換中に、変換ルールが追加された順番通りにルールを適用します。

変換ルールを作成するには

1. [移行プロジェクトの作成] ページで、[変換ルールの追加] をクリックします。詳細については、「 移行プロジェクトの作成」を参照してください。
2. [ルールターゲット] では、このルールを適用するデータベースオブジェクトのタイプを選択します。
3. [ソーススキーマ] では、[スキーマの入力] を選択します。次に、このルールを適用するソーススキーマ、テーブル、列の名前を入力します。正確な名前を入力して特定のオブジェクトを選択するか、パターンを入力して複数のオブジェクトを選択できます。パーセント (%) をワイルドカードとして使用すると、データベースオブジェクト名に含まれる任意の数の記号を置き換えることができます。
4. [アクション] では、実行するタスクを選択します。

5. ルールタイプに応じて、追加の値を 1 つまたは 2 つ入力します。例えば、オブジェクトの名前を変更するには、オブジェクトの新しい名前を入力します。プレフィックスを置換するには、現在のプレフィックスおよび置換後のプレフィックスを入力します。
6. [変換ルールの追加] をクリックすると、別の変換ルールを追加できます。

   ルールの追加が完了したら、[移行プロジェクトを作成] をクリックします。

既存の変換ルールを複製するには、[複製] を選択します。既存の変換ルールを編集するには、リストからルールを選択します。既存の変換ルールを削除するには、[削除] を選択します。

変換ルールの編集

移行プロジェクトでは、新しい変換ルールの追加、既存の変換ルールの削除、または編集ができます。DMS Schema Conversion は、スキーマ変換の開始時に変換ルールを適用するため、ルールを編集した後は必ずスキーマ変換を閉じてから再起動します。

変換ルールを編集するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択して、編集する移行プロジェクトを選択します。
3. [スキーマ変換] を選択して、[スキーマ変換を閉じる] をクリックします。
4. がスキーマ変換を AWS DMS 閉じたら、変更 を選択して移行プロジェクト設定を編集します。
5. [変換ルール] で、次のいずれかのアクションを選択します。

   - 既存の変換ルールを複製して、リストの最後に追加するには、[複製] を選択する。
   - 既存の変換ルールを削除するには、[削除] を選択する。
   - 編集するには、既存の変換ルールを選択する。
6. ルールの編集を完了した後、[変更を保存] を選択します。
7. [移行プロジェクト] ページのリストでプロジェクトを選択します。[スキーマ変換] を選択して、[スキーマ変換を起動] をクリックします。

DMS Schema Conversion でのデータベーススキーマの変換

移行プロジェクトを作成して、ソースデータベースとターゲットデータベースに接続した後、ソースデータベースオブジェクトをターゲットデータベースと互換性のある形式に変換できます。DMS

Schema Conversion は、ソースデータベースのスキーマを左側のパネルにツリービュー形式で表示します。

データベースツリーの各ノードは遅延ロードされます。ユーザーがツリービューでノードを選択した時点で、DMS Schema Conversion はソースデータベースからスキーマ情報をリクエストします。スキーマ情報のロード速度を上げるには、スキーマを選択して、[アクション] メニューから [メタデータをロード] を選択します。DMS Schema Conversion は次に、データベースのメタデータを読み取り、その情報を Amazon S3 バケットに保存します。これで、データベースオブジェクトの閲覧がより迅速になります。

変換は、データベーススキーマ全体で行うことも、ソースデータベースから任意のスキーマ項目を選択して行うこともできます。選択したスキーマ項目が親項目に依存する場合、DMS Schema Conversion はその親項目のスキーマも生成します。例えば、変換するテーブルを選択すると、DMS Schema Conversion は変換したテーブルと、そのテーブルが格納されているデータベーススキーマを作成します。

データベースオブジェクトの変換

DMS Schema Conversion を使用すると、データベーススキーマ全体を変換することも、個別のデータベーススキーマオブジェクトを変換することもできます。

データベーススキーマ全体を変換するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. [移行プロジェクト]、[スキーマ変換] の順に選択します。
4. [スキーマ変換を起動] をクリックします。[スキーマ変換] ページが開きます。
5. ソースデータベースペインで、スキーマ名のチェックボックスをオンにします。
6. 移行プロジェクトの左側のペインで、このスキーマを選択します。DMS Schema Conversion はスキーマ名を青色で強調表示して、[アクション] メニューをアクティブ化します。
7. [アクション] で [変換] を選択します。[変換] ダイアログボックスが表示されます。
8. ダイアログボックスの [変換] をクリックして、この選択を確定します。

ソースデータベースオブジェクトを変換するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. [移行プロジェクト]、[スキーマ変換] の順に選択します。
4. [スキーマ変換を起動] をクリックします。[スキーマ変換] ページが開きます。
5. ソースデータベースペインで、ソースデータベースオブジェクトを選択します。
6. 変換するオブジェクトのチェックボックスをすべてオンにした後、左側のパネルで選択したすべてのオブジェクトの親ノードを選択します。

   DMS Schema Conversion は親ノードを青色で強調表示して、[アクション] メニューをアクティブ化します。
7. [アクション] で [変換] を選択します。[変換] ダイアログボックスが表示されます。
8. ダイアログボックスの [変換] をクリックして、この選択を確定します。

例えば、10 のテーブルのうち 2 つを変換するには、変換する 2 つのテーブルのチェックボックスをオンにします。[アクション] メニューが非アクティブであることに注意します。[テーブル] ノードを選択すると、DMS Schema Conversion はテーブル名を青色で強調表示して、[アクション] メニューをアクティブ化します。その後、このメニューから [変換] を選択できます。

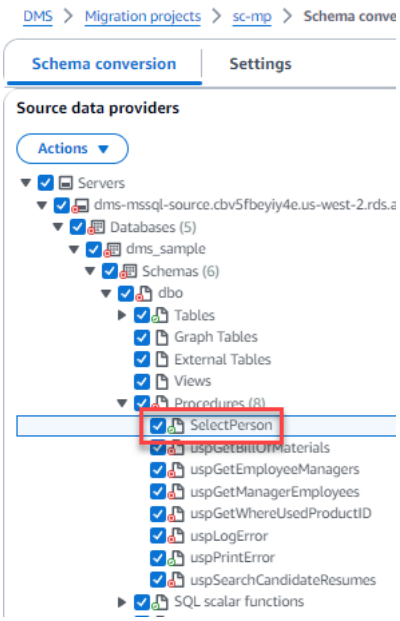
2 つのテーブルと 3 つのプロシージャを変換する場合も同様に、オブジェクト名のチェックボックスをオンにします。次に、スキーマノードを選択して [アクション] メニューをアクティブ化し、[スキーマを変換] を選択します。

変換された SQL コードの編集と保存

スキーマ変換ページでは、データベースオブジェクト内の統合 SQL コードを編集できます。以下の手順を使用して、変換された SQL コードを編集、変更適用、保存します。

変換された SQL コードを編集、変更適用、保存するには

1. スキーマ変換ページで、ソースデータプロバイダーペインのツリービューを開き、コードオブジェクトを表示します。



2. ソースデータプロバイダーペインで、アクション 、変換 を選択します。アクションを確認します。

3. 変換が完了したら、変換された SQL を表示するには、必要に応じて中央ペインを展開します。変換された SQL を編集するには、ターゲット SQL ペインの編集アイコンを選択します。

4. ターゲット SQL を編集したら、ページ上部のチェックアイコンを選択して変更を確認します。アクションを確認します。

5. ターゲットデータプロバイダーペインで、アクション 、変更の適用 を選択します。アクションを確認します。

6. DMS は、編集された手順をターゲットデータストアに書き込みます。

変換したデータベースオブジェクトの確認

ソースデータベースオブジェクトを変換した後、プロジェクトの左側のペインでオブジェクトを選択できます。これで、そのオブジェクトのソースと変換済みのコードを確認できます。DMS Schema Conversion は、左側のペインで選択したオブジェクトの変換済みのコードを自動的にロードします。選択したオブジェクトのプロパティやパラメータも確認できます。

移行プロジェクトの一環として、DMS Schema Conversion は変換したコードを自動的に保存します。このようなコード変更のターゲットデータベースへの適用は行いません。変換したコードをターゲットデータベースに適用する方法の詳細については、「変換したコードの適用」を参照してください。変換したコードを移行プロジェクトから削除するには、右側のペインでターゲットスキーマを選択して、[アクション] から [データベースから更新] を選択します。

ソースデータベースオブジェクトを変換すると、中央下部のペインに変換の概要とアクション項目が表示されます。評価レポートを作成した場合も同じ情報を確認できます。評価レポートは、DMS Schema Conversion が変換できないスキーマ項目を特定して解決するうえで役立ちます。評価レポートの概要と変更の概要アクション項目のリストは CSV ファイルに保存できます。詳細については、「データベース移行評価レポート」を参照してください。

移行プロジェクトのスキーマ変換設定の指定

移行プロジェクトを作成した後、DMS Schema Conversion でコンバージョン設定を指定できます。スキーマ変換設定を設定すると、変換済みのコードのパフォーマンスが向上します。

コンバージョン設定を編集するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. 移行プロジェクトを選択します。[スキーマ変換] 選択して、[スキーマ変換を起動] をクリックします。
4. [設定] を選択します。[設定] ページが開きます。
5. [変換] セクションで、設定を変更します。
6. [適用]、[スキーマ変換] の順に選択します。

すべての変換ペアについて、アクション項目を含め、変換済みのコードのコメント数を制限できます。変換済みのコード内のコメント数を制限するには、移行プロジェクトのコンバージョン設定を開きます。

[Comments in converted SQL code] で、アクション項目の重大度レベルを選択します。DMS Schema Conversion は、選択した重大度以上のアクション項目について、変換済みのコードにコメントを追加します。例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。

変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。

その他のコンバージョン設定は、ソースデータベースとターゲットデータベースのペアにより異なります。

トピック

- Oracle から MySQL へのコンバージョン設定
- Oracle から PostgreSQL へのコンバージョン設定
- SQL Server から MySQL へのコンバージョン設定
- SQL Server から PostgreSQL へのコンバージョン設定
- PostgreSQL から MySQL への変換設定
- DB2 for z/OS から DB2 LUW への変換設定

Oracle から MySQL へのコンバージョン設定

DMS Schema Conversion の Oracle から MySQL へのコンバージョン設定は、次のとおりです。

- ソースの Oracle データベースでは、ROWID 疑似列を使用できるが、MySQL は同様の機能をサポートしていない。DMS Schema Conversion は、変換したコードで ROWID 疑似列をエミュレートできる。このためには、[行 ID を生成] オプションをオンにする。

  ソースの Oracle コードが ROWID 疑似列を使用していない場合は、[行 ID を生成] オプションをオフにする。この場合、変換したコードの処理が迅速化する。
- ソースの Oracle のコードには、MySQL ではサポートされないパラメータを使用する TO_CHAR 関数、TO_DATE 関数、TO_NUMBER 関数が使用されている場合がある。デフォルトでは、DMS Schema Conversion は変換したコード内のこのようなパラメータの使用をエミュレートする。

  ソースの Oracle コードに MySQL でサポートされないパラメータがない場合は、MySQL のネイティブの TO_CHAR 関数、TO_DATE 関数、TO_NUMBER 関数を使用できる。この場合、変換したコードの処理が迅速化する。これには、次の値を選択する。
  - Use a native MySQL TO_CHAR function

  - Use a native MySQL TO_DATE function
  - Use a native MySQL TO_NUMBER function
- データベースとアプリケーションは別々のタイムゾーンで実行できる。デフォルトでは、DMS Schema Conversion は変換したコードのタイムゾーンをエミュレートする。ただし、データベースとアプリケーションが同じタイムゾーンを使用している場合は、このエミュレーションは必要ない。この場合、[Improve the performance of the converted code where the database and applications use the same time zone] を選択する。

Oracle から PostgreSQL へのコンバージョン設定

DMS Schema Conversion の Oracle から PostgreSQL へのコンバージョン設定は、次のとおりです。

- AWS DMS は、Oracle マテリアライズドビューを PostgreSQL のテーブルまたはマテリアライズドビューに変換できます。[マテリアライズドビュー] で、ソースのマテリアライズドビューの変換方法を選択する。
- ソースの Oracle データベースでは、ROWID 疑似列を使用できるが、PostgreSQL は同様の機能をサポートしていない。DMS Schema Conversion は、bigint または character varying データ型を使用して、変換したコードで ROWID 疑似列をエミュレートできる。このためには、[行 ID] で [Use the bigint data type to emulate the ROWID pseudocolumn] または [Use the character varying data type to emulate the ROWID pseudocolumn] を選択する。

  ソースの Oracle コードが ROWID 疑似列を使用していない場合は、[Don't generate] を選択する。この場合、変換したコードの処理が迅速化する。
- ソースの Oracle のコードには、PostgreSQL ではサポートされないパラメータを使用する TO_CHAR 関数、TO_DATE 関数、TO_NUMBER 関数が使用されている場合がある。デフォルトでは、DMS Schema Conversion は変換したコード内のこのようなパラメータの使用をエミュレートする。

  ソースの Oracle コードに PostgreSQL でサポートされないパラメータがない場合は、PostgreSQL のネイティブの TO_CHAR 関数、TO_DATE 関数、TO_NUMBER 関数を使用できる。この場合、変換したコードの処理が迅速化する。これには、次の値を選択する。
  - Use a native PostgreSQL TO_CHAR function
  - Use a native PostgreSQL TO_DATE function
  - Use a native PostgreSQL TO_NUMBER function

- データベースとアプリケーションは別々のタイムゾーンで実行できる。デフォルトでは、DMS Schema Conversion は変換したコードのタイムゾーンをエミュレートする。ただし、データベースとアプリケーションが同じタイムゾーンを使用している場合は、このエミュレーションは必要ない。この場合、[Improve the performance of the converted code where the database and applications use the same time zone] を選択する。
- 変換したコードで引き続きシーケンスを使用するには、[Populate converted sequences with the last value generated on the source side] を選択する。
- 場合によっては、ソース Oracle で、NUMBER データ型のプライマリキー列または外部キー列に整数値のみを格納していることがある。このような場合、 はこれらの列を BIGINT データ型に変換 AWS DMS できます。このアプローチを採用すると、変換したコードのパフォーマンスが向上する。これを行うには、[Convert primary and foreign key columns of the NUMBER data type to the BIGINT data type] を選択する。データの損失を避けるため、ソースのこれらの列に浮動小数点値が含まれていないことを確認する。
- ソースコード内の非アクティブ化されたトリガーと制約をスキップするには、[アクティブなトリガーと制約のみを変換] を選択する。
- DMS Schema Conversion を使用すると、動的 SQL として呼び出される文字列変数を変換できる。このような文字列変数の値は、データベースコードで変更できる。 AWS DMS が常にこの文字列変数の最新の値を変換するようにするには、「呼び出されたルーチン で作成された動的 SQL コードを変換する」を選択します。
- PostgreSQL バージョン 10 以前のバージョンでは、プロシージャをサポートしていない。PostgreSQL でのプロシージャの使用に慣れていない場合は、Oracle プロシージャを PostgreSQL 関数に変換 AWS DMS できます。このためには、[プロシージャを関数に変換] を選択する。
- 発生したアクション項目に関する追加情報を確認するには、拡張パックに特定の関数を追加する。このためには、[Add extension pack functions that raise user-defined exceptions] を選択する。次に、ユーザー定義の例外を発生させる重大度レベルを選択する。ソースデータベースオブジェクトを変換した後は、必ず拡張パックスキーマを適用する。拡張パックの詳細については、「拡張パックの使用」を参照する。
- ソースの Oracle データベースでは、自動的に生成された名前を使用した制約が使用されている場合がある。ソースコードでこのような名前を使用している場合は、必ず [Keep the names of system generated constraints] を選択する。ソースコードでこのような制約が使用されていても、名前は使用されていない場合は、このオプションをオフにすると変換速度が向上する。
- ソースデータベースとターゲットデータベースが別々のタイムゾーンで実行されている場合、Oracle の SYSDATE 組み込み関数をエミュレートする関数がソース関数とは異なる値を返す。

ソース関数とターゲット関数が同じ値を返すようにするには、[ソースデータベースのタイムゾーンを設定] を選択する。

- 変換したコードでは、orafce 拡張機能の関数を使用できる。このためには、[Oracle 組み込みルーチン] で使用する関数を選択する。orafce の詳細については、「」の「orafce」を参照してください GitHub。

SQL Server から MySQL へのコンバージョン設定

DMS Schema Conversion の SQL Server から MySQL へのコンバージョン設定は、次のとおりです。

- ソースの SQL Server データベースは、EXEC の出力をテーブルに保存できる。DMS Schema Conversion は、一時テーブルと、この機能をエミュレートする追加のプロシージャが作成する。このエミュレーションを使用するには、 [Create additional routines to handle open datasets] を選択する。

SQL Server から PostgreSQL へのコンバージョン設定

DMS Schema Conversion の SQL Server から PostgreSQL へのコンバージョン設定は、次のとおりです。

- SQL Server では、複数のテーブルで同じインデックス名を使用できるが、PostgreSQL では、スキーマで使用するインデックス名がすべて一意である必要がある。DMS Schema Conversion がすべてのインデックスのために固有の名前を生成するようにするには、[インデックスの一意の名前を生成] を選択する。
- PostgreSQL バージョン 10 以前のバージョンでは、プロシージャをサポートしていない。PostgreSQL でのプロシージャの使用に慣れていない場合は、SQL Server プロシージャを PostgreSQL 関数に変換 AWS DMS できます。このためには、[プロシージャを関数に変換] を選択する。
- ソースの SQL Server データベースは、EXEC の出力をテーブルに保存できる。DMS Schema Conversion は、一時テーブルと、この機能をエミュレートする追加のプロシージャが作成する。このエミュレーションを使用するには、 [Create additional routines to handle open datasets] を選択する。
- 変換されるコード内のスキーマ名に使用するテンプレートを定義できる。[スキーマ名] で、次のオプションのいずれかを選択する。
  - DB – SQL Server のデータベース名を PostgreSQL のスキーマ名として使用する。

  - SCHEMA – SQL Server のスキーマ名を PostgreSQL のスキーマ名として使用する。
  - DB_SCHEMA – SQL Server データベースとスキーマ名の組み合わせを PostgreSQL のスキーマ名として使用する。
- ソースオブジェクト名の大文字と小文字の区別は維持できる。オブジェクト名が小文字に変換されないようにするには、[オブジェクト名に同じ大文字と小文字を使用] を選択する。このオプションは、ターゲットデータベースで大文字と小文字を区別するオプションをオンにした場合にのみ適用される。
- ソースデータベースのパラメータ名は維持できる。DMS Schema Conversion は、変換されるコードのパラメータ名に二重引用符を追加できる。このためには、[元のパラメータ名を保持] を選択する。
- ソースデータベースのルーチンパラメータの長さを保持できます。DMS スキーマ変換はドメインを作成し、このドメインを使用してルーチンパラメータの長さを指定します。そのためには、[パラメータの長さを保持] を選択します。

PostgreSQL から MySQL への変換設定

DMS Schema Conversion の PostgreSQL から MySQL への変換設定には、次のものがあります。

- 変換された SQL コードのコメント: この設定には、選択した重要度以上のアクション項目の変換されたコードにコメントが含まれます。この設定では、以下の値がサポートされます。
  - エラーのみ
  - エラーおよび警告
  - すべてのメッセージ

DB2 for z/OS から DB2 LUW への変換設定

DMS Schema Conversion の DB2 for z/OS から DB2 LUW への変換設定には、次のものがあります。

- 変換された SQL コードのコメント: この設定には、選択した重要度以上のアクション項目の変換されたコードにコメントが含まれます。この設定では、以下の値がサポートされます。
  - エラーのみ
  - エラーおよび警告
  - すべてのメッセージ

DMS Schema Conversion でのデータベーススキーマの更新

移行プロジェクトを作成すると、DMS Schema Conversion はソーススキーマとターゲットスキーマに関する情報をこのプロジェクトに保存します。DMS Schema Conversion は、遅延ロードを使用して、データベースツリーのノードを選択する際など、必要な場合にのみメタデータをロードします。積極的なロードを使用すると、スキーマ情報のロードが迅速化します。このためには、スキーマを選択して、[アクション] から [メタデータをロード] を選択します。

オブジェクトの移行プロジェクトへの自動または手動ロードが完了したら、DMS Schema Conversion は再度遅延ロードを使用することはありません。そのため、データベース内のテーブルやプロシージャなどのオブジェクトを変更する場合は、必ず移行プロジェクトで更新を実行します。

データベースからスキーマを更新するには、更新するオブジェクトを選択して、[アクション] から[データベースから更新] を選択します。ソースとターゲットのデータベーススキーマの次のデータベースオブジェクトを更新できます。

- ソース – ソースのデータベーススキーマを更新する場合は、[データベースから更新] を選択して、プロジェクトのスキーマをソースデータベースの最新のスキーマに置き換える。
- ターゲット – ターゲットデータベースのスキーマを更新すると、DMS Schema Conversion はプロジェクトのスキーマをターゲットデータベースの最新のスキーマに置き換える。DMS Schema Conversion は、変換済みのコードをターゲットデータベースのコードに置き換える。[データベースから更新] を選択する前に、変換済みのコードをターゲットデータベースに適用したことを確認する。そうしないと、ソースデータベースが再度変換される。

DMS Schema Conversion での変換したコードの保存と適用

DMS Schema Conversion は、ソースデータベースオブジェクトを変換した後、変換したコードをターゲットデータベースにすぐには適用しません。代わりに、DMS Schema Conversion は、ターゲットデータベースに適用する準備が整うまで、変換済みのコードをプロジェクトに保存します。

既存のアクション項目に対処するために、変換したコードを適用する前にソースデータベースコードを更新して、更新済みのオブジェクトをもう一度変換できます。DMS Schema Conversion が自動的に変換できない項目の詳細については、「DMS Schema Conversion でのデータベース移行評価レポートの作成」を参照してください。DMS Schema Conversion の移行プロジェクトでのソースデータベースオブジェクトの更新の詳細については、「データベーススキーマの更新」を参照してください。

DMS Schema Conversion で変換したコードを直接データベースに適用する代わりに、コードを SQL スクリプトとしてファイルに保存することができます。このような SQL スクリプトを確認し、必要に応じて編集してから手動で SQL スクリプトをターゲットデータベースに適用できます。

変換したコードの SQL ファイルへの保存

変換されたスキーマを SQL スクリプトとしてテキストファイルに保存できます。変換したコードを変更して、DMS Schema Conversion が自動的に変換できないアクション項目に対処できます。その後、SQL スクリプトをターゲットデータベースで実行して、変換済みのコードをターゲットデータベースに適用できます。

変換されたスキーマを SQL スクリプトとして保存するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. [移行プロジェクト]、[スキーマ変換] の順に選択します。
4. [スキーマ変換を起動] をクリックします。[スキーマ変換] ページが開きます。
5. 右側のペインで、ターゲットデータベーススキーマを選択するか、保存する変換済みのオブジェクトを選択します。DMS Schema Conversion が親ノード名を青色で強調表示し、ターゲットデータベースの[アクション] メニューがアクティブ化したことを確認します。
6. [アクション] では、[SQL として保存] を選択します。[保存] ダイアログボックスが開きます。
7. [SQL として保存] をクリックして、選択を確定します。

   DMS Schema Conversion は、SQL ファイルを使用してアーカイブを作成し、このアーカイブを Amazon S3 バケットに保存します。
8. (オプション) S3 バケットを変更するには、インスタンスプロファイルのスキーマ変換設定を編集します。
9. S3 バケットから SQL スクリプトを開きます。

変換したコードの適用

変換したコードをターゲットデータベースに適用する準備が整ったら、プロジェクトの右側のペインでデータベースオブジェクトを選択します。変更は、データベーススキーマ全体にも、選択したデータベーススキーマオブジェクトにも適用できます。

データベースオブジェクトを選択すると、DMS Schema Conversion が選択したノード名または親ノード名を青色で強調表示します。次に [アクション] メニューがアクティブ化されます。[アクション] で、[変更を適用] を選択します。表示されたダイアログボックスで [適用] をクリックして、選択を確定し、変換済みのコードをターゲットデータベースに適用します。

拡張パックのスキーマの適用

変換したスキーマをターゲットデータベースに初めて適用する際、DMS Schema Conversion が拡張パックのスキーマを適用する場合もあります。拡張パックのスキーマは、ターゲットデータベース用に変換したコードを実行するために必要なソースデータベースのシステム関数をエミュレートします。変換済みのコードで拡張パックの関数が使用されている場合は、必ず拡張パックのスキーマを適用します。

拡張パックをターゲットデータベースに手動で適用するには、[アクション] で [変更を適用] を選択します。表示されたダイアログボックスで [確認] を選択して、拡張パックをターゲットデータベースに適用します。

変換したコードが原因で予期しない結果になるのを避けるため、拡張パックのスキーマは変更しないことをお勧めします。

詳細については、「DMS Schema Conversion での拡張パックの使用」を参照してください。

DMS Schema Conversion での拡張パックの使用

DMS Schema Conversion の拡張パックは、ターゲットデータベースでサポートされていないソースデータベース関数をエミュレートするアドオンのモジュールです。拡張パックを使用すると、変換されたコードがソースコードと同じ結果を生成することを確認できます。拡張パックをインストールする前に、データベーススキーマを変換します。

各拡張パックにはデータベーススキーマが含まれています。このスキーマは、特定のオンライン トランザクション処理 (OLTP) オブジェクトやサポートされていない組み込み関数をソースデータベースからエミュレートするための SQL 関数、プロシージャ、テーブル、ビューを提供します。

ソースデータベースを変換すると、DMS Schema Conversion がターゲットデータベースに追加のスキーマを追加します。このスキーマは、ターゲットデータベース用に変換したコードを実行するために必要なソースデータベースの SQL システム関数を実装します。この追加のスキーマは、拡張パックスキーマと呼ばれます。

拡張パックのスキーマは、ソースデータベースに応じて次のとおりの名前が付けられています。

- Microsoft SQL Server – `aws_sqlserver_ext`
- Oracle – `aws_oracle_ext`

拡張パックは次の 2 つの方法で適用できます。

- DMS Schema Conversion では、変換されたコードを適用する際に拡張パックを自動的に適用できます。DMS Schema Conversion は、その他のすべてのスキーマオブジェクトを適用する前に拡張パックを適用します。
- 拡張パックは手動で適用できます。これを行うには、ターゲットデータベースツリーで拡張パックスキーマを選択して、[適用]、[拡張パックを適用] の順に選択します。

を使用したデータベースの Amazon RDS と同等のものへの移行 AWS DMS

AWS Database Migration Service（ AWS DMS) での同種データ移行により、セルフマネージド型のオンプレミスデータベースから Amazon Relational Database Service (Amazon RDS) に相当するデータベースへの移行が簡素化されます。例えば、同種データ移行を使用して、オンプレミスの PostgreSQL データベースを Amazon RDS for PostgreSQL または Aurora PostgreSQL に移行できます。同種データ移行の場合、 AWS DMS はネイティブデータベースツールを使用して、簡単でパフォーマンスの高い like-to-like 移行を提供します。

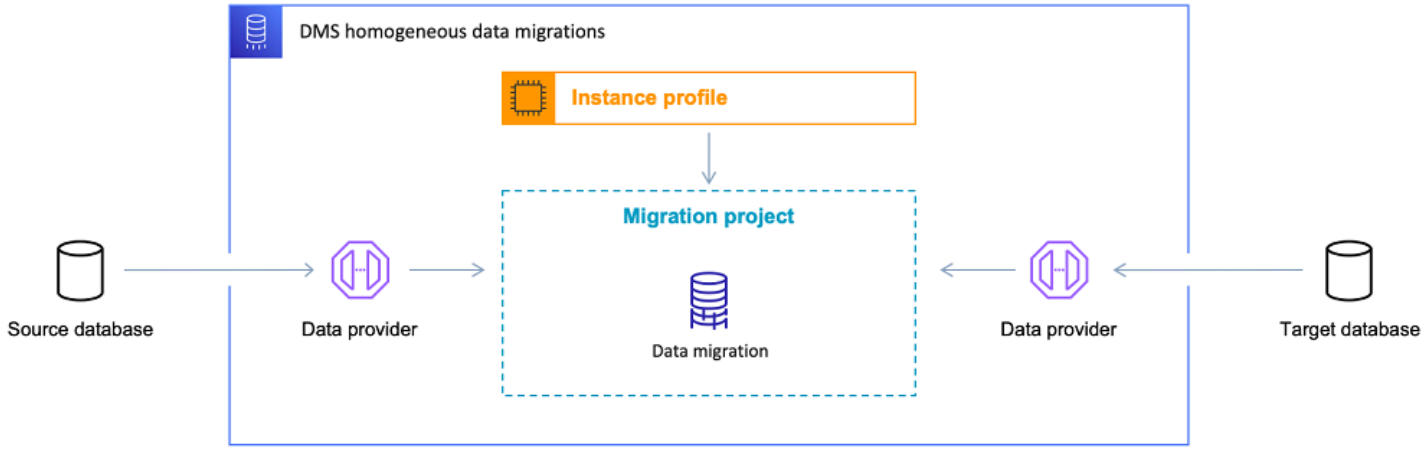
同種データ移行はサーバーレスです。つまり、 は移行に必要なリソース AWS DMS を自動的にスケーリングします。同種データ移行では、データ、テーブルパーティション、データ型、関数、また、ストアドプロシージャなどのセカンダリオブジェクトを移行できます。

概括すると、同種データ移行は、インスタンスプロファイル、データプロバイダー、移行プロジェクトを使用して行われます。互換性のあるソースデータプロバイダーとターゲットデータプロバイダーが同じタイプの移行プロジェクトを作成すると、 はデータ移行を実行するサーバーレス環境を AWS DMS デプロイします。次に、ソースデータプロバイダー AWS DMS に接続し、ソースデータを読み取り、ディスクにファイルをダンプして、ネイティブデータベースツールを使用してデータを復元します。インスタンスプロファイル、データプロバイダー、移行プロジェクトの詳細については、「でのデータプロバイダー、インスタンスプロファイル、移行プロジェクトの使用 AWS DMS」を参照してください。

サポート対象のソースデータベースの一覧については、「DMS 同種データ移行のソース」を参照してください。

サポート対象のターゲットデータベースの一覧については、「DMS 同種データ移行のターゲット」を参照してください。

次の図は、同種データ移行の仕組みを説明しています。



次のセクションでは、同種データ移行の使用について説明します。

トピック

- サポート対象 AWS リージョン
- 機能
- 同種データ移行の制限
- AWS DMS の同種データ移行プロセスの概要
- AWS DMS での同種データ移行のセットアップ
- での同種データ移行用のソースデータプロバイダーの作成 AWS DMS
- での同種データ移行のターゲットデータプロバイダーの作成 AWS DMS
- での同種データ移行の実行 AWS DMS
- AWS DMS での同種データ移行のトラブルシューティング

サポート対象 AWS リージョン

同種データ移行は、次の で実行できます AWS リージョン。

リージョン名	リージョン
米国東部 (バージニア北部)	us-east-1
米国東部 (オハイオ)	us-east-2
米国西部 (オレゴン)	us-west-2

リージョン名	リージョン
アジアパシフィック (東京)	ap-northeast-1
アジアパシフィック (シンガポール)	ap-southeast-1
アジアパシフィック (シドニー)	ap-southeast-2
欧州 (フランクフルト)	eu-central-1
欧州 (ストックホルム)	eu-north-1
欧州 (アイルランド)	eu-west-1

機能

同種データ移行では、次の機能が利用できます。

- AWS DMS は、同種データ移行に必要な 内の AWS クラウド コンピューティングリソースとストレージリソースを自動的に管理します。 は、データ移行を開始するときに、これらのリソースをサーバーレス環境に AWS DMS デプロイします。
- AWS DMS は、ネイティブデータベースツールを使用して、同じタイプのデータベース間で完全に自動化された移行を開始します。
- 同種データ移行を使用すると、データのみでなく、パーティション、関数、ストアドプロシージャなどのセカンダリオブジェクトも移行できます。
- 同種データ移行は、フルロード、継続的なレプリケーション、継続的なレプリケーションを実行するフルロードの 3 種類の移行モードで実行できます。
- 同種データ移行では、オンプレミス、Amazon EC2、Amazon RDS データベースをソースとして使用できます。同種データ移行では、移行ターゲットとして Amazon RDS または Amazon Aurora を選択できます。

同種データ移行の制限

同種データ移行を使用する場合、次のとおりの制限が適用されます。

- 同種データ移行は、MongoDB および Amazon DocumentDB 移行の選択ルールのみをサポートします。DMS は、他のデータベースエンジンの選択ルールをサポートしていません。また、変換ルールを使用した列のデータ型の変更、オブジェクトのスキーマの移動、オブジェクト名の変更はできません。
- 同種データ移行では、データ検証用の組み込みツールは提供されません。
- PostgreSQL で同種データ移行を使用する場合、 はビューをテーブルとしてターゲットデータベース AWS DMS に移行します。
- 同種データ移行の継続的データレプリケーションでは、スキーマレベルの変更はキャプチャされません。ソースデータベースに新しいテーブルを作成すると、 AWS DMS はこのテーブルを移行できません。新たに作成したテーブルを移行するには、データ移行をもう一度開始します。
- で同種データ移行を使用して AWS DMS 、上位のデータベースバージョンから下位のデータベースバージョンにデータを移行することはできません。
- CLI と API では同種データ移行を利用できません。
- 同種データ移行では、VPC セカンダリ CIDR 範囲内のデータベースインスタンスとの接続の確立はサポートされていません。
- 8081 ポートは、データプロバイダーからの同種移行には使用できません。
- 同種データ移行は、暗号化された MySQL データベースとテーブルの移行をサポートしていません。

AWS DMS の同種データ移行プロセスの概要

AWS DMS の同種データ移行を使用すると、同じタイプの 2 つのデータベース間でデータを移行できます。データ移行を作成して実行するには、次のワークフローを使用します。

1. 必要となる AWS Identity and Access Management (IAM) ポリシーとルールを作成します。詳細については、「IAM リソースの作成」を参照してください。
2. ソースデータベースとターゲットデータベースを設定して、AWS DMS での同種データ移行に必要となる最小限のアクセス権限でデータベースユーザーを作成します。詳細については、「ソースデータプロバイダーの作成」と「ターゲットデータプロバイダーの作成」を参照してください。
3. ソースデータベースとターゲットデータベースの認証情報は、AWS Secrets Manager に保存します。詳細については、「AWS Secrets Manager ユーザーガイド」の「ステップ 1: シークレットを作成する」を参照してください。

4. AWS DMS コンソールで、サブネットグループ、インスタンスプロファイル、データプロバイダーを作成します。詳細については、「サブネットグループの作成」、「インスタンスプロファイルの作成」、「データプロバイダーの作成」を参照してください。
5. 前のステップで作成したリソースを使用して、移行プロジェクトを作成します。詳細については、「移行プロジェクトの作成」を参照してください。
6. データ移行を作成、設定、開始します。詳細については、「データ移行の作成」を参照してください。
7. フルロードまたは継続的なレプリケーションが完了したら、カットオーバーして新しいターゲットデータベースの使用を開始できます。
8. リソースをクリーンアップします。Amazon は、移行完了後 3 日以内に移行プロジェクトでのデータ移行を終了します。ただし、インスタンスプロファイル、データプロバイダー、IAM ポリシーとロール、AWS Secrets Manager のシークレットなどのリソースは手動で削除する必要があります。

AWS DMS での同種データ移行の詳細については、PostgreSQL to Amazon RDS for PostgreSQL のステップバイステップの移行ウォークスルーを参照してください。

次の動画 では、AWS DMS の同種データ移行のユーザーフェイスとこの機能の概要を紹介しています。

AWS DMS での同種データ移行のセットアップ

AWS DMS で同種データ移行をセットアップするには、次の前提条件タスクを実行します。

トピック

- AWS DMS での同種データ移行に必要な IAMリソースの作成
- AWS DMS での同種データ移行のためのネットワークのセットアップ

AWS DMS での同種データ移行に必要な IAMリソースの作成

同種データ移行を実行するには、その他の AWS サービスと連携するための IAM ポリシーと IAM ロールをアカウントに作成する必要があります。このセクションでは、このような必要となる IAM リソースを作成します。

トピック

- AWS DMS での同種データ移行のための IAM ポリシーの作成

- AWS DMS での同種データ移行のための IAM ロールの作成

AWS DMS での同種データ移行のための IAM ポリシーの作成

データベースにアクセスしてデータを移行するために、AWS DMS は同種データ移行用のサーバーレス環境を作成します。AWS DMS はこの環境で、VPC ピアリング、ルートテーブル、セキュリティグループ、その他の AWS リソースにアクセスする必要があります。また、AWS DMS は、各データ移行のログ、メトリクス、進行状況を Amazon CloudWatch に保存します。データ移行プロジェクトを作成するには、AWS DMS がこのようなサービスにアクセスする必要があります。

このステップでは、AWS DMS に Amazon EC2 と CloudWatch のリソースへのアクセスを付与する IAM ポリシーを作成します。その後、IAM ロールを作成して、このポリシーをアタッチします。

AWS DMS での同種データ移行のために IAM ポリシーを作成するには

1. AWS Management Console にサインインして、https://console.aws.amazon.com/iam/ で Amazon IAM コンソールを開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. [ポリシーの作成] をクリックします。
4. [ポリシーの作成] ページで、[JSON] タブをクリックします。
5. 次の JSON コードをエディタに貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeVpcs",
                "ec2:DescribePrefixLists",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "servicequotas:GetServiceQuota"
            ],
            "Resource": "arn:aws:servicequotas:*:*:vpc/L-0EA8095F"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*:log-
stream:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateRoute",
                "ec2:DeleteRoute"
            ],
            "Resource": "arn:aws:ec2:*:*:route-table/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:security-group-rule/*",
                "arn:aws:ec2:*:*:route-table/*",
                "arn:aws:ec2:*:*:vpc-peering-connection/*",
```

```
                "arn:aws:ec2:*:*:vpc/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group-rule/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AcceptVpcPeeringConnection",
                "ec2:ModifyVpcPeeringConnectionOptions"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-peering-connection/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:AcceptVpcPeeringConnection",
            "Resource": "arn:aws:ec2:*:*:vpc/*"
        }
    ]
}
```

6. [次へ: タグ] をクリックしてから、[次へ: 確認] をクリックします。
7. [名前*] には **HomogeneousDataMigrationsPolicy** と入力して、[ポリシーを作成] をクリックします。

AWS DMS での同種データ移行のための IAM ロールの作成

このステップでは、AWS DMS に AWS Secrets Manager、Amazon EC2、CloudWatch へのアクセスを付与する IAM ロールを作成します。

AWS DMSでの同種データ移行用の IAM ロールを作成するには

1. AWS Management Console にサインインして、https://console.aws.amazon.com/iam/ で Amazon IAM コンソールを開きます。
2. ナビゲーションペインで [ロール] を選択します。
3. [ロールの作成] をクリックします。
4. [信頼されたエンティティを選択] ページの [信頼されたエンティティタイプ] では、AWS[サービス] を選択します。[その他の AWS サービスのユースケース] では、[DMS] を選択します。
5. [DMS] チェックボックスをオンにして、[次へ] をクリックします。
6. [許可を追加] ページで、作成した [HomogeneousDataMigrationsPolicy] を選択します。[SecretsManagerReadWrite] も選択します。[次へ] をクリックします。
7. [名前、確認、および作成] ページで、[ロール名] に **HomogeneousDataMigrationsRole** と入力して、[ロールの作成] をクリックします。
8. [ロール] ページの [ロール名] には、**HomogeneousDataMigrationsRole** と入力します。[HomogeneousDataMigrationsRole] を選択します。
9. [HomogeneousDataMigrationsRole] ページで、[信頼関係] タブをクリックします。[信頼ポリシーを編集] をクリックします。
10. [信頼ポリシーを編集] ページで、次の JSON コードをエディタに貼り付けて、既存のテキストを置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "dms-data-migrations.amazonaws.com",
                    "dms.your_region.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
```

```
        }
    ]
}
```

上記の例の *your_region* は独自の AWS リージョン ユーザー名に置き換えます。

上記のリソースベースのポリシーは、AWS マネージドの SecretsManagerReadWrite と カスタマーマネージドの HomogeneousDataMigrationsPolicy ポリシーに沿ってタスクを実行するアクセス許可を AWS DMS サービスプリンシパルに付与します。

11. [ポリシーを更新] をクリックします。

AWS DMS での同種データ移行のためのネットワークのセットアップ

AWS DMS は、Amazon VPC サービスを基盤に仮想プライベートクラウド (VPC) 内に同種データ移行のためのサーバーレス環境を作成します。使用する VPC は、インスタンスプロファイルを作成する際に指定します。アカウントと AWS リージョン リージョンのデフォルトの VPC を使用することも、新しい VPC を作成することもできます。

AWS DMS は、データ移行ごとにインスタンスプロファイルに使用する VPC との VPC ピアリング接続を確立します。AWS DMS はその後、インスタンスプロファイルに関連付けられているセキュリティグループに CIDR ブロックを追加します。AWS DMS は、パブリック IP アドレスをインスタンスプロファイルにアタッチするため、同じインスタンスプロファイルを使用するデータ移行にはすべて同じパブリック IP アドレスが割り当てられます。データ移行が停止したり失敗したりすると、AWS DMS は VPC ピアリング接続を削除します。

CIDR ブロックがインスタンスプロファイル VPC の VPC とするのを避けるため、AWS DMS は、`10.0.0.0/8`、`172.16.0.0/12`、`192.168.0.0/16` CIDR ブロック のいずれかの `/24` プレフィックスを使用します。例えば、3 つのデータ移行を並行して実行する場合、AWS DMS は次の CIDR ブロックを使用して VPC ピアリング接続を確立します。

- `192.168.0.0/24` – 最初のデータ移行用
- `192.168.1.0/24` – 2 番目のデータ移行用
- `192.168.2.0/24` – 3 番目のデータ移行用

AWS DMS のソースデータベースとターゲットデータベースとの連携のセットアップには、さまざまなネットワーク構成を使用できます。継続的なデータ レプリケーションの場合は、ソースデータベースとターゲットデータベース間の連携もセットアップする必要があります。このような設定は、

ソースデータプロバイダーの場所とネットワーク設定により異なります。次のセクションでは、一般的なネットワーク設定について説明します。

トピック

- ソースデータプロバイダーとターゲットデータプロバイダーでの単一の VPC の使用
- ソースデータプロバイダーとターゲットデータプロバイダーでの複数の VPC の使用
- オンプレミスのソースデータプロバイダーの使用
- 継続的なレプリケーションの設定

ソースデータプロバイダーとターゲットデータプロバイダーでの単一の VPC の使用

この構成では、AWS DMS はプライベートネットワーク内のソースデータプロバイダーとターゲットデータプロバイダーに接続します。

ソースデータプロバイダーとターゲットデータプロバイダーが同じ VPC に配置されている場合にネットワークを設定するには

1. AWS DMS コンソールで、ソースデータプロバイダーとターゲットデータプロバイダーが使用する VPC とサブネットを使用してサブネットグループを作成します。詳細については、「サブネットグループの作成」を参照。
2. 作成した VPC とサブネットグループを使用して、AWS DMS コンソールでインスタンスプロファイルを作成します。ソースデータプロバイダーとターゲットデータプロバイダーが使用する VPC セキュリティグループも選択します。詳細については、「インスタンスプロファイルの作成」を参照。

この構成では、データ移行にパブリック IP アドレスを使用する必要はありません。

ソースデータプロバイダーとターゲットデータプロバイダーでの複数の VPC の使用

このような構成の場合、AWS DMS はプライベート ネットワークを使用してソースデータプロバイダーまたはターゲットデータプロバイダーに接続します。別のデータプロバイダーの場合、AWS DMS はパブリックネットワークを使用します。どちらのデータプロバイダーがインスタンスプロファイルと同じ VPC に配置されているかにより、次の設定のいずれかを選択します。

ソースデータプロバイダーにプライベートネットワーク、ターゲットデータプロバイダーにはパブリックネットワークを設定するには

1. ソースデータプロバイダーが使用する VPC とサブネットを使用して、AWS DMS コンソールでサブネットグループを作成します。詳細については、「サブネットグループの作成」を参照。
2. 作成した VPC とサブネットグループを使用して、AWS DMS コンソールでインスタンスプロファイルを作成します。ソースデータプロバイダーが使用する VPC セキュリティグループも選択します。詳細については、「インスタンスプロファイルの作成」を参照。
3. 移行プロジェクトを開きます。[データの移行] タブで作業するデータ移行を選択します。[詳細] タブの [接続とセキュリティ] の下に表示されている パブリック IP アドレス をメモしておきます。
4. ターゲットデータベースセキュリティグループのデータ移行のパブリック IP アドレスからのアクセスを許可します。詳細については、「Amazon Relational Database Service ユーザーガイド」の「セキュリティグループによるアクセス制御」を参照してください。

ソースデータプロバイダーにパブリックネットワーク、ターゲットデータプロバイダーにはプライベートネットワークを設定するには

1. ターゲットデータプロバイダーが使用する VPC とサブネットを使用して、AWS DMS コンソールでサブネットグループを作成します。詳細については、「サブネットグループの作成」を参照。
2. 作成した VPC とサブネットグループを使用して、AWS DMS コンソールでインスタンスプロファイルを作成します。ターゲットデータプロバイダーが使用する VPC セキュリティグループも選択します。詳細については、「インスタンスプロファイルの作成」を参照。
3. 移行プロジェクトを開きます。[データの移行] タブで作業するデータ移行を選択します。[詳細] タブの [接続とセキュリティ] の下に表示されている パブリック IP アドレス をメモしておきます。
4. ソースデータベースセキュリティグループのデータ移行のパブリック IP アドレスからのアクセスを許可します。詳細については、「Amazon Relational Database Service ユーザーガイド」の「セキュリティグループによるアクセス制御」を参照してください。

オンプレミスのソースデータプロバイダーの使用

この構成では、AWS DMS はパブリックネットワーク内のソースデータプロバイダーに接続します。AWS DMS は、プライベートネットワークを使用してターゲットデータプロバイダーに接続します。

オンプレミスのソースデータプロバイダーのネットワークを設定するには

1. ターゲットデータプロバイダーが使用する VPC とサブネットを使用して、AWS DMS コンソールでサブネットグループを作成します。詳細については、「サブネットグループの作成」を参照。
2. 作成した VPC とサブネットグループを使用して、AWS DMS コンソールでインスタンスプロファイルを作成します。ターゲットデータプロバイダーが使用する VPC セキュリティグループも選択します。詳細については、「インスタンスプロファイルの作成」を参照。
3. 移行プロジェクトを開きます。[データの移行] タブで作業するデータ移行を選択します。[詳細] タブの [接続とセキュリティ] の下に表示されている パブリック IP アドレス をメモしておきます。
4. AWS DMS のデータ移行のパブリック IP アドレスからソースデータベースへのアクセスを許可します。

AWS DMSは、VPC セキュリティグループにインバウンドルールまたはアウトバウンドルールを作成します。このようなルールを削除するとデータ移行の失敗につながる可能性があるため、削除しないように注意します。VPC セキュリティグループでは、独自のルールを設定できます。ルール管理の観点から、ルールには説明を追加することをお勧めします。

継続的なレプリケーションの設定

[フルロードと変更データキャプチャ (CDC)] または [変更データキャプチャ (CDC)] のタイプのデータ移行を実行するには、ソースデータベースとターゲットデータベース間の接続を許可する必要があります。

パブリックにアクセスできるソースデータベースとターゲットデータベース間の接続を設定するには

1. ソースデータベースとターゲットデータベースのパブリック IP アドレスをメモしておきます。
2. ターゲットデータベースのパブリック IP アドレスからソースデータベースへのアクセスを許可します。
3. ソースデータベースのパブリック IP アドレスからターゲットデータベースへのアクセスを許可します。

単一の VPC 内でプライベートにアクセスできるソースデータベースとターゲットデータベース間の接続を設定するには

1. ソースデータベースとターゲットデータベースのプライベート IP アドレスをメモしておきます。

   > **Important**
   >
   > ソースデータベースとターゲットデータベースが別の VPC 内、または別のネットワークに配置されている場合、ソースデータベースとターゲットデータベースに使用できるのはパブリック IP アドレスのみです。使用できるのは、データプロバイダーのパブリックホスト名または IP アドレスのみです。

2. ターゲットデータベースのプライベート IP アドレスからソースデータベースへのアクセスを許可します。
3. ソースデータベースのプライベート IP アドレスからターゲットデータベースへのアクセスを許可します。

での同種データ移行用のソースデータプロバイダーの作成 AWS DMS

MySQL 互換、PostgreSQL、および MongoDB 互換データベースを 同種データ移行の のソースデータプロバイダーとして使用できます AWS DMS。

サポートされているデータベースバージョンについては、「」を参照してくださいDMS 同種データ移行のソースデータプロバイダー。

ソースデータプロバイダーは、オンプレミス、Amazon EC2、または Amazon RDS データベースにすることができます。

トピック

- での同種データ移行のソースとしての MySQL 互換データベースの使用 AWS DMS
- での同種データ移行のソースとしての PostgreSQL データベースの使用 AWS DMS
- での同種データ移行のソースとしての MongoDB 互換データベースの使用 AWS DMS

での同種データ移行のソースとしての MySQL 互換データベースの使用 AWS DMS

AWS DMS での 同種データ移行 のソースとして、MySQL 互換のデータベース (MySQL または MariaDB) を使用できます。この場合、ソースデータプロバイダーは、オンプレミス、Amazon EC2、RDS for MySQL または MariaDB データベースにすることができます。

同種データ移行を実行するには、レプリケーションのすべてのソーステーブルとセカンダリオブジェクトに対する SELECT 権限を持つデータベースユーザーを使用する必要があります。変更データキャプチャ (CDC) タスクの場合、REPLICATION CLIENT (10.5.2 以降の MariaDBバージョンの場合は BINLOG MONITOR) と REPLICATION SLAVE 権限もユーザーに必要です。フルロードデータ移行の場合、この 2 つの権限は必要ありません。

次のスクリプトを使用して、MySQL データベースでの必要な権限を持つデータベースユーザーを作成します。に移行するすべてのデータベースに対してGRANTクエリを実行します AWS。

```
CREATE USER 'your_user'@'%' IDENTIFIED BY 'your_password';

GRANT REPLICATION SLAVE, REPLICATION CLIENT  ON *.* TO 'your_user'@'%';
GRANT SELECT, RELOAD, LOCK TABLES, SHOW VIEW, EVENT, TRIGGER ON *.* TO 'your_user'@'%';

GRANT BACKUP_ADMIN ON *.* TO 'your_user'@'%';
```

上記の例の各 *user input placeholder* は独自の情報に置き換えます。ソース MySQL データベースのバージョンが 8.0 以前の場合、GRANT BACKUP_ADMIN コマンドはスキップできます。

次のスクリプトを使用して、MariaDB データベースでの必要な権限を持つデータベースユーザーを作成します。に移行するすべてのデータベースに対して GRANT クエリを実行します AWS。

```
CREATE USER 'your_user'@'%' IDENTIFIED BY 'your_password';
GRANT SELECT, RELOAD, LOCK TABLES, REPLICATION SLAVE, BINLOG MONITOR, SHOW VIEW ON  *.*
 TO 'your_user'@'%';
```

上記の例の各 *user input placeholder* は独自の情報に置き換えます。

次のセクションでは、セルフマネージド型の MySQL データベースと AWS が管理する MySQL データベースの設定の具体的な前提条件について説明します。

トピック

- 同種データ移行のソースとしてのセルフマネージド型の MySQL 互換データベースの使用
- での同種データ移行のソースとしての AWSマネージド MySQL 互換データベースの使用 AWS DMS
- MySQL 互換データベースを同種データ移行のソースとして使用する場合の制限

同種データ移行のソースとしてのセルフマネージド型の MySQL 互換データベースの使用

このセクションでは、オンプレミスまたは Amazon EC2 インスタンスでホストされる MySQL 互換データベースの設定方法について説明します。

ソースの MySQL データベースまたは MariaDB データベースのバージョンを確認します。で説明されているように、 がソース MySQL または MariaDB データベースのバージョン AWS DMS をサポートしていることを確認してくださいDMS 同種データ移行のソース。

CDC を使用するには、バイナリログを有効にしてください。バイナリログを有効にするには、MySQL データベースまたは MariaDB データベースの `my.ini` (Windows) または `my.cnf` (UNIX) ファイルで次のパラメータを設定します。

パラメータ	値
`server-id`	このパラメータは、1 以上の値に設定します。
`log-bin`	パスをバイナリログファイル (`log-bin=E:\MySql_Logs\BinLog`) に設定します。ファイル拡張子を含めないでください。
`binlog_format`	このパラメータは `ROW` に設定します。`binlog_format` が `STATEMENT` に設定されているときは場合によっては、ターゲットにデータレプリケーション時に矛盾が生じる可能性があるため、レプリケーション中にこの設定をお勧めします。データベースエンジンが自動的に `STATEMENT` ベースのログに切り替わるため、`binlog_format` が `MIXED` に設定されていると、ターゲットのデータの書き込みで同様の不整合が発生する。
`expire_logs_days`	このパラメータは、1 以上の値に設定します。ディスク容量の使いすぎを防ぐため、デフォルト値の 0 は使用しないことをお勧めします。
`binlog_checksum`	このパラメータは `NONE` に設定します。

パラメータ	値
binlog_row_image	このパラメータは FULL に設定します。
log_slave_updates	MySQL または MariaDB のレプリカをソースとして使用している場合、このパラメータは TRUE と設定する。

での同種データ移行のソースとしての AWSマネージド MySQL 互換データベースの使用 AWS DMS

このセクションでは、Amazon RDS for MySQL データベースインスタンスと Amazon RDS for MariaDB データベースインスタンスを設定する方法について説明します。

で AWSマネージド MySQL または MariaDB データベースを同種データ移行のソースとして使用する場合は AWS DMS、CDC に次の前提条件があることを確認してください。

- RDS for MySQL と RDS for MariaDB のバイナリログを有効にするには、インスタンスレベルで自動バックアップを有効にします。Aurora MySQL クラスターのバイナリログを有効にするには、パラメータグループで変数 binlog_format を変更します。Aurora MySQL クラスターの自動バックアップについては有効にする必要はありません。

  次に、binlog_format パラメータを ROW に設定します。

  自動バックアップの設定の詳細については、「Amazon RDS ユーザーガイド」の「自動バックアップの有効化」を参照してください。

  Amazon RDS for MySQL または MariaDB データベースのバイナリログ設定の詳細については、「Amazon RDS ユーザーガイド」の「バイナリログ形式の設定」を参照してください。

  Aurora MySQL クラスターのバイナリログ設定の詳細については、「Amazon Aurora MySQL クラスターのバイナリログを有効にする方法」をご参照ください。
- バイナリログが で使用可能であることを確認します AWS DMS。 AWSが管理する MySQL および MariaDB データベースはバイナリログをできるだけ早く消去するため、ログが利用可能なままである時間を長くする必要があります。たとえば、ログ保持を 24 時間に伸ばすには、次のコマンドを実行します。

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

- `binlog_row_image` パラメータを `Full` に設定します。
- `binlog_checksum` パラメータを `NONE` に設定します。
- Amazon RDS for MySQL または MariaDB レプリカをソースとして使用する場合は、リードレプリカでバックアップを有効にして、`log_slave_updates` パラメータが `TRUE` と設定されていることを確認します。

MySQL 互換データベースを同種データ移行のソースとして使用する場合の制限

MySQL 互換データベースを同種データ移行のソースとして使用する場合、次の制限が適用されます。

- シーケンスなどの MariaDB オブジェクトは、同種移行タスクではサポートされていません。
- MariaDB から Amazon RDS for MySQL または Aurora MySQL への移行は、互換性のないオブジェクトの差異により失敗する可能性があります。
- データソースへの接続に使用するユーザー名には以下の制限があります。
  - 2 ~ 64 文字を使用できます。
  - スペースは使用できません。
  - 文字として a ~ z、A ~ Z、0 ~ 9、アンダースコア (_) を使用できます。
  - a ~ z または A ~ Z で始める必要があります。
- データソースへの接続に使用するパスワードには以下の制限があります。
  - 1 ~ 128 文字を使用できます。
  - 一重引用符 (')、二重引用符 (")、セミコロン (;)、スペースのいずれも使用できません。

での同種データ移行のソースとしての PostgreSQL データベースの使用 AWS DMS

AWS DMS での 同種データ移行 のソースとして、PostgreSQL データベースを使用できます。この場合、ソースデータプロバイダーは、オンプレミス、Amazon EC2、または RDS for PostgreSQL データベースにすることができます。

同種データ移行を実行するには、PostgreSQL ソースデータベースの で指定したデータベースユーザーにスーパーユーザーアクセス許可を付与 AWS DMS します。データベースユーザーがソース内のレプリケーション固有の機能にアクセスするには、スーパーユーザーのアクセス許可が必要となり

ます。フルロードデータ移行の場合、データベースユーザーにはテーブルを移行するための SELECT 権限が必要です。

次のスクリプトを使用して、PostgreSQL のソースデータベースで必要な権限を持つデータベースユーザーを作成します。に移行するすべてのデータベースに対してGRANTクエリを実行しますAWS。

```
CREATE USER your_user WITH LOGIN PASSWORD 'your_password';
ALTER USER your_user WITH SUPERUSER;
GRANT SELECT ON ALL TABLES IN SCHEMA schema_name TO your_user;
```

上記の例の各 *user input placeholder* は独自の情報に置き換えます。

次のセクションでは、セルフマネージド型の PostgreSQL データベースと AWS が管理する PostgreSQL データベースの設定の具体的な前提条件について説明します。

トピック

- での同種データ移行のソースとしてのセルフマネージド PostgreSQL データベースの使用 AWS DMS
- での同種データ移行のソースとしての AWSマネージド PostgreSQL データベースの使用 AWS DMS
- PostgreSQL 互換データベースを同種データ移行のソースとして使用する場合の制限

での同種データ移行のソースとしてのセルフマネージド PostgreSQL データベースの使用 AWS DMS

このセクションでは、オンプレミスまたは Amazon EC2 インスタンスでホストされる PostgreSQL データベースの設定方法について説明します。

ソースの PostgreSQL データベースのバージョンを確認します。「」の説明に従って、 がソース PostgreSQL データベースのバージョン AWS DMS をサポートしていることを確認してくださいDMS 同種データ移行のソース。

同種データ移行は、論理レプリケーションを使用した変更データキャプチャ (CDC) をサポートします。セルフマネージド型 PostgreSQL ソースデータベースの論理レプリケーションを有効にするには、postgresql.conf の設定ファイルで次のパラメータと値を設定します。

- wal_level を logical に設定します。
- max_replication_slots を 1 より大きい値に設定します。

max_replication_slots 値は、実行するタスクの数に従って設定してください。たとえば、5 つのタスクを実行するには、少なくとも 5 つのスロットを設定する必要があります。スロットは、タスクが開始するとすぐに自動的に開き、タスクが実行されなくなった場合でも開いたままです。開いているスロットは手動で削除してください。

- max_wal_senders を 1 より大きい値に設定します。

  max_wal_senders パラメータは、実行可能な同時タスクの数を設定します。

- wal_sender_timeout パラメータは、指定されたミリ秒数が過ぎても非アクティブなレプリケーション接続を終了します。デフォルト値は 60,000 ミリ秒 (1 秒)。値を 0 (ゼロ) に設定するとタイムアウトメカニズムが無効になる。この設定は DMS では有効。

一部のパラメータは静的であり、サーバースタート時にのみ設定できます。設定ファイルへの変更は、サーバーが再起動されるまで無視されます。詳細については、PostgreSQL ドキュメントをご参照ください。

での同種データ移行のソースとしての AWSマネージド PostgreSQL データベースの使用 AWS DMS

このセクションでは、Amazon RDS for PostgreSQL データベースインスタンスを設定する方法について説明します。

での同種データ移行の PostgreSQL ソースデータプロバイダーのユーザーアカウントとして、PostgreSQL DB インスタンスの AWS マスターユーザーアカウントを使用します AWS DMS。マスターユーザーアカウントには、CDC を設定するために必要なロールがあります。管理ユーザーアカウント以外のアカウントを使用する場合、アカウントに rds_superuser ロールと rds_replication ロールが必要です。rds_replication ロールは、論理スロットを管理し、論理スロットを使用してデータをストリーミングするアクセス権許可付与します。

次のコード例を使用して、rds_superuser ロールと rds_replication ロールを付与します。

```
GRANT rds_superuser to your_user;
GRANT rds_replication to your_user;
```

上記の例の *your_user* は、使用するユーザー名に置き換えます。

論理レプリケーションを有効にするには、DB パラメータグループの rds.logical_replication パラメータを 1 に設定します。この静的パラメータを有効にするには、DB インスタンスを再起動する必要があります。

PostgreSQL 互換データベースを同種データ移行のソースとして使用する場合の制限

PostgreSQL 互換データベースを同種データ移行のソースとして使用する場合、次の制限が適用されます。

- データソースへの接続に使用するユーザー名には以下の制限があります。
  - 2 ~ 64 文字を使用できます。
  - スペースは使用できません。
  - 文字として a ~ z、A ~ Z、0 ~ 9、アンダースコア (_) を使用できます。
  - a ~ z または A ~ Z で始める必要があります。
- データソースへの接続に使用するパスワードには以下の制限があります。
  - 1 ~ 128 文字を使用できます。
  - 一重引用符 (')、二重引用符 (")、セミコロン (;)、スペースのいずれも使用できません。

での同種データ移行のソースとしての MongoDB 互換データベースの使用 AWS DMS

MongoDB 互換データベースを での同種データ移行のソースとして使用できます AWS DMS。この場合、ソースデータプロバイダーは、オンプレミスの Amazon EC2 for MongoDB データベースまたは Amazon DocumentDB (MongoDB 互換) データベースにすることができます。

サポートされているデータベースバージョンについては、「」を参照してくださいDMS 同種データ移行のソースデータプロバイダー。

以下のセクションでは、セルフマネージド MongoDB データベースと AWSマネージド Amazon DocumentDB データベースの特定の設定前提条件について説明します。

トピック

- での同種データ移行のソースとしてのセルフマネージド MongoDB データベースの使用 AWS DMS
- での同種データ移行のソースとしての Amazon DocumentDB データベースの使用 AWS DMS
- MongoDB 互換データベースを同種データ移行のソースとして使用する機能
- MongoDB 互換データベースを同種データ移行のソースとして使用する場合の制限
- MongoDB 互換データベースを同種データ移行のソースとして使用するためのベストプラクティス

での同種データ移行のソースとしてのセルフマネージド MongoDB データベースの使用 AWS DMS

このセクションでは、オンプレミスまたは Amazon EC2 インスタンスでホストされている MongoDB データベースを設定する方法について説明します。

ソース MongoDB データベースのバージョンを確認します。「」の説明に従って、 がソース MongoDB データベースのバージョン AWS DMS をサポートしていることを確認してくださいDMS 同種データ移行のソースデータプロバイダー。

MongoDB ソースで同種データ移行を実行するには、ルート権限を持つユーザーアカウントを作成するか、移行するデータベースに対するアクセス許可のみを持つユーザーを作成します。ユーザー作成の詳細については、「」を参照してくださいAWS DMS のソースとして MongoDB を使用する場合に必要なアクセス許可。

MongoDB で継続的なレプリケーションまたは CDC を使用するには、 に MongoDB オペレーションログ (oplog) へのアクセス AWS DMS が必要です。詳細については、「CDC 用 MongoDB レプリカセットの設定」を参照してください。

MongoDB 認証方法の詳細については、「」を参照してくださいAWS DMS のソースとして MongoDB を使用する場合のセキュリティ要件。

ソースとしての MongoDB の場合、同種データ移行は Amazon DocumentDB がサポートするすべてのデータ型をサポートします。

MongoDB をソースとして Secrets Manager にユーザー認証情報を保存するには、その他のタイプのシークレットタイプを使用してプレーンテキストでユーザー認証情報を提供する必要があります。詳細については、「シークレットを使用して AWS Database Migration Service エンドポイントにアクセスするには」を参照してください。

次のコードサンプルは、プレーンテキストを使用してデータベースシークレットを保存する方法を示しています。

```
{
  "username": "dbuser",
  "password": "dbpassword"
}
```

での同種データ移行のソースとしての Amazon DocumentDB データベースの使用 AWS DMS

このセクションでは、同種データ移行のソースとして使用する Amazon DocumentDB データベースインスタンスを設定する方法について説明します。

Amazon DocumentDB インスタンスのマスターユーザー名を、 での同種データ移行のための MongoDB 互換ソースデータプロバイダーのユーザーアカウントとして使用します AWS DMS。マスターユーザーアカウントには、CDC を設定するために必要なロールがあります。マスターユーザーアカウント以外のアカウントを使用する場合、そのアカウントにはルートロールが必要です。ルートアカウントとしてのユーザー作成の詳細については、「」を参照してくださいソースとして Amazon DocumentDB を使用するためのアクセス許可の設定。

論理レプリケーションを有効にするには、データベースchange_stream_log_retention_durationパラメータグループの パラメータを、トランザクションワークロードに適した設定に設定します。この静的パラメータを変更するには、DB インスタンスを再起動して有効にする必要があります。フルロードのみを含むすべてのタスクタイプのデータ移行を開始する前に、特定のデータベース内のすべてのコレクション、または選択したコレクションに対してのみ Amazon DocumentDB 変更ストリームを有効にします。Amazon DocumentDB の変更ストリームの有効化の詳細については、「Amazon Amazon DocumentDB デベロッパーガイド」の「変更ストリームの有効化」を参照してください。

> **Note**
>
> AWS DMS は Amazon DocumentDB 変更ストリームを使用して、継続的なレプリケーション中に変更をキャプチャします。DMS がレコードを読み取る前に Amazon DocumentDB が変更ストリームからレコードをフラッシュすると、タスクは失敗します。少なくとも 24 時間変更を保持するように change_stream_log_retention_durationパラメータを設定することをお勧めします。

Amazon DocumentDB を同種データ移行に使用するには、Amazon DocumentDB データベースの認証情報 の Secrets Manager にユーザー認証情報を保存します。

MongoDB 互換データベースを同種データ移行のソースとして使用する機能

- Amazon DocumentDB が全ロードフェーズでサポートするすべてのセカンダリインデックスを移行できます。

- AWS DMS はコレクションを並行して移行します。同種データ移行は、パフォーマンスを最大限に高めるために、コレクション内の各ドキュメントの平均サイズに基づいて実行時にセグメントを計算します。
- DMS は、CDC フェーズで作成したセカンダリインデックスをレプリケートできます。DMS は、MongoDB バージョン 6.0 でこの機能をサポートしています。
- DMS は、ネストレベルが 97 を超えるドキュメントをサポートします。

MongoDB 互換データベースを同種データ移行のソースとして使用する場合の制限

- ドキュメントに$プレフィックスが付いたフィールド名を含めることはできません。
- AWS DMS は、時系列収集の移行をサポートしていません。
- AWS DMS は、CDC フェーズ中の `create`、`drop`、または `rename collection` DDL イベントをサポートしていません。
- AWS DMS は、 `_id`フィールドのコレクション内の一貫性のないデータ型をサポートしていません。例えば、次のサポートされていないコレクションには、 `_id`フィールドに対して複数のデータ型があります。

```
rs0 [direct: primary] test> db.collection1.aggregate([
...   {
...     $group: {
...       _id: { $type: "$_id" },
...       count: { $sum: 1 }
...     }
...   }
... ])
[ { _id: 'string', count: 6136 }, { _id: 'objectId', count: 848033 } ]
```

- CDC のみのタスクの場合、 は `immediate`開始モード AWS DMS のみをサポートします。
- AWS DMS は、無効な UTF8 文字を含むドキュメントをサポートしていません。
- AWS DMS はシャードコレクションをサポートしていません。

MongoDB 互換データベースを同種データ移行のソースとして使用するためのベストプラクティス

- 同じ MongoDB インスタンスでホストされている複数の大規模なデータベースとコレクションについては、データベースとコレクションごとに選択ルールを使用して、タスクを複数のデータ移行タ

スクとプロジェクトに分割することをお勧めします。データベースとコレクションの分割を調整して、パフォーマンスを最大限に高めることができます。

での同種データ移行のターゲットデータプロバイダーの作成 AWS DMS

MySQL 互換、PostgreSQLおよび Amazon DocumentDB データベースを、 での同種データ移行のターゲットデータプロバイダーとして使用できます AWS DMS。

サポートされているデータベースのバージョンについては、「」を参照してくださいDMS 同種データ移行のターゲットデータプロバイダー。

ターゲットデータプロバイダーは、Amazon RDS DB インスタンス、Amazon Aurora DB クラスターにすることができます。ターゲットデータプロバイダーのデータベースバージョンは、ソースデータプロバイダーのデータベースバージョン以上である必要があります。

トピック

- での同種データ移行のターゲットとしての MySQL 互換データベースの使用 AWS DMS
- での同種データ移行のターゲットとしての PostgreSQL データベースの使用 AWS DMS
- での同種データ移行のターゲットとしての Amazon DocumentDB データベースの使用 AWS DMS

での同種データ移行のターゲットとしての MySQL 互換データベースの使用 AWS DMS

AWS DMSでの同種データ移行の移行ターゲットとして、MySQL 互換データベースを使用できます。

AWS DMS では、ターゲット Amazon RDS for MySQL、MariaDB、または Amazon Aurora MySQL データベースにデータを移行するための特定のアクセス許可が必要です。次のスクリプトを使用して、ターゲットの MySQL データベースで必要な権限を持つデータベースユーザーを作成します。

```
CREATE USER 'your_user'@'%' IDENTIFIED BY 'your_password';

GRANT ALTER, CREATE, DROP, INDEX, INSERT, UPDATE, DELETE, SELECT, CREATE VIEW, CREATE
 ROUTINE, ALTER ROUTINE, EVENT, TRIGGER, EXECUTE, REFERENCES ON *.* TO 'your_user'@'%';
GRANT REPLICATION SLAVE, REPLICATION CLIENT  ON *.* TO 'your_user'@'%';
```

上記の例の各 *user input placeholder* は独自の情報に置き換えます。

次のスクリプトを使用して、MariaDB データベースでの必要な権限を持つデータベースユーザーを作成します。に移行するすべてのデータベースに対して GRANT クエリを実行します AWS。

```
CREATE USER 'your_user'@'%' IDENTIFIED BY 'your_password';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, CREATE
 ROUTINE, ALTER ROUTINE, EVENT, TRIGGER, EXECUTE,SLAVE MONITOR, REPLICATION SLAVE ON
 *.* TO 'your_user'@'%';
```

上記の例の各 *user input placeholder* は独自の情報に置き換えます。

> **Note**
>
> Amazon RDS では、MySQL または Maria データベースインスタンスの自動バックアップを有効にすると、バイナリログも有効になります。この設定が有効になっている場合、ターゲットデータベースで関数、プロシージャ、トリガーなどのセカンダリオブジェクトを作成する差異に、データ移行タスクが次のようなエラーで失敗することがあります。ターゲットデータベースでバイナリログが有効になっている場合は、タスクを開始する前にデータベースパラメータグループで`log_bin_trust_function_creators` を `true` に設定します。
>
> ```
> ERROR 1419 (HY000): You don't have the SUPER privilege and binary logging is
>  enabled (you might want to use the less safe log_bin_trust_function_creators
>  variable)
> ```

MySQL 互換データベースを同種データ移行のターゲットとして使用する場合の制限

MySQL 互換データベースを同種データ移行のターゲットとして使用する場合、次の制限が適用されます。

- データソースへの接続に使用するユーザー名には以下の制限があります。
  - 2～64 文字を使用できます。
  - スペースは使用できません。
  - 文字として a～z、A～Z、0～9、アンダースコア (_) を使用できます。
  - ハイフン (-) を含めることはできません。
  - a～z または A～Z で始める必要があります。
- データソースへの接続に使用するパスワードには以下の制限があります。

- 1 ~ 128 文字を使用できます。
- 一重引用符 (')、二重引用符 (")、セミコロン (;)、スペースのいずれも使用できません。

での同種データ移行のターゲットとしての PostgreSQL データベースの使用 AWS DMS

AWS DMSでの同種データ移行の移行ターゲットとして、PostgreSQL 互換データベースを使用できます。

AWS DMS では、ターゲット Amazon RDS for PostgreSQL または Amazon Aurora PostgreSQL データベースにデータを移行するための特定のアクセス許可が必要です。次のスクリプトを使用して、ターゲットの PostgreSQL データベースで必要な権限を持つデータベースユーザーを作成します。

```
CREATE USER your_user WITH LOGIN PASSWORD 'your_password';
GRANT USAGE ON SCHEMA schema_name TO your_user;
GRANT CONNECT ON DATABASE db_name to your_user;
GRANT CREATE ON DATABASE db_name TO your_user;
GRANT CREATE ON SCHEMA schema_name TO your_user;
GRANT UPDATE, INSERT, SELECT, DELETE, TRUNCATE ON ALL TABLES IN SCHEMA schema_name
 TO your_user;
            #For "Full load and change data capture (CDC)" and "Change data capture
 (CDC)" data migrations, setting up logical replication requires rds_superuser
 privileges
GRANT rds_superuser TO your_user;
```

上記の例の各 *user input placeholder* は独自の情報に置き換えます。

RDS for PostgreSQL のターゲットの論理レプリケーションを有効にするには、DB パラメータグループの rds.logical_replication パラメータを 1 に設定します。この静的パラメータを有効にするには、DB インスタンスまたは DB クラスターを再起動する必要があります。一部のパラメータは静的であり、サーバーを再起動するまではサーバースタート時にのみ設定できます。 は DB パラメータグループのエントリへの変更 AWS DMS を無視します。

PostgreSQL はトリガーを使用して外部キーの制約を実装します。全ロードフェーズでは、 は各テーブルを一度に 1 つずつ AWS DMS ロードします。フルロード中は、ターゲットデータベースの外部キー制約をオフにすることをお勧めします。これには、次のいずれかの方法を使用します。

- インスタンスのすべてのトリガーを一時的にオフにして、フルロードを終了する。

- PostgreSQL で session_replication_role パラメータの値を変更する。

  特定の時間において、トリガーは origin、replica、always、または disabled のいずれかの状態になります。session_replication_role パラメータを replica と設定すると、replica の状態のトリガーのみがアクティブになります。それ以外の場合、トリガーは非アクティブなままです。

MySQL 互換データベースを同種データ移行のターゲットとして使用する場合の制限

PostgreSQL 互換データベースを同種データ移行のターゲットとして使用する場合、次の制限が適用されます。

- データソースへの接続に使用するユーザー名には以下の制限があります。
  - 2 ~ 64 文字を使用できます。
  - スペースは使用できません。
  - 文字として a ~ z、A ~ Z、0 ~ 9、アンダースコア (_) を使用できます。
  - a ~ z または A ~ Z で始める必要があります。
- データソースへの接続に使用するパスワードには以下の制限があります。
  - 1 ~ 128 文字を使用できます。
  - 一重引用符 (')、二重引用符 (")、セミコロン (;)、スペースのいずれも使用できません。

での同種データ移行のターゲットとしての Amazon DocumentDB データベースの使用 AWS DMS

での同種データ移行の移行ターゲットとして、Amazon DocumentDB (MongoDB 互換) データベースと DocumentDB Elastic クラスターを使用できます AWS DMS。

Amazon DocumentDB ターゲットに対して同種データ移行を実行するには、管理者権限を持つユーザーアカウントを作成するか、移行するデータベースに対してのみ読み取り/書き込み権限を持つユーザーを作成できます。

同種データ移行は、Amazon DocumentDB がサポートするすべての BSON データ型をサポートします。これらのデータ型のリストについては、Amazon DocumentDB デベロッパーガイド」の「データ型」を参照してください。

DocumentDB Elastic クラスターのシャード機能を使用してソースから非シャードコレクションを移行するには、データ移行タスクを開始する前に、シャードコレクションを作成して移行

します。Amazon DocumentDB Elastic クラスターのシャードコレクションの詳細については、「Amazon DocumentDB デベロッパーガイド Amazon DocumentDB」の「ステップ 5: コレクションをシャードする」を参照してください。

Amazon DocumentDB ターゲットの場合、 は noneまたは `require` SSL モード AWS DMS をサポートします。

での同種データ移行の実行 AWS DMS

同種データ移行 の AWS DMS を使用して、ソースデータベースから Amazon Relational Database Service (Amazon RDS)、Amazon Aurora、または Amazon DocumentDB 上の同等のエンジンにデータを移行できます。 は、ソースデータベースとターゲットデータベースのネイティブデータベースツールを使用して、データ移行プロセス AWS DMS を自動化します。

同種データ移行のためのインスタンスプロファイルとスキーマ変換用の互換性のあるデータプロバイダーを作成した後、移行プロジェクトを作成します。詳細については、「 移行プロジェクトの作成」を参照してください。

次のセクションでは、同種データ移行を作成、設定、実行する方法を説明します。

トピック

- でのデータ移行の作成 AWS DMS
- 同種データ移行の選択ルール
- でのデータ移行の管理 AWS DMS
- でのデータ移行のモニタリング AWS DMS
- での同種データ移行のステータス AWS DMS
- での同種データ移行による MySQL データベースからのデータの移行 AWS DMS
- での同種データ移行による PostgreSQL データベースからのデータの移行 AWS DMS
- での同種データ移行による MongoDB データベースからのデータの移行 AWS DMS

でのデータ移行の作成 AWS DMS

同じタイプの互換性のあるデータプロバイダーを使用して移行プロジェクトを作成すると、このプロジェクトを同種データ移行に使用できます。詳細については、「 移行プロジェクトの作成」を参照してください。

同種データ移行の使用を開始するには、新しいデータ移行を作成します。単一の移行プロジェクトで、異なるタイプの複数の同種データ移行を作成できます。

AWS DMS には、 用に作成できる同種データ移行の最大数があります AWS アカウント。Service AWS DMS Quotas の詳細については、次のセクションを参照してくださいAWS Database Migration Service のクォータ。

データ移行を作成する前に、ソースデータベースとターゲットデータベース、IAM ポリシーとロール、インスタンスプロファイル、データプロバイダーなどの必要なリソースを設定していることを確認します。詳細については、IAM リソースの作成、インスタンスプロファイルの作成、および データプロバイダーの作成を参照してください。

上位のデータベースバージョンから下位のデータベースバージョンにデータを移行する場合は、同種データ移行の使用はお勧めしません。ソースデータプロバイダーとターゲットデータプロバイダーに使用しているデータベースのバージョンを確認して、必要に応じてターゲットデータベースのバージョンをアップグレードします。

データ移行を作成するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. 移行プロジェクトを選択して、[データの移行] タブで、[データ移行の作成] をクリックします。
4. [名前] には、データ移行名を入力します。識別しやすいように、データ移行には必ず一意の名前を使用します。
5. [レプリケーションタイプ] では、設定するデータ移行のタイプを選択します。次のオプションのいずれかを選択できます。

   - フルロード — 既存のソースデータを移行する。
   - フルロードと変更データキャプチャ (CDC) — 既存のソースデータを移行し、継続的に変更をレプリケートする。
   - 変更データキャプチャ (CDC) — 継続的に変更をレプリケートする。
6. CloudWatch 「ログを有効にする」のチェックボックスをオンにして、データ移行ログを Amazon に保存します CloudWatch。このオプションを選択しないと、データ移行が失敗してもログファイルを参照できません。
7. (オプション) [詳細設定] を展開します。ジョブの数 には、ソースデータをターゲットに移行するために AWS DMS が使用できる並列スレッドの数を入力します。

8. [IAM サービスロール] では、前提条件のステップで作成した IAM ロールを選択します。詳細については、「AWS DMS での同種データ移行のための IAM ロールの作成」を参照してください。
9. [変更データキャプチャ (CDC)] タイプのデータ移行の [開始モード] を設定します。次のオプションのいずれかを選択できます。

    - すぐに — 継続的なレプリケーションをデータ移行の開始時に開始する。
    - [ネイティブ開始点を使用する] — 継続的なレプリケーションを指定された時点から開始する。

      PostgreSQL データベースの場合、[スロット名] に論理レプリケーション名を入力して、[ネイティブ開始点] にトランザクションログのシーケンス番号を入力します。

      MySQL データベースの場合は、[ログシーケンス番号 (LSN)] にトランザクションログのシーケンス番号を入力します。
10. [変更データキャプチャ (CDC)] タイプまたは [フルロードと変更データキャプチャ (CDC)] タイプのデータ移行の [停止モード] を設定します。次のオプションのいずれかを選択できます。

    - CDC を停止しない — データ移行を停止するまで、進行中のレプリケーション AWS DMS を続行します。
    - サーバー時間ポイントの使用 — 指定された時間に進行中のレプリケーションを AWS DMS 停止します。

      このオプションを選択した場合、継続的なレプリケーションを自動的に停止する日時を [停止日時] に入力します。
11. [データ移行の作成] をクリックします。

AWS DMS はデータ移行を作成し、移行プロジェクトのデータ移行タブのリストに追加します。このリストでデータ移行のステータスを確認できます。詳細については、「移行ステータス」を参照してください。

> **Important**
>
> 全ロードおよび全ロードおよび変更データキャプチャ (CDC) タイプのデータ移行の場合、はターゲットデータベース上のすべてのデータ、テーブル、およびその他のデータベースオブジェクト AWS DMS を削除します。ターゲットデータベースのバックアップがあることを確認します。

AWS DMS がデータ移行を作成すると、このデータ移行のステータスは Ready に設定されます。データを移行するには、データ移行を手動で開始する必要があります。開始するには、リストからデータ移行を選択します。次に、[アクション] で [開始] を選択します。詳細については、「データ移行の管理」を参照してください。

同種データ移行の最初の起動には、何らかの setup. AWS DMS creates a serverless environment for your data migration が必要です。このプロセスは最長 15 分かかる。データ移行を停止して再起動した後は、環境を再度作成 AWS DMS せず、データ移行にすばやくアクセスできます。

同種データ移行の選択ルール

選択ルールを使用して、レプリケーションに含めるスキーマ、テーブル、またはその両方を選択できます。

> **Note**
>
> AWS DMS は、MongoDB 互換データベースをソースとして使用する場合にのみ、同種データ移行の選択ルールをサポートします。

データ移行タスクを作成するときは、選択ルールの追加 を選択します。

ルール設定には、次の値を指定します。

- スキーマ : スキーマ を入力します を選択します。
- スキーマ名 : レプリケートするスキーマの名前を指定するか、ワイルドカード%として使用します。
- テーブル名 : : レプリケートするテーブルの名前を指定するか、ワイルドカード%として使用します。

デフォルトでは、DMS がサポートするルールアクションは のみでInclude、DMS がサポートするワイルドカード文字は のみです%。

Example スキーマ内のすべてのテーブルの移行

以下の例では、ソース内の dmsst という名前のスキーマからすべてのテーブルをターゲットエンドポイントに移行します。

```
{
    "rules": [
```

```
        {
            "rule-type": "selection",
            "rule-action": "include",
            "object-locator": {
                "schema-name": "dmsst",
                "table-name": "%"
            },
            "filters": [],
            "rule-id": "1",
            "rule-name": "1"
        }
    ]
}
```

Example スキーマの一部のテーブルの移行

次の例では、 で始まる名前のすべてのテーブルをcollectionTest、ソースdmsstの という名前のスキーマからターゲットエンドポイントに移行します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-action": "include",
            "object-locator": {
                "schema-name": "dmsst",
                "table-name": "collectionTest%"
            },
            "filters": [],
            "rule-id": "1",
            "rule-name": "1"
        }
    ]
}
```

Example 複数のスキーマから特定のテーブルを移行する

次の例では、ソース内の dmsstおよび という名前の複数のスキーマからターゲットエンドポイントにテーブルの一部Testを移行します。

```
{
    "rules": [
        {
```

```
            "rule-type": "selection",
            "rule-action": "include",
            "object-locator": {
                "schema-name": "dmsst",
                "table-name": "collectionTest1"
            },
            "filters": [],
            "rule-id": "1",
            "rule-name": "1"
        },
        {
            "rule-type": "selection",
            "rule-action": "include",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "products"
            },
            "filters": [],
            "rule-id": "2",
            "rule-name": "2"
        }
    ]
}
```

でのデータ移行の管理 AWS DMS

データ移行を作成した後、 AWS DMS データ移行を自動的に開始しません。データ移行は、必要に応じて手動で開始します。

データ移行を開始する前には、データ移行のすべての設定を変更できます。データ移行を開始した後でレプリケーションタイプを変更することはできません。別のレプリケーションタイプを使用するには、新しいデータ移行を作成します。

データ移行の開始

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. 移行プロジェクトを選択します。[データの移行] タブで作業するデータ移行を選択します。データ移行の [概要] ページが開きます。
4. [Actions] (アクション) で [Start] (開始) を選択します。

その後、 はデータ移行用のサーバーレス環境 AWS DMS を作成します。このプロセスは最長 15 分かかる。

データ移行を開始すると、 AWS DMS はステータスを開始 に設定します。がデータ移行 AWS DMS に使用する次のステータスは、データ移行設定で選択したレプリケーションのタイプによって異なります。詳細については、「移行ステータス」を参照してください。

データ移行を変更するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. 移行プロジェクトを選択します。[データの移行] タブで作業するデータ移行を選択します。データ移行の [概要] ページが開きます。
4. [変更] を選択します。
5. データ移行の設定を行います。

> **Important**
>
> データ移行を開始している場合、レプリケーションタイプは変更できません。

6. Amazon でデータ移行ログを表示するには CloudWatch、ログを有効にする のチェックボックスをオンにします CloudWatch 。
7. [変更の保存] を選択します。

がデータ移行 AWS DMS を開始したら、停止できます。停止するには、[データの移行] タブでデータ移行を選択します。次に、[アクション] で [停止] を選択します。

データ移行を停止すると、 はステータスを Stopping AWS DMS に設定します。次に、 AWS DMS はデータ移行のステータスを [停止済み] に変更します。がデータ移行 AWS DMS を停止すると、データ移行を変更、再開、再起動、または削除できます。

データレプリケーションを続行するには、[データの移行] タブで停止したデータ移行を選択します。次に、[アクション] で [処理を再開] を選択します。

データのロードを再開するには、[データの移行] タブで停止したデータ移行を選択します。次に、アクション で、「再起動」を選択します。ターゲットデータベースからすべてのデータ AWS DMS を削除し、ゼロからデータ移行を開始します。

停止したり開始したりしていないデータ移行は削除できます。データ移行を削除するには、[データの移行] タブで削除するデータ移行を選択します。次に、[アクション] で [削除] を選択します。移行プロジェクトを削除するには、すべてのデータ移行を停止して削除します。

でのデータ移行のモニタリング AWS DMS

同種データ移行を開始した後、移行のステータスと進行状況をモニタリングできます。数百ギガバイトなどの大規模なデータセットのデータ移行の場合、完了するまでに数時間かかります。データ移行の信頼性、可用性、パフォーマンスを維持するには、移行の進捗を定期的にモニタリングします。

データ移行のステータスと進捗を確認するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. 移行プロジェクトを選択して、[データの移行] タブに移動します。
4. データ移行について、[ステータス] 列を確認します。この列の値の詳細については、「移行ステータス」を参照してください。
5. 実行中のデータ移行の場合、[移行の進行状況] 列には移行済みのデータの割合が % で表示されます。

データ移行の詳細を確認するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. 移行プロジェクトを選択します。[データの移行] タブで作業するデータ移行を選択します。
4. [詳細]タブ移行の進捗を確認できます。特に、次のメトリクスに注目します。

   - パブリック IP アドレス – データ移行のパブリック IP アドレス。この値は、ネットワーク設定に必要となる。詳細については、「ネットワークのセットアップ」を参照してください。
   - ロード済みテーブル数 – ロードが正常に完了したテーブルの数

- ロード中テーブル数 – 現在ロード中のテーブルの数
- キューされたテーブル数 – 現在ロード待機中のテーブルの数
- エラーのあるテーブル数 – ロードに失敗したテーブルの数
- 経過時間 – データ移行の開始から経過した時間
- CDC レイテンシー — ソーステーブルで変更が発生してから、 がこの変更をターゲットテーブル AWS DMS に適用するまでにかかる平均時間。
- 移行を開始しました – このデータ移行を開始した時刻
- 移行を停止しました – このデータ移行を停止した時刻

5. データ移行のログファイルを表示するには、同種データ移行設定 で CloudWatch ログの表示 を選択します。データ移行を作成または変更するときに、 CloudWatch ログを有効にできます。詳細については、「データ移行の作成」および「データ移行の管理」を参照してください。

Amazon CloudWatch アラームまたはイベントを使用して、データ移行を詳細に追跡できます。詳細については、「Amazon ユーザーガイド」の「Amazon CloudWatch、Amazon CloudWatch Events、Amazon CloudWatch Logs とは」を参照してください。 CloudWatch Amazon の使用には料金がかかることに注意してください CloudWatch。

同種データ移行の場合、Amazon には以下のメトリクス AWS DMS が含まれます CloudWatch。

メトリクス	説明
OverallCDCLatency	CDC フェーズ中の全体的なレイテンシー。 MySQL データベースの場合、このメトリクスは、ソースバイナリログの変更とその変更のレプリケーション間の経過秒数を表示します。 PostgreSQL データベースの場合、このメトリクスは `pg_stat_subscription` ビューからの `last_msg_receipt_time` と `last_msg_send_time` 間の経過秒数を表示します。 単位: 秒
StorageConsumption	データ移行が使用するストレージ容量。 単位: バイト

での同種データ移行のステータス AWS DMS

実行するデータ移行ごとに、 は ステータスを AWS DMS コンソールに AWS DMS 表示します。提供されるステータスは次のリストのとおりです。

- `Creating` – AWS DMS はデータ移行を作成しています。
- `Ready` – データ移行を開始する準備が整っている。
- `Starting` – AWS DMS は、データ移行用のサーバーレス環境を作成しています。このプロセスは最長 15 分かかる。
- `Load running` – AWS DMS は全ロード移行を実行しています。
- `Load complete, replication ongoing` - 全ロード AWS DMS を完了し、進行中の変更をレプリケートするようになりました。 は、全ロードおよび変更データキャプチャ (CDC) タイプのデータ移行にのみこのステータス AWS DMS を使用します。
- `Replication ongoing` – AWS DMS は進行中の変更をレプリケートしています。 は、変更データキャプチャ (CDC) タイプの移行にのみこのステータス AWS DMS を使用します。
- `Reloading target` – AWS DMS はデータ移行を再開し、指定された移行タイプを実行します。
- `Stopping` – AWS DMS は、アクションメニューでデータ移行を停止することを選択 AWS DMS した後に、データ移行を停止します。
- `Stopped` — AWS DMS はデータ移行を停止しました。
- `Failed` – データ移行が失敗した。詳細については、ログファイルを確認する。

  ログファイルを確認するには、[データの移行] タブでデータ移行を選択します。次に、同種データ移行設定 で CloudWatch ログの表示 を選択します。

  > **Important**
  >
  > データ移行の作成 CloudWatch 時にログを有効にするのチェックボックスをオンにすると、ログファイルを表示できます。

- `Deleting` - AWS DMS は、アクションメニューでデータ移行を削除することを選択した後、データ移行を削除します。 はこのステータス AWS DMS を設定します。

での同種データ移行による MySQL データベースからのデータの移行 AWS DMS

同種データ移行 を使用して、セルフマネージド型 MySQL データベースを RDS for MySQL または Aurora MySQL に移行できます。 AWS DMS は、データ移行のためのサーバーレス環境を作成します。さまざまなタイプのデータ移行では、 はさまざまなネイティブ MySQL データベースツール AWS DMS を使用します。

フルロードタイプの同種データ移行の場合、 は mydumper AWS DMS を使用してソースデータベースからデータを読み取ってサーバーレス環境にアタッチされたディスクに保存します。がすべてのソースデータを AWS DMS 読み取りた後、ターゲットデータベースの myloader を使用してデータを復元します。

フルロードおよび変更データキャプチャ (CDC) タイプの同種データ移行の場合、 は mydumper AWS DMS を使用してソースデータベースからデータを読み取ってサーバーレス環境にアタッチされたディスクに保存します。がすべてのソースデータを AWS DMS 読み取りた後、ターゲットデータベースの myloader を使用してデータを復元します。が全ロード AWS DMS を完了すると、バイナリログ位置を全ロードの開始に設定してバイナリログレプリケーションを設定します。データの不整合を避けるには、[ジョブの数] を 1 に設定して、既存のデータの一貫性ある状態をキャプチャします。詳細については、「データ移行の作成」を参照してください。

[変更データキャプチャ (CDC)] タイプの同種データ移行の場合、 AWS DMS がレプリケーションを開始するには [ネイティブな CDC 開始点] が必要です。ネイティブ CDC 開始点を指定すると、 はその時点からの変更を AWS DMS キャプチャします。または、データ移行設定で [すぐに] を選択すると、実際のデータ移行の開始時にレプリケーションの開始点を自動的にキャプチャできます。

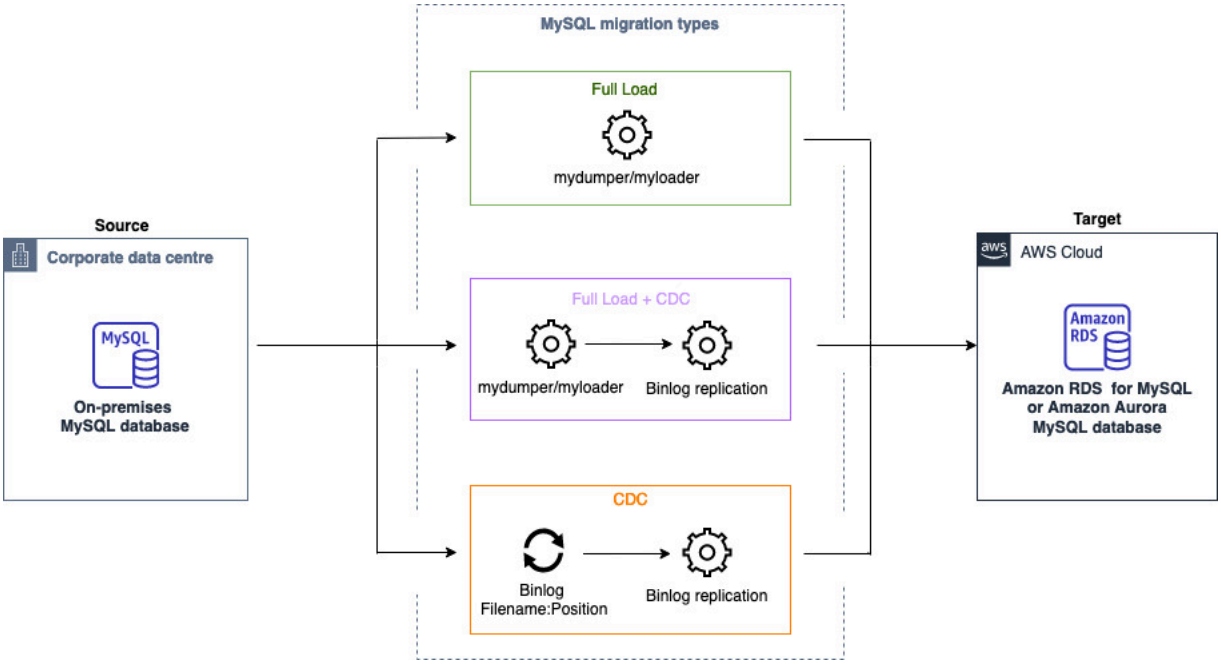
> **Note**
>
> CDC のみの移行を正常に実行するには、ソースデータベースのすべてのスキーマとオブジェクトがターゲットデータベースに既に存在している必要があります。ただし、ターゲットにはソースには存在しないオブジェクトが含まれている場合があります。

次のコード例を使用すると、MySQL データベース内の現在のログシーケンス番号 (LSN) を取得できます。

```
show master status
```

このクエリは、binlog ファイル名と位置を返します。ネイティブ開始点には、binlog ファイル名と位置の組み合わせを使用します。例えば mysql-bin-changelog.000024:373 です。この例では、mysql-bin-changelog.000024はバイナリログファイル名で、 373は が変更のキャプチャ AWS DMS を開始する位置です。

次の図は、 で同種データ移行を使用して MySQL データベース AWS DMS を RDS for MySQL または Aurora MySQL に移行するプロセスを示しています MySQL 。



での同種データ移行による PostgreSQL データベースからのデータの移行 AWS DMS

同種データ移行 を使用して、セルフマネージド型 PostgreSQL データベースを RDS for PostgreSQL または Aurora PostgreSQL に移行できます。 AWS DMS は、データ移行用のためのサーバーレス環境を作成します。 AWS DMS はさまざまなネイティブ PostgreSQL データベースツールを使用して、さまざまなタイプのデータ移行に対応します。

フルロードタイプの同種データ移行の場合、pg_dump AWS DMS を使用してソースデータベースからデータを読み取ってサーバーレス環境にアタッチされたディスクに保存します。がすべてのソースデータを AWS DMS 読み取りた後、ターゲットデータベースの pg_restore を使用してデータを復元します。

フルロードおよび変更データキャプチャ (CDC) タイプの同種データ移行の場合、 AWS DMS を使用してソースデータベースからテーブルデータなしでスキーマオブジェクトをpg_dump読み取り、サーバーレス環境にアタッチされたディスクに保存します。次に、ターゲットデータベー

スpg_restoreの を使用してスキーマオブジェクトを復元します。がpg_restoreプロセス AWS DMS を完了すると、論理レプリケーション用のパブリッシャーモデルとサブスクライバーモデルに自動的に切り替わり、ソースデータベースからターゲットデータベースInitial Data Synchronizationに初期テーブルデータを直接コピーし、継続的なレプリケーションを開始します。このモデルでは、単一または複数のサブスクライバーがパブリッシャノード上の単独または複数のパブリケーションにサブスクライブします。

変更データキャプチャ (CDC) タイプの同種データ移行の場合、レプリケーションを開始するにはネイティブ開始点 AWS DMS が必要です。ネイティブ開始点を指定すると、 はその時点からの変更を AWS DMS キャプチャします。または、データ移行設定で [すぐに] を選択すると、実際のデータ移行の開始時にレプリケーションの開始点を自動的にキャプチャできます。

> **Note**
>
> CDC のみの移行を正常に実行するには、ソースデータベースのすべてのスキーマとオブジェクトがターゲットデータベースに既に存在している必要があります。ただし、ターゲットにはソースには存在しないオブジェクトが含まれている場合があります。

次のコード例を使用すると、PostgreSQL データベースのネイティブ開始点を取得できます。

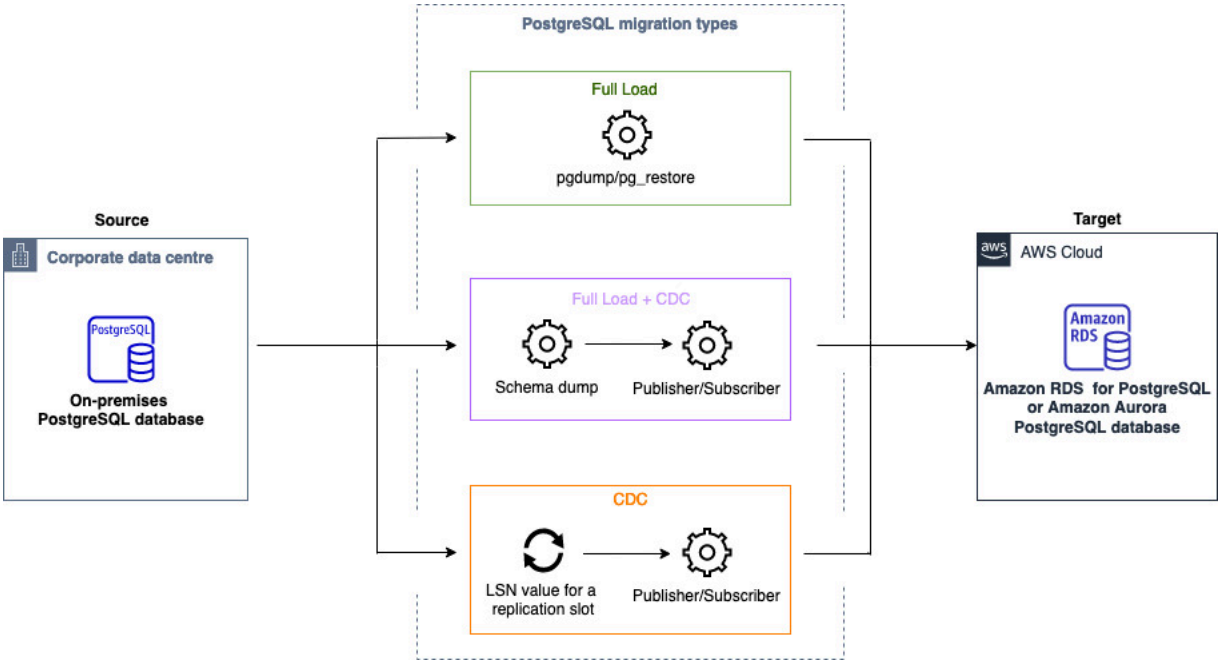
```
select confirmed_flush_lsn from pg_replication_slots where
 slot_name='migrate_to_target';
```

このクエリは、PostgreSQL データベースの pg_replication_slots ビューを使用してログシーケンス番号 (LSN) の値を取得します。

AWS DMS が PostgreSQL 同種データ移行のステータスを Stopped 、Failed 、または Deleted に設定した後、パブリッシャーとレプリケーションは削除されません。移行を再開しない場合は、次のコマンドを使用してレプリケーションスロットとパブリッシャーを削除します。

```
SELECT pg_drop_replication_slot('migration_subscriber_{ARN}');
            DROP PUBLICATION publication_{ARN};
```

次の図は、 で同種データ移行を使用して PostgreSQL データベースを RDS for PostgreSQL または Aurora PostgreSQL AWS DMS に移行するプロセスを示しています。



での同種データ移行による MongoDB データベースからのデータの移行 AWS DMS

を使用して同種データ移行、セルフマネージド MongoDB データベースを Amazon DocumentDB に移行できます。 AWS DMS は、データ移行用のサーバーレス環境を作成します。さまざまなタイプのデータ移行に対して、 はさまざまなネイティブ MongoDB データベースツール AWS DMS を使用します。

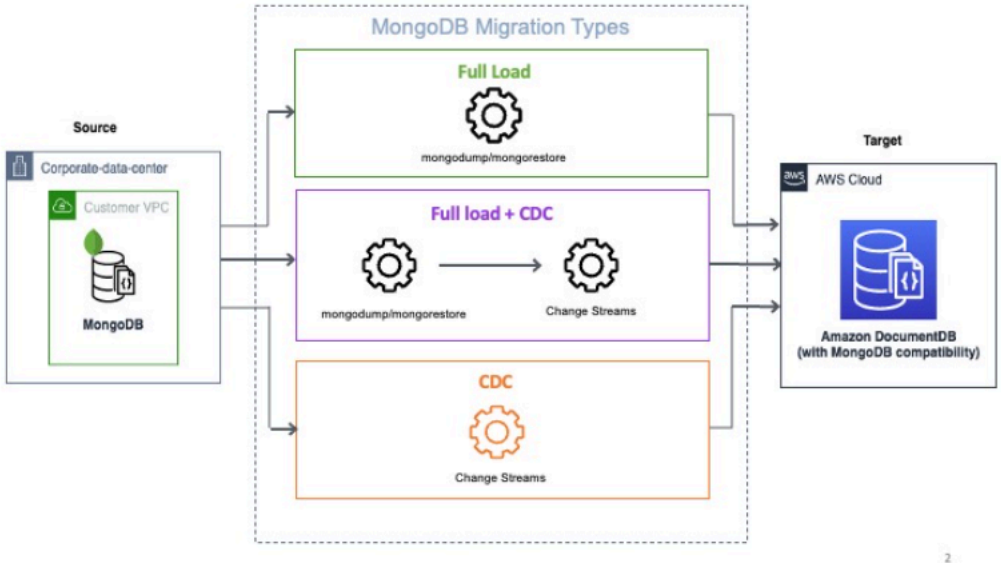
フルロードタイプの同種データ移行の場合、 AWS DMS を使用してソースデータベースからデータをmongodump読み取り、サーバーレス環境にアタッチされたディスクに保存します。は、すべてのソースデータを AWS DMS 読み取り、ターゲットデータベースmongorestoreで を使用してデータを復元します。

フルロードおよび変更データキャプチャ (CDC) タイプの同種データ移行の場合、 AWS DMS を使用してソースデータベースからデータmongodumpを読み取ってサーバーレス環境にアタッチされたディスクに保存します。は、すべてのソースデータを AWS DMS 読み取り、ターゲットデータベースmongorestoreで を使用してデータを復元します。が全ロード AWS DMS を完了すると、論理レプリケーションのためにパブリッシャーモデルとサブスクライバーモデルに自動的に切り替わります。このモデルでは、変更を少なくとも 24 時間保持するように oplog のサイズを設定することをお勧めします。

変更データキャプチャ (CDC) タイプの同種データ移行の場合、データ移行設定immediatelyで を選択すると、実際のデータ移行の開始時にレプリケーションの開始点が自動的にキャプチャされます。

> **Note**
>
> 新規または名前を変更したコレクションについては、同種データ移行として、それらのコレクションの新しいデータ移行タスクを作成する必要があります。MongoDB 互換のソース AWS DMS の場合、 `createrename`および `drop collection`オペレーションはサポートされていません。

次の図は、 で同種データ移行を使用して MongoDB データベースを Amazon DocumentDB AWS DMS に移行するプロセスを示しています。



AWS DMS での同種データ移行のトラブルシューティング

次のリストは、AWS DMS での同種データ移行で問題が発生した場合に実行すべきアクションを説明しています。

トピック

- AWS DMS で同種データ移行を作成できない
- AWS DMS で同種データ移行を開始できない
- AWS DMS でのデータの移行の実行時にターゲットデータベースに接続できない

- PostgreSQL で AWS DMS がビューをテーブルとして保存する

AWS DMS で同種データ移行を作成できない

[データ移行の作成] をクリックした後、AWS DMS はデータプロバイダーに接続できないというエラーメッセージが表示された場合、必要となる IAM ロールが設定されているかを確認します。詳細については、「IAM ロールの作成」を参照してください。

IAM ロールを設定してもこのようなエラーメッセージが表示される場合は、AWS KMS キー設定でこの IAM ロールをキーユーザーとして追加します。詳細については、「AWS Key Management Service デベロッパーガイド」の「KMS キーの使用をキーユーザーに許可する」を参照してください。

AWS DMS で同種データ移行を開始できない

移行プロジェクトでデータ移行を開始した際に、ステータスが `Failed` と表示された場合は、ソースデータプロバイダーとターゲットデータプロバイダーのバージョンを確認します。確認するには、MySQL または PostgreSQL データベースで `SELECT VERSION();` クエリを実行します。サポート対象のデータベースのバージョンを使用しているかを確認します。

サポート対象のソースデータベースの一覧については、「DMS 同種データ移行のソース」を参照してください。

サポート対象のターゲットデータベースの一覧については、「DMS 同種データ移行のターゲット」を参照してください。

サポートされていないデータベースのバージョンを使用している場合は、ソースデータベースまたはターゲットデータベースをアップグレードして、もう一度試してください。

AWS DMS コンソールでデータ移行のエラーメッセージを確認します。確認するには、移行プロジェクトを開いて、データ移行を選択します。[詳細] タブの [一般] の下にある [最後のエラーメッセージ] を確認します。

最後に、CloudWatch ログを分析します。確認するには、移行プロジェクトを開いて、データ移行を選択します。分析するには、[詳細] タブで、[CloudWatch ログを表示] をクリックします。

AWS DMS でのデータの移行の実行時にターゲットデータベースに接続できない

「ターゲットに接続できません」というエラーメッセージが表示されたら、次のアクションを実行します。

1. ソースデータベースとターゲットデータベースにアタッチされているセキュリティグループに、インバウンドトラフィックとアウトバウンドトラフィックのルールがあるかを確認します。詳細については、「継続的なレプリケーションの設定」を参照してください。
2. ネットワークアクセスコントロールリスト (ACL) とルートテーブルのルールを確認します。
3. データベースは、作成した VPC からアクセスできる必要があります。VPC セキュリティグループにパブリック IP アドレスを追加して、ファイアウォールでの受信接続を許可します。
4. 移行プロジェクトの [データの移行] タブで、作業するデータ移行を選択します。[詳細] タブの [接続とセキュリティ] の下に表示されている パブリック IP アドレス をメモしておきます。次に、ソースデータベースとターゲットデータベースで、データ移行のパブリック IP アドレスからのアクセスを許可します。
5. 継続的なデータレプリケーションでは、ソースデータベースとターゲットデータベース間の相互通信が有効であるかを確認します。

詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「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。

PostgreSQL で AWS DMS がビューをテーブルとして保存する

同種データ移行では、PostgreSQL のビューをビューとして移行することはサポート対象外です。PostgreSQL の場合、AWS DMS はビューをテーブルとして移行します。

でのデータプロバイダー、インスタンスプロファイル、移行プロジェクトの使用 AWS DMS

で DMS Schema Conversion と同種データ移行を使用する場合は AWS Database Migration Service、移行プロジェクトを使用します。 AWS DMS 移行プロジェクトでは、サブネットグループ、インスタンスプロファイル、データプロバイダーを使用します。

サブネットは、VPC の IP アドレスの範囲です。レプリケーションサブネットグループには、インスタンスプロファイルで使用できるさまざまなアベイラビリティーゾーンのサブネットが含まれます。レプリケーションサブネットグループは DMS リソースであり、Amazon VPC および Amazon RDS が使用するサブネットグループとは異なることに注意してください。

インスタンスプロファイルは、移行プロジェクトを実行するサーバーレス環境のネットワークとセキュリティの設定を指定します。

データプロバイダーは、データストアのタイプとデータベースの場所の情報を保存します。移行プロジェクトにデータプロバイダーを追加すると、 からデータベース認証情報を指定します AWS Secrets Manager。 はこの情報 AWS DMS を使用してデータベースに接続します。

データプロバイダー、インスタンスプロファイル、その他の AWS リソースを作成したら、移行プロジェクトを作成できます。移行プロジェクトでは、インスタンスプロファイル、ソースとターゲットのデータプロバイダー、および からのシークレットについて説明します AWS Secrets Manager。さまざまなソースデータプロバイダーとターゲットデータプロバイダーに対して複数の移行プロジェクトを作成できます。

作業のほとんどは移行プロジェクトで行います。DMS Schema Conversion の場合、移行プロジェクトを使用してソースデータプロバイダーのオブジェクトを評価し、ターゲットデータベースと互換性のある形式に変換します。その後、変換したコードをターゲットデータプロバイダーに適用したり、SQL スクリプトとして保存したりすることができます。同種データ移行の場合は、移行プロジェクトを使用して、ソースデータベースから AWS クラウド内の同じタイプのターゲットデータベースにデータを移行します。

の移行プロジェクト AWS DMS はサーバーレスのみです。 は、移行プロジェクトのクラウドリソース AWS DMS を自動的にプロビジョニングします。

AWS DMS には、 用に作成できるインスタンスプロファイル、データプロバイダー、移行プロジェクトの最大数があります AWS アカウント。 AWS DMS サービスのクォータについては、以降のセクション「AWS Database Migration Service のクォータ」を参照してください。

トピック

AWS DMS 移行プロジェクトのサブネットグループの作成

インスタンスプロファイルを作成する前に、インスタンスプロファイルのサブネットグループを設定します。

サブネットグループを作成するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. ナビゲーションペインで、[サブネットグループ] を選択して、[サブネットグループの作成] をクリックします。
3. [名前] には、サブネットグループの一意の名前を入力します。
4. [説明] には、このサブネットグループの簡単な説明を入力します。
5. [VPC] では、2 つ以上のアベイラビリティーゾーンに少なくとも 1 つのサブネットがある VPC を選択します。
6. [サブネットの追加] では、サブネットグループに含めるサブネットを選択します。少なくとも 2 つのアベイラビリティーゾーンにあるサブネットを選択する必要があります。

   Amazon RDS データベースへの接続のために、サブネットグループにパブリックサブネットを追加します。オンプレミスのデータベースに接続するには、プライベートサブネットをサブネットグループに追加します。
7. [サブネットグループの作成] を選択します。

のインスタンスプロファイルの作成 AWS Database Migration Service

AWS DMS コンソールで複数のインスタンスプロファイルを作成できます。 AWS DMSで作成する各移行プロジェクトに使用するインスタンスプロファイルは必ず選択します。

インスタンスプロファイルを作成するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. ナビゲーションペインで [インスタンスプロファイル] を選択します。
3. [インスタンスプロファイルの作成] をクリックします。
4. [インスタンスプロファイルの作成] ページで、インスタンスプロファイルの [名前] に判別しやすい値を入力します。
5. IPv4 と IPv6 の アドレス指定をサポートするインスタンスプロファイルを作成するには、[ネットワークタイプ] で [デュアルスタックモード] を選択します。IPv4 アドレス指定のみをサポートするインスタンスプロファイルを作成するには、デフォルトのオプションをそのままにします。
6. 次に、選択したネットワークタイプのインスタンスを実行するために、[仮想プライベートクラウド (VPC)] を選択します。次に、インスタンスプロファイルのための [サブネットグループ] と [VPC セキュリティグループ] を選択します。

   Amazon RDS データベースに接続するには、パブリックサブネットを備えたサブネットグループを使用します。オンプレミスのデータベースに接続するには、プライベートサブネットを備えたサブネットグループを使用します。が NAT ゲートウェイのパブリック IP アドレスを使用してソースのオンプレミスデータベース AWS DMS にアクセスできるようにネットワークを設定していることを確認します。詳細については、「Amazon VPC を基盤に VPC を作成する」を参照してください。
7. (オプション) DMS Schema Conversion の移行プロジェクトを作成する場合は、次に [スキーマ変換設定 - オプション] で、移行プロジェクトからの情報を保存する Amazon S3 バケットを選択します。次に、この Amazon S3 バケットへのアクセスを提供する AWS Identity and Access Management (IAM) ロールを選択します。 Amazon S3 詳細については、「Amazon S3 バケットを作成する」を参照してください。
8. [インスタンスプロファイルの作成] をクリックします。

インスタンスプロファイルを作成した後は、変更または削除できます。

インスタンスプロファイルを変更するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [インスタンスプロファイル] を選択します。[インスタンスプロファイル] ページが開きます。
3. インスタンスプロファイルを選択して、[変更] をクリックします。
4. インスタンスプロファイル名を更新して、VPC または Amazon S3 バケット設定を編集します。
5. [変更の保存] を選択します。

インスタンスプロファイルを削除するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [インスタンスプロファイル] を選択します。[インスタンスプロファイル] ページが開きます。
3. インスタンスプロファイルを選択して、[削除] をクリックします。
4. [削除] をクリックして、選択を確定します。

でのデータプロバイダーの作成 AWS Database Migration Service

データプロバイダーを作成し、 AWS DMS 移行プロジェクトで使用できます。データプロバイダーとして、オンプレミスまたは Amazon EC2 インスタンスで実行されているセルフマネージド型エンジンを使用できます。また、Amazon Relational Database Service (Amazon RDS) または Amazon Aurora などのフルマネージドエンジンもデータプロバイダーとして使用できます。

データベースごとに単一のデータプロバイダーを作成できます。単一のデータプロバイダーを複数の移行プロジェクトで使用できます。

移行プロジェクトを作成する前に、少なくとも 2 つのデータプロバイダーが作成済みであることを確認します。どちらかのデータプロバイダーが AWS のサービス上にある必要があります。スキーマの変更やオンプレミスのデータベースへのデータの移行には、 AWS DMS を使用できません。

次の手順は、 AWS DMS コンソールウィザードでデータプロバイダーを作成する方法を示しています。

データプロバイダーを作成するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [データプロバイダー] を選択します。[データプロバイダー] ページが開きます。
3. [データプロバイダーの作成] をクリックします。次の表で設定について説明します。

オプション	アクション
設定	データプロバイダーに関する情報を手動で入力するか、Amazon RDS DB インスタンスを使用するかを選択する。
名前	データプロバイダー名を入力する。データプロバイダーを識別しやすいように、必ず一意の名前を使用する。
エンジンのタイプ	データプロバイダーのデータベースのエンジンタイプを選択する。
[Server name] (サーバー名)	データベースサーバーのドメインネームサービス (DNS) 名または IP アドレスを入力する。同種レプリケーションに使用するデータプロバイダーのサーバー名は、英数文字で始める必要があり、英数字、ハイフン (-)、ピリオド (.)、またはアンダースコア (_) のみを使用できます。
[ポート]	データベースサーバーへの接続に使用するポートを入力する。
サービス ID (SID) またはサービス名	Oracle システム ID (SID) を入力する。Oracle SID を見つけるには、Oracle データベースに対して以下のクエリを発行します。 `SELECT sys_context('userenv','instance_name') AS SID FROM dual;`
[Database name] (データベース名)	このデータプロバイダーのデータベースの名前を入力する。同種レプリケーションに使用するデータプロバイダーのデータベース名は最大 63 文字で、スペースを含めることはできません。

オプション	アクション
[Secure Socket Layer (SSL) mode] (Secure Socket Layer (SSL) モード)	このデータプロバイダーの接続暗号化を有効にする場合は、SSL モードを選択する。選択したモードによっては、証明書とサーバー証明書の情報を指定する必要がある場合がある。詳細については、「での SSL AWS Database Migration Serviceの使用」を参照してください。
[Authentication mode] (認証モード)	MongoDB ソースの場合、エンドポイント接続の認証に が AWS DMS 使用する認証モード。
[Authentication source] (認証ソース)	MongoDB ソースの場合、認証用の認証情報の検証に使用する MongoDB データベースの名前。
[Authentication mechanism] (認証メカニズム)	MongoDB ソースの場合、MongoDB がパスワードの暗号化に使用する認証方法。

4. [データプロバイダーの作成] をクリックします。

データプロバイダーを作成した後、必ず AWS Secrets Managerにデータベース接続認証情報を追加します。

での移行プロジェクトの作成 AWS Database Migration Service

で移行プロジェクトを作成する前に AWS DMS、必ず次のリソースを作成してください。

- ソースデータベースとターゲットデータベースを記述するデータプロバイダー
- に保存されているデータベース認証情報を持つシークレット AWS Secrets Manager
- Secrets Manager へのアクセスを提供する AWS Identity and Access Management (IAM) ロール
- ネットワーク設定とセキュリティ設定済みのインスタンスプロファイル

移行プロジェクトを作成するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. [移行プロジェクトの作成] をクリックします。次の表で設定について説明します。

オプション	アクション
名前	移行プロジェクト名を入力する。移行プロジェクトには、簡単に識別できるように必ず一意の名前を使用する。
インスタンスプロファイル	移行プロジェクトに使用するインスタンスプロファイルを選択する。
ソース	[参照] をクリックして、ソースデータプロバイダーを選択する。
Secret ID (シークレット ID)	ソースデータベースの認証情報を保存する Secrets Manager で、シークレットの Amazon リソースネーム (ARN) を選択する。
IAM ロール	Secrets Manager でソースデータベースの認証情報へのアクセスを提供する IAM ロールを選択する。
[Target] (ターゲット)	[参照] をクリックして、ターゲットデータプロバイダーを選択する。
Secret ID (シークレット ID)	ターゲットデータベースの認証情報を保存する Secrets Manager で、シークレットの ARN を選択する。
IAM ロール	Secrets Manager でターゲットデータベースの認証情報へのアクセスを提供する IAM ロールを選択する。
変換ルール	(オプション) DMS Schema Conversion の移行プロジェクトを作成する場合は、[変換ルールの追加] を選択して変換ルールを設定する。変換ルールを使用すると、指定したルールに沿ってオブジェクト名を変更できる。詳細については、「変換ルールの設定」を参照してください。

4. [移行プロジェクトの作成] をクリックします。

AWS DMS が移行プロジェクトを作成したら、DMS Schema Conversion または同種データ移行でこのプロジェクトを使用できます。移行プロジェクトの作業を開始するには、[移行プロジェクト] ページのリストからプロジェクトを選択します。

での移行プロジェクトの管理 AWS Database Migration Service

移行プロジェクトを作成した後は、変更または削除できます。例えば、ソースデータプロバイダーまたはターゲットデータプロバイダーを変更するには、移行プロジェクトを変更します。

移行プロジェクトを変更または削除できるのは、スキーマ変換の実行またはデータ移行の実行を終了した後のみです。終了するには、リストから移行プロジェクトを選択して、[スキーマ変換] または [データの移行] をクリックします。DMS Schema Conversion の場合は次に、[スキーマ変換を閉じる] をクリックして、選択を確定します。同種データ移行の場合は、データの移行を選択して、[アクション] メニューで [停止] を選択します。移行プロジェクトを編集した後は、スキーマ変換を開始したり、データの移行を再開したりできます。

移行プロジェクトを変更するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. 移行プロジェクトを選択して、[変更] をクリックします。
4. プロジェクト名の更新、インスタンスプロファイルの編集、ソースデータプロバイダーやターゲットデータプロバイダーの変更などを行います。必要に応じて、変換中にオブジェクト名を変更する移行ルールを追加または編集します。
5. [変更の保存] を選択します。

移行プロジェクトを削除するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. [移行プロジェクト] を選択します。[移行プロジェクト] ページが開きます。
3. 移行プロジェクトを選択して、[削除] をクリックします。
4. [削除] をクリックして、選択を確定します。

AWS Database Migration Service のベストプラクティス

AWS Database Migration Service (AWS DMS) を最も効果的に使用するためには、データ移行の最も効率的な方法に関するこのセクションの推奨事項をご参照ください。

トピック

AWS Database Migration Service での移行の計画

AWS Database Migration Service を使用してデータベース移行を計画するときは、以下の点を考慮してください。

- ソースとターゲットのデータベースを AWS DMS レプリケーション インスタンスに接続するにはネットワークを設定してください。これを行うには、レプリケーション インスタンスと同じ Virtual Private Cloud (VPC) にある2 つの AWS リソースを接続するのと同じくらい簡単です。これには、Virtual Private Cloud (VPN) 経由で Amazon RDS DB インスタンスへのオンプレミスデータベースの接続など、より複雑な設定までカバーしています。詳細については、「データベース移行のネットワーク設定」を参照してください。
- ソースとターゲットのエンドポイント ソースデータベース内のどの情報とテーブルをターゲットデータベースに移行する必要があるかを知っている必要があります。AWS DMS では、テーブルとプライマリキーの作成を含む、基本的なスキーマの移行がサポートされています。ただし、AWS DMS では、ターゲットデータベースのセカンダリインデックス、外部キー、ユーザーアカウントなどは自動的に作成されません。ソースおよびターゲットデータベースエンジンによっては、サプリメンタルロギングをセットアップしたり、ソースまたはターゲットデータベースの他の設定を変更したり必要がある点に注意してください。詳細については、「データの移行のソース」と「「データ移行のターゲット」」をご参照ください。
- [Schema and code migration] (スキーマとコードの移行) – AWS DMS はスキーマまたはコードの変換を実行しません。Oracle SQL Developer、MySQL Workbench、または pgAdmin III などのツールを使用してスキーマを変換できます。異なるデータベースエンジンに既存のスキーマを変換する場合、AWS Schema Conversion Tool (AWS SCT) を使用できます。このツールは、ターゲットスキーマを作成でき、スキーマ全体 (テーブル、インデックス、ビューなど) を生成および作成することもできます。このツールを使用して PL/SQL または TSQL を PgSQL や他の形式に変換することもできます。AWS SCT の詳細については、AWS SCT ユーザーガイド をご参照ください。
- [Unsupported data types] (サポートされていないデータ型) – ソースの一部のデータ型は、ターゲットデータベースのデータ型と同等のデータ型に変換する必要があります。サポートされているデータ型の詳細については、データストアのソースまたはターゲットセクションをご参照ください。
- [Diagnostic support script results] (診断サポートスクリプトの結果) — 移行を計画する場合は、診断サポートスクリプトを実行することをお勧めします。これらのスクリプトの結果から、移行の失敗の可能性に関する詳細情報を確認できます。

  サポートスクリプトがデータベースで使用できる場合は、次のセクションにある対応するスクリプトトピックのリンクを使用して、サポートスクリプトをダウンロードします。スクリプトを検証して確認したら、ローカル環境のスクリプトトピックで説明されている手順に従ってスクリプトを実行できます。スクリプトの実行が完了すると、結果を確認できます。トラブルシューティング作業の最初のステップとして、これらのスクリプトを実行することをお勧めします。この結果は、AWS Support チームと協力しているとき有用です。詳細については、「AWS DMS での診断サポート スクリプトの操作」を参照してください。

- [Premigration assessments] (移行前評価) — 移行前評価は、データベース移行タスクの指定されたコンポーネントを評価して、移行タスクが期待どおりに実行されない可能性のある問題を特定するのに役立ちます。この評価を使用すると、新規または変更されたタスクを実行する前に、潜在的な問題を特定できます。移行前の評価での作業の詳細については、「タスクの移行前評価の有効化と操作」をご参照ください。

スキーマの変換

AWS DMS は、スキーマまたはコード変換を実行しません。異なるデータベースエンジンに既存のスキーマを変換する場合、AWS SCT を使用します。AWS SCT は、ソースオブジェクト、テーブル、インデックス、ビュー、トリガー、およびその他のシステムオブジェクトをターゲットデータ定義言語 (DDL) 形式に変換します。AWS SCT を使用して、PL/SQL または TSQL などのほとんどのアプリケーションコードを同等のターゲット言語に変換することもできます。

AWS SCT は、AWS から無料でダウンロードできます。AWS SCT の詳細については、AWS SCT ユーザーガイドをご参照ください。

ソースエンドポイントとターゲットエンドポイントが同じデータベースエンジン上にある場合は、Oracle SQL Developer、MySQL Workbench、4 などのツールを使用してスキーマ PgAdminを移動できます。

AWS DMS 公開ドキュメントのレビュー

最初の移行前のソース エンドポイントとターゲット エンドポイントについて AWS DMS 公開ドキュメントページを確認することを強くお勧めします。このドキュメントは、移行の前提条件を特定し、開始する前に現在の制限事項を理解するのに役立ちます。詳細については、「AWS DMS エンドポイントの使用」を参照してください。

移行中、公開ドキュメントは、AWS DMS に関する問題のトラブルシューティングに役立ちます。これらのトピックは、AWS DMS と選択したエンドポイント データベースの両方を使用して一般的な問題を解決するのに役立ちます。詳細については、「AWS Database Migration Service での移行タスクのトラブルシューティング」を参照してください。

PoC (概念実証) を実行する

データベース移行の初期段階で環境に関する問題を発見するために、小規模なテスト移行を実行することをお勧めします。これにより、より現実的な移行の時間帯を設定するのにも役立ちます。さ

らに、フルスケールのテスト移行を実行して、AWS DMS が、ネットワーク上でデータベースのスループットを処理できるかどうかを測定する必要がある場合があります。この間、最初の全負荷と継続的なレプリケーションをベンチマークして最適化することをお勧めします。これにより、ネットワークのレイテンシーを把握し、全体的なパフォーマンスを測定するのに役立ちます。

この時点で、次のようなデータプロファイルとデータベースの大きさを理解する機会もあります。

- 大、中、小のテーブルがいくつあるか。
- AWS DMS がデータ型と文字セットの変換を処理する方法。
- ラージオブジェクト (LOB) 列を持つテーブルの数。
- テスト移行の実行所要時間。

AWS DMS 移行のパフォーマンスの向上

いくつかの要因が AWS DMS 移行のパフォーマンスに影響を与えます。

- ソースのリソース可用性
- 利用可能なネットワークスループット
- レプリケーション サーバーのリソース容量
- ターゲットの変更取り込み機能
- ソースデータのタイプとディストリビューション
- 移行するオブジェクトの数

パフォーマンスを向上させるには、次のベストプラクティスの一部またはすべてを使用します。これらのプラクティスを使用できるかどうかは、特定のユースケースに大きく左右されます。以下の制限があります。

適切なレプリケーション サーバーのプロビジョニング

AWS DMS は Amazon EC2 インスタンス上で実行されるマネージドサービスです。このサービスはソース データベースに接続するレプリケーション インスタンスを使用し、ソースデータを読み取り、ターゲットデータベースが使用できるようにデータをフォーマットし、ターゲットデータベースにデータをロードします。

この処理のほとんどはメモリ内で行われます。ただし、大きいトランザクションではディスクでのバッファリングが必要になることがあります。キャッシュされたトランザクションとログファ

イルもディスクに書き込まれます。以下のセクションでは、レプリケーション サーバーの選択時に考慮すべき点について説明します。

CPU

AWS DMS は異機種間移行用に設計されていますが、同種の移行もサポートしています。同種移行を実行するには、まず各ソースデータ型を同等の AWS DMS データ型に変換します。次に、それぞれのデータ型をAWS DMS ターゲット データ型に変換します。データベースエンジンごとにこれらの変換の参照は、AWS DMSユーザーガイドをご参照ください。

AWS DMS がこれらの変換を最適に実行するには、変換が発生したときに CPU が使用可能であることが前提です。CPU をオーバーロードし、十分な CPU リソースがないと、移行が遅くなり、他の副作用も引き起こす可能性があります。

レプリケーション インスタンスクラス

サービスのテストや小規模な移行の場合、いくつかの小さいインスタンスクラスで十分です。移行に多数のテーブルが関与する場合や、複数の同時レプリケーション タスクを実行する予定の場合、大きいインスタンスを 1 つ使用することを検討してください。サービスがかなりの量のメモリと CPU を消費するため、より大きなインスタンスを使用することをお勧めします。

T2 タイプのインスタンスは、適度なベースラインパフォーマンスを実現したり、ワークロードの必要に応じて非常に高いパフォーマンスまでバーストする機能を実現できるように設計されています。常時または一貫して CPU をフルに使用するわけではないが、バーストが必要なことがあるワークロード向けに用意されています。T2 インスタンスは、ウェブサーバー、デベロッパー環境、小規模データベースなどの汎用ワークロードに適しています。低速移行のトラブルシューティングで T2 インスタンスタイプを使用する場合は、CPU 使用率ホストメトリックスを確認します。これは、そのインスタンスタイプのベースラインを超えてバーストしているかどうかを確認できます。

この C4 インスタンスクラスは、大量の演算を行うワークロードで最高レベルのプロセッサパフォーマンスを実現するように設計されています。毎秒パケット (PPS) パフォーマンスが非常に高く、ネットワークジッターが低く、ネットワーク レイテンシーが低くなります。また、AWS DMS は特に Oracle から PostgreSQL への移行など、異機種間の移行やレプリケーションを実行する場合に、CPU を大量に消費することもあります。C4 インスタンスは、このような状況に適しています。

R4 インスタンスクラスは、メモリ負荷の大きいワークロード用に最適化されたメモリです。AWS DMS を使用する高スループット トランザクション システムの継続的移行やレプリケーションが原因で、大量の CPU やメモリを消費することがあります。R4 インスタンスは、vCPU 別のメモリが含まれます。

AWS DMS による R5 および C5 インスタンスクラスのサポート

R5 インスタンスは、メモリ内の大きいデータ セットを処理するワークロードに対して高速なパフォーマンスを実現するように設計されています。AWS DMS を使用する高スループット トランザクション システムの継続的移行やレプリケーションが原因で、大量の CPU やメモリを消費することがあります。R5 インスタンスは、R4 より vCPU あたり 5% の追加メモリを提供し、最大サイズは 768 GiB のメモリを提供します。さらに、R5 インスタンスは GiB あたりの価格を 10% 向上させ、R4 に比べて CPU パフォーマンスを約 20% 向上させます。

C5 インスタンスクラスは、計算負荷の高いワークロードに最適化されており、コンピューティング比あたりの低価格で費用対効果の高いパフォーマンスを提供します。これにより、ネットワークパフォーマンスが大幅に向上します。Elastic Network Adapter (ENA) は、Amazon EBS に最大 25 Gbps のネットワーク帯域幅と最大 14 Gbps の専用帯域幅を C5 インスタンスで提供します。AWS DMS は特に、異機種間 (例:Oracle から PostgreSQL) の移行やレプリケーションを実行する場合に、CPU を大量に消費する可能性があります。C5 インスタンスは、このような状況に適しています。

[Storage (ストレージ)]

選択した インスタンスクラスに応じて、レプリケーション インスタンスには 50 GB または 100 GB のデータストレージが付属しています。このストレージは、ロード中に収集されるログファイルおよびキャッシュされた変更に使用されます。ソースシステムがビジー状態であるか、大量のトランザクションを実行する場合は、ストレージを増やす必要がある場合があります。レプリケーション サーバーで複数のタスクを実行している場合は、ストレージを増やすことも必要になる場合があります。ただし、通常はデフォルトの量で十分です。

AWS DMS 内のすべてのストレージボリュームは GP2 または汎用ソリッド ステート ドライブ (SSD) です。GP2 ボリュームには、1 秒あたり 3 つの I/O オペレーション (IOPS) のベースパフォーマンスがあり、クレジットベースで最大 3,000 IOPS をバーストできます。経験則として、`ReadIOPS` と `WriteIOPS`のレプリケーション　インスタンスのメトリクスを確認します。これらの値の合計が、そのボリュームのベースパフォーマンスを超えないようにしてください。

マルチ AZ

マルチ AZ インスタンスを選択すると、ストレージ障害から移行を保護できます。ほとんどの移行は一時的なものであり、長期間実行することを意図していません。継続的レプリケーションで AWS DMS を使用する場合、マルチ AZ インスタンスを選択すると、ストレージの問題が発生した場合に可用性が向上します。

フルロード中に単一の AZ またはマルチ AZ レプリケーションインスタンスを使用して、フェイルオーバーまたはホストの交換が発生すると、フルロードのタスクが失敗することが予想されま

す。移行が完了しなかったテーブル、またはエラー状態にある残りのテーブルについては、障害が発生した時点からタスクを再開できます。

複数のテーブルを並列的にロードする

デフォルトでは AWS DMS は、一度に 8 つのテーブルをロードします。dms.c4.xlarge またはより大きなインスタンスなどの大量レプリケーションサーバーを使用する際、これを少し増やすことによりパフォーマンスがある程度向上する場合があります。ただし、ある時点で並列処理を増やすことによりパフォーマンスが低下します。dms.t2.medium などのレプリケーションサーバーが比較的小さい場合は、並行してロードされるテーブルの数を減らすことをおすすめします。

AWS Management Console でこの数を変更するには、コンソールを開き、[Tasks] (タスク) を選択後、タスクの作成または変更を選択してから、[Advanced Settings] (詳細設定) を選択します。[Tuning Settings] (チューニング設定) で、[Maximum number of tables to load in parallel] (並列にロードするテーブルの最大数) オプションを変更します。

AWS CLI を使用してこの数を変更するには、`TaskSettings` の `MaxFullLoadSubTasks` パラメータを変更します。

並列全ロードの使用

パーティションとサブパーティションに基づいて、Oracle、Microsoft SQL Server、MySQL、Sybase、および IBM Db2 LUW ソースからの並列ロードを使用できます。これにより、全体の全負荷時間が短縮されます。さらに、AWS DMS 移行タスクの実行中に、大きなテーブルまたはパーティションテーブルの移行を高速化できます。これを行うには、テーブルをセグメントに分割し、同じ移行タスクでセグメントを並行してロードします。

並列ロードを使用するには、`parallel-load` オプションを指定した `table-settings` タイプのテーブルマッピングルールを作成します。`table-settings` ルール内では、並列でロードする 1 つ以上のテーブルに選択条件を指定します。選択条件を指定するには、`parallel-load` の `type` 要素を次のいずれかに設定します。

- `partitions-auto`
- `subpartitions-auto`
- `partitions-list`
- `ranges`
- `none`

これらの設定の詳細については、「テーブルとコレクション設定のルールとオペレーション」をご参照ください。

インデックス、トリガーおよび参照整合性制約の使用

インデックス、トリガーおよび参照整合性制約は、移行パフォーマンスに影響を及ぼすことがあるため、移行が失敗する原因になります。これらが移行に及ぼす影響は、レプリケーション タスクが全ロードタスクであるか、継続的なレプリケーション (CDC) タスクであるかによって異なります。

全ロードタスクの場合は、プライマリキーインデックス、参照整合性制約、データ操作言語 (DML) トリガーを削除することをお勧めします。または、全ロードタスクが完了するまで、その作成を遅延させることができます。全ロードタスク中、インデックスは不要であり、存在する場合、インデックスはメンテナンスのオーバーヘッドが発生します。全ロードタスクは一度にテーブルのグループをロードするため、参照整合性の制約に違反します。同様に、挿入、更新、および削除トリガーは、たとえば、以前に一括ロードされたテーブルに対して行の挿入がトリガーされた場合にエラーを引き起こす可能性があります。他のタイプのトリガーも、追加された処理のためにパフォーマンスに影響を及ぼします。

データボリュームが比較的少なく、追加の移行時間が重要でない場合は、全ロードタスクの前にプライマリ キーとセカンダリ インデックスを作成できます。参照整合性制約とトリガーは常に無効にする必要があります。

全ロード + CDC タスクの場合は、CDC フェーズの前にセカンダリ インデックスを追加することをお勧めします。AWS DMS は論理的なレプリケーションを使用するため、DML オペレーションをサポートするセカンダリ インデックスをインプレースして、完全なテーブル スキャンを防止する必要があります。CDC フェーズの前にレプリケーション タスクを一時停止して、そのタスクを再開する前にインデックスを作成後、トリガーを作成し、参照整合性制約を作成することができます。

カットオーバーの直前にトリガーを有効にする必要があります。

バックアップとトランザクションログを無効にする

Amazon RDS データベースに移行する場合、カットオーバーの準備ができるまでは、ターゲットのバックアップとマルチ AZ を無効にすることをお勧めします。同様に、以外の Amazon RDS システムに移行する場合、カットオーバーまではターゲットのロギングを無効にすることをお勧めします。

複数のタスクを使用する

時には、単一の移行のために複数のタスクを使用することでパフォーマンスが向上します。共通のトランザクションに関与しないテーブルのセットがある場合、複数のタスクに移行を分割できる場合があります。トランザクションの整合性はタスク内に維持されます。従って、個別のタス

クのテーブルが、共通のトランザクションに関与しないことが重要です。また、各タスクはそれぞれ独立してトランザクションのストリームを読み込みため、ソースデータベースに過度のストレスをかけないよう注意が必要です。

複数のタスクを使用して、レプリケーションの個別のストリームを作成できます。これにより、ソース上の読み取り、レプリケーション インスタンス上のプロセス、ターゲットデータベースへの書き込みを並列化することができます。

変更処理の最適化

デフォルトでは、AWS DMS は、トランザクションの完全性を維持するトランザクションモードで変更を処理します。トランザクションの完全性を一時的に失効できる場合は、代わりに [最適化バッチ] オプションを使用できます。このオプションでは、効率的にトランザクションがグループ化され、効率化のためにバッチに適用します。バッチ最適化適用オプションを使用すると、ほとんどの場合、参照整合性の制約に違反します。そのため、移行プロセス中にこれらの制約をオフにし、カットオーバー プロセスの一環として再度オンにすることをお勧めします。

独自のオンプレミスネームサーバーの使用

通常、AWS DMS レプリケーション インスタンスは、Amazon EC2 インスタンスでドメインネームシステム (DNS) リゾルバーを使用してドメイン エンドポイントを解決します。ただし、Amazon Route 53 Resolver を使用する場合は、独自のオンプレミス ネームサーバーを使用して特定のエンドポイントを解決できます。このツールを使用すると、オンプレミスとの間でクエリを実行できます。AWS インバウンドおよびアウトバウンド エンドポイント、転送ルール、プライベート接続を使用する。オンプレミスのネームサーバーを使用するメリットには、セキュリティの向上とファイアウォールの背後にある使いやすさが含まれます。

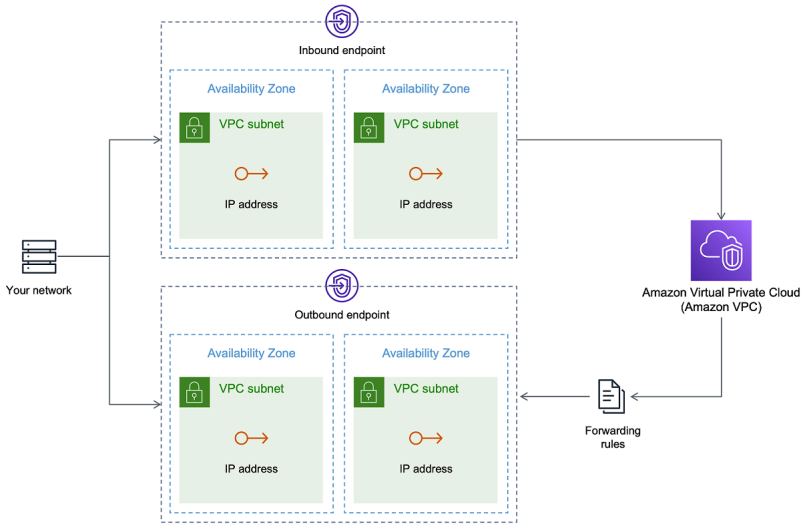
インバウンド エンドポイントがある場合、オンプレミスの DNS クエリを使用して AWS にホストされたドメインを解決できます。。エンドポイントは、リゾルバーを提供する各サブネットの IP アドレスを割り当てることによって設定されます。オンプレミスの DNS インフラストラクチャと AWS の間の接続を確立するには、AWS Direct Connectまたは、仮想プライベートネットワーク (VPN) を使用します。

アウトバウンド エンドポイントは、オンプレミスのネームサーバーに接続します。ネームサーバーは、許可リストに含まれ、アウトバウンド エンドポイントに設定された IP アドレスにのみアクセス権を付与します。ネームサーバーの IP アドレスがターゲットの IP アドレスです。アウトバウンド エンドポイントのセキュリティグループを選択するときは、レプリケーション インスタンスで使用されるものと同じセキュリティグループを選択します。

転送ルールは、ネームサーバーに転送するドメインを選択するために使用されます。1 つのアウトバウンド エンドポイントに複数の転送ルールが存在する可能性があります。転送ルールの範囲は、Virtual Private Cloud (VPC) です。VPC に関連付けられた転送ルールを使用すると、AWS クラウドの論理的に分離されたセクションをプロビジョニングできます。この論理的に隔離されたセクションから、AWS 仮想ネットワーク内のリソースを起動できます。

オンプレミスの DNS インフラストラクチャ内でホストされるドメインは、アウトバウンド DNS クエリを有効にする条件付き転送ルールとして構成できます。これらのドメインの 1 つに対してクエリが実行されると、ルールによって構成された DNS サーバーに DNS リクエストを転送する試みがトリガーされます。繰り返しますが、プライベート接続オーバー AWS Direct Connect または VPN が必要です。

Route 53 レゾルバーのアーキテクチャを次の図表に示します。



Route 53 DNS リゾルバーの詳細については、Amazon Route 53 デベロッパーガイドの「Route 53 レゾルバーの使用開始」をご参照ください。

Amazon Route 53 Resolver の AWS DMS での使用

Amazon Route 53 Resolver AWS DMS を使用してエンドポイントを解決するためのオンプレミスネームサーバーを作成できます。

Route 53 に基づく AWS DMS のオンプレミスネームサーバーを作成するには

1. AWS Management Console にサインインし、Route 53 コンソール (https://console.aws.amazon.com/route53/) を開きます。

2. Route 53 コンソールで、Route 53 リゾルバーを設定する AWS リージョンを選択します。Route 53 リゾルバーはリージョン固有です。
3. クエリの方向 (インバウンド、アウトバウンド、またはその両方) を選択します。
4. インバウンドクエリ設定を指定します。
    a. エンドポイント名を入力し、VPC を選択します。
    b. VPC 内からサブネットを 1 つ以上割り当てます (たとえば、可用性のため 2 つを選択します)。
    c. これらのサブネットから、エンドポイントとして使用する特定の IP アドレスを割り当てるか、Route 53 リゾルバーで自動的に割り当てます。
5. VPC 内のワークロードが DNS クエリを DNS インフラストラクチャにルーティングできるように、オンプレミスドメインのルールを作成します。
6. オンプレミス DNS サーバーの 1 つ以上の IP アドレスを入力し、ルールを作成します。
7. ルールを提出してください。

すべてが作成され、VPC はインバウンドおよびアウトバウンド ルールに関連付けられ、トラフィックのルーティングを開始できます。

Route 53 リゾルバーの詳細については、Amazon Route 53 デベロッパーガイドの「Route 53 Resolver の使用開始」をご参照ください。

ラージバイナリオブジェクト (LOB) の移行

一般に、AWS DMS は LOB データを 2 つのフェーズに移行します。

1. AWS DMS は、ターゲットテーブルに新しい行を作成し、その行に関連する LOB 値を除くすべてのデータを入力します。
2. AWS DMS は、ターゲットテーブルの行を LOB データで更新します。

この LOB の移行プロセスでは、移行中はターゲットテーブルのすべての LOB 列で null が許容されるようにする必要があります。これは、ソーステーブルの LOB 列で null が許容されていない場合にも該当します。AWS DMS がターゲットテーブルを作成すると、LOB 列はデフォルトで NULL に設定されます。場合によっては、インポートやエクスポートなどの他のメカニズムを使用してターゲット テーブルを作成することがあります。このような場合、移行タスクを開始する前に LOB 列で NULL が許容されることを確認してください。

この要件には 1 つの例外があります。Oracle ソースから Oracle ターゲットへの同種移行を実行し、[制限付き LOB モード] を選択しているとします。この場合、すべての LOB 値を含めて、行全体が一度に入力されます。このような場合、AWS DMS では、必要に応じて、ターゲットテーブルの LOB 列を null 許容制約なしで作成できます。

制限付き LOB モードの使用

AWS DMS は、移行に LOB 値が含まれている場合に、パフォーマンスと利便性のバランスをとる 2 つのメソッドを使用します。

1. [制限付き LOB モード] は、すべての LOB 値をユーザー指定のサイズ制限 (デフォルトは 32 KB) に移行します。サイズ制限を超える LOB 値は、手動で移行する必要があります。[制限付き LOB モード]、すべての移行タスクのデフォルトで、通常最高のパフォーマンスが得られます。ただし、[Max LOB size](最大 LOB サイズ) パラメータの設定が正しいことを確認する必要があります。このパラメータは、すべてテーブルに対して最大の LOB サイズに設定する必要があります。
2. [完全 LOB モード] は、サイズに関係なくテーブル内のすべての LOB データを移行します。[完全 LOB モード] は、テーブル内のすべての LOB データを移動する際の利便性を提供しますが、そのプロセスがパフォーマンスに重大な影響を与える可能性があります。

PostgreSQL など一部のデータベースエンジンに対して、AWS DMS は JSON データ型を LOB のように扱います。[Limited LOB mode] (制限付き LOB モード) を使用している場合、[Max LOB size] (最大 LOB サイズ) オプションが、JSON データが切り捨てられない値に設定されていることを確認します。

AWS DMS は、大きなオブジェクトのデータ型 (BLOB、CLOB、NCLOB) の使用を完全にサポートするようになりました。以下のソースエンドポイントは、完全に LOB サポートされています。

- Oracle
- Microsoft SQL Server
- ODBC

以下のターゲットエンドポイントは、完全に LOB サポートされています。

- Oracle
- Microsoft SQL Server

以下のターゲットエンドポイントには、制限付きの LOB サポートがあります。このターゲットエンドポイントに無制限の LOB サイズは使用できません。

- Amazon Redshift
- Amazon S3

完全な LOB サポートがあるエンドポイントについては、LOB データ型にサイズ制限を設定できます。

LOB パフォーマンスが向上しました。

LOB データの移行時に、次の異なる LOB 最適化設定を指定できます。

テーブルごとの LOB 設定

テーブルごとの LOB 設定を使用すると、一部またはすべてのテーブルのタスクレベルの LOB 設定を上書きできます。これを行うには、`table-settings` ルール内の `lob-settings` を定義します。次に、大きな LOB 値を含むテーブルの例を示します。

```
SET SERVEROUTPUT ON
CREATE TABLE TEST_CLOB
(
ID NUMBER,
C1 CLOB,
C2 VARCHAR2(4000)
);
DECLARE
bigtextstring CLOB := '123';
iINT;
BEGIN
WHILE Length(bigtextstring) <= 60000 LOOP
bigtextstring := bigtextstring || '00000000000000000000000000000000';
END LOOP;
INSERT INTO TEST_CLOB (ID, C1, C2) VALUES (0, bigtextstring,'AnyValue');
END;
/
SELECT * FROM TEST_CLOB;
COMMIT
```

次に、移行タスクを作成し、新しい lob-settings ルールを使用してテーブルの LOB 処理を変更します。bulk-max-siz 値は、最大 LOB サイズ (KB) を決定します。指定されたサイズより大きい場合は切り捨てられます。

```
{
 "rules": [{
   "rule-type": "selection",
   "rule-id": "1",
   "rule-name": "1",
   "object-locator": {
    "schema-name": "HR",
    "table-name": "TEST_CLOB"
   },
   "rule-action": "include"
  },
  {
   "rule-type": "table-settings",
   "rule-id": "2",
   "rule-name": "2",
   "object-locator": {
    "schema-name": "HR",
    "table-name": "TEST_CLOB"
   },
   "lob-settings": {
    "mode": "limited",
    "bulk-max-size": "16"
   }
  }
 ]
}
```

たとえ AWS DMS タスクが FullLobMode ： true で作成されても、テーブルごとの LOB 設定で AWS DMS はこの特定テーブルの LOB データを 16,000 に切り捨てるようになります。タスクログを確認して、これを確認できます。

```
721331968: 2018-09-11T19:48:46:979532 [SOURCE_UNLOAD] W: The value of column 'C' in
 table
'HR.TEST_CLOB' was truncated to length 16384
```

インライン LOB 設定

AWS DMS タスクを作成するときは、LOB モードにより、LOB の処理方法が決まります。

フル LOB モードと限定 LOB モードでは、それぞれ独自の利点とデメリットがあります。インライン LOB モードは、フル LOB モードと制限付き LOB モードの両方の利点を組み合わせたものです。

小さい LOB と大きい LOB を両方ともレプリケーションの必要があり、ほとんどの LOB が小さい場合は、インライン LOB モードを使用できます。このオプションを選択すると、フルロード中に AWS DMS タスクは小さな LOB をインラインで転送するため、効率が向上します。AWS DMS タスクは、ソース テーブルからルックアップを実行して、大きな LOB を転送します。

変更処理中に、ソーステーブルからルックアップを実行して、小規模と大規模 LOB ともレプリケーションされます。

インライン LOB モードを使用する場合、AWS DMS タスクは、すべての LOB サイズをチェックして、インラインで転送するサイズを決定します。指定されたサイズよりも大きい LOB は、フル LOB モードを使用してレプリケーションされます。従って、ほとんどの LOB が指定された設定よりも大きいことがわかっている場合は、このオプションを使用しない方がよいでしょう。代わりに、無制限の LOB サイズを許可します。

このオプションは、`FullLobMode` が `true` に設定されているときのみ利用可能なタスク設定の属性 `InlineLobMaxSize` を使用して構成します。`InlineLobMaxSize` のデフォルト値は 0 で、範囲は 1 ~ 102,400 キロバイト (100 MB) です。

たとえば、次の AWS DMS タスク設定を使用できます。ここで `InlineLobMaxSize` 値を 5 に設定すると、5 KiB (5,120 バイト) 以下のすべての LOB がインラインで転送されます。

```
{
  "TargetMetadata": {
  "TargetSchema": "",
  "SupportLobs": true,
  "FullLobMode": true,
  "LobChunkSize": 64,
  "LimitedSizeLobMode": false,
  "LobMaxSize": 32,
  "InlineLobMaxSize": 5,
  "LoadMaxFileSize": 0,
  "ParallelLoadThreads": 0,
  "ParallelLoadBufferSize":0,
  "BatchApplyEnabled": false,
  "TaskRecoveryTableEnabled": false},
  . . .
}
```

大規模なテーブルを移行する場合のパフォーマンスの向上

大規模なテーブルを移行する際のパフォーマンスを向上するには、移行を複数のタスクに分割します。行フィルタリングを使用して複数のタスクへの移行を分割するには、キーまたはパーティションキーを使用します。たとえば、1〜8,000,000 の整数プライマリキー ID がある場合、行フィルタリングを使用して 8 つのタスクを作成し、それぞれ 100 万レコードを移行できます。

コンソールで行フィルタリングを適用するには、次を実行します。

1. AWS Management Consoleを開きます。
2. [タスク] を選択して、新しいタスクを作成します。
3. [Table mappings] (テープル マッピング) を選択し、[Selection rules] (選択ルール) を展開します。
4. [Add new selection rule] (新規選択ルールの追加) を選択します。これで、次以下:、次以上:、次と等しい:、または 2 つの値の範囲の条件の列フィルターを追加できます。列フィルターの詳細については、「コンソールからテーブル選択および変換を指定する」を参照してください。

日付でパーティション化された大規模なパーティションテーブルがある場合は、日付に基づいてデータを移行できます。たとえば、月別にパーティション化されたテーブルがあり、現在の月のデータのみが更新されたとします。この場合、静的な毎月のパーティションごとに全ロードタスクを作成し、現在更新されているパーティションに対して全ロード + CDC タスクを作成することができます。

テーブルに単一列のプライマリキーまたは固有インデックスがある場合は、AWS DMS タスクで範囲タイプの並列ロードを使用してテーブルをセグメント化し、データを並列にロードすることができます。詳細については、「テーブルとコレクション設定のルールとオペレーション」を参照してください。

継続的なレプリケーション

AWS DMS は、ソースとターゲットのデータベースを同期させたまま、データの継続的なレプリケーションを行います。これは、限られた量のデータ定義言語 (DDL) のみレプリケーションします。AWS DMS は、インデックス、ユーザー、権限、ストアドプロシージャ、およびテーブルデータに直接関係しない他のデータベースの変更などの項目を反映しません。

継続的レプリケーションの予定があれば、レプリケーション インスタンスで [Multi-AZ] (マルチ AZ) オプションを有効にする必要があります。[Multi-AZ] (マルチ AZ) オプションは、レプリケーションインスタンスに対して高可用性とフェイルオーバーに対応しています。ただし、このオプションに

よって、パフォーマンスに影響を及ぼす可能性があり、ターゲット システムに変更を適用するときにレプリケーションの速度が低下する可能性があります。

ソース データベースまたはターゲット データベースをアップグレードする前に、これらのデータベースで実行されている AWS DMS のタスクを停止することをお勧めします。アップグレードが完了したら、タスクを再開します。

継続的レプリケーションでは、ソース データベース システムとAWS DMS レプリケーション インスタンス間のネットワーク帯域幅を特定することが重要です。継続的レプリケーション中にネットワークがボトルネックを引き起こさないようにしてください。

また、ソース データベース システムの 1 時間あたりの変更率とアーカイブログ生成率を特定することも重要です。これにより、継続的レプリケーション中に発生する可能性のあるスループットを理解するのに役立ちます。

ソースデータベースのロードを縮小する

AWS DMS は、ソースデータベースでいくつかのリソースを使用します。全ロードタスク中、AWS DMS では、並行処理される各テーブルに対してソーステーブルのテーブル全体のスキャンが実行されます。さらに、移行の一環として作成する各タスクは、CDC プロセスの一部としてソースに変更を問い合わせます。AWS DMS で、Oracle などの一部のソースの CDC を実行する場合は、データベースの変更ログに書き込むデータ量を増やす必要がある場合があります。

ソースデータベースに過度に負荷が掛かっている場合は、移行のタスクの数または、タスクごとのテーブルの数を減らすことができます。各タスクは独立してソースの変更を取得するため、タスクを統合することで変更キャプチャの作業負荷を減らすことができます。

ターゲットデータベースのボトルネックを減らす

移行中に、ターゲットデータベース上の書き込みリソースと競合するプロセスをすべて削除してください。

- 不要なトリガーをオフにします。
- 初期ロード時にセカンダリ インデックスをオフにし、継続的レプリケーション中に後でオンに戻します。
- Amazon RDS データベースでは、カットオーバーまでバックアップとマルチ AZ をオフにすることをお勧めします。

- 非 RDS システムに移行する場合、カットオーバーまでターゲット上のロギングをオフにすることをお勧めします。

移行中にデータ検証を使用する

データがソースからターゲットに正確に移行されたことを確認するため、データ検証が推奨されます。タスクの検証を有効にすると、AWS DMS は、テーブルに対して全ロードが実行された後で、ソースデータとターゲットデータの比較をすぐに開始します。

データ検証は、AWS DMS がソースとターゲットのエンドポイントでサポートしている次のデータベースで動作します。

- Oracle
- PostgreSQL
- MySQL
- MariaDB
- Microsoft SQL Server
- Amazon Aurora MySQL 互換エディション
- Amazon Aurora PostgreSQL 互換エディション
- IBM Db2 LUW
- Amazon Redshift

詳細については、「AWS DMS データ検証」を参照してください。

メトリクスを使用する AWS DMS タスクのモニタリング

AWS DMS コンソールを使用して、タスクのメトリクス モニタリング用にいくつかのオプションがあります:

ホスト メトリクス

ホストメトリクスは、特定のレプリケーション インスタンスのCloudWatch メトリクス タブにあります。ここでは、レプリケーション インスタンスのサイズが適切かどうかを監視できます。

レプリケーションタスクのメトリクス

受信およびコミットされた変更、レプリケーションホストとソース/ターゲットデータベース間のレイテンシーなどのレプリケーションタスクのメトリクスは、特定のタスクのCloudWatch メトリクスタブにあります。

テーブル メトリクス

個々のテーブル メトリクスは、[Table statistics] (テーブルの統計) タブで個々のタスクごとに表示されます。これらのメトリクスには、次の数値が含まれます。

- 全ロード中にロードされた行。
- タスクの開始後に挿入、更新、削除を行います。
- タスク開始後の DDL オペレーション。

モニタリング メトリクスの詳細については、「AWS DMS タスクのモニタリング」をご参照ください。

イベントと通知

AWS DMS は Amazon SNS を使用して、レプリケーション インスタンスが作成されたときや削除されたときなど AWS DMS イベントが発生したとき通知を送信します。これらの通知は、AWS リージョンについて Amazon SNS でサポートされている任意の形式で操作できます。これには、電子メールメッセージ、テキストメッセージ、または HTTP エンドポイントへの呼び出しが含まれます。

詳細については、「AWS Database Migration Service での Amazon SNS イベントと通知の使用」を参照してください。

タスクログを使用して、移行の問題のトラブルシューティングをする

場合によっては、AWS DMS で、タスクログでのみ表示可能な問題が発生することがあります。特に、データ切り捨て問題または外部キー違反による行の拒否は、タスクログに書き込まれます。そのため、データベースを移行するときは、タスクログを確認することが重要です。タスクログを表示するには、タスク作成 CloudWatch の一部として Amazon を設定します。

詳細については、「Amazon を使用したレプリケーションタスクのモニタリング CloudWatch」を参照してください。

Time Travel を使用したレプリケーションタスクのトラブルシューティング

AWS DMS の移行に関する問題のトラブルシューティングには、Time Travel を利用できます。Time Travel の詳細については、「Time Travel タスクの設定」を参照してください。

Time Travel の使用を開始するには、次の考慮事項に注意します。

- DMS レプリケーションインスタンスのオーバーヘッドを避けるため、デバッグが必要なタスクに対してのみ Time Travel をオンにします。
- Time Travel を使用して数日間実行される可能性があるレプリケーションタスクのトラブルシューティングを行う場合は、リソースのオーバーヘッドがないかレプリケーションインスタンスのメトリクスをモニタリングします。ソースデータベースで長時間にわたり、高いトランザクション負荷がかかる場合には特にこのアプローチをとることが適切となります。詳細については、「AWS DMS タスクのモニタリング」を参照してください。
- Time Travel タスクの設定で`EnableRawData` が `true` と設定されている場合、DMS のレプリケーション中のタスクのメモリ使用量は、Time Travel が有効になっていない場合よりも高くなる場合があります。Time Travel を長時間オンにする場合は、タスクをモニタリングします。
- 現時点では、Time Travel はタスクレベルでのみ有効にできます。すべてのテーブルへの変更は Time Travel ログに記録されます。トランザクション量の多いデータベース内の特定のテーブルをトラブルシューティングする場合は、別のタスクを作成します。

Oracle ターゲットのユーザーおよびスキーマの変更

Oracle をターゲットとして使用するとき、AWS DMS はターゲット　エンドポイントのユーザーが所有するスキーマにデータを移行します。

たとえば、`PERFDATA` という名前のスキーマを Oracle のターゲット　エンドポイントに移行するとして、ターゲット　エンドポイントのユーザー名が `MASTER` だとします。AWS DMS は Oracle ターゲットを `MASTER` として接続し、`MASTER` スキーマに `PERFDATA` からのデータベースオブジェクトを代入します。

この動作をオーバーライドするには、スキーマ変換を指定する必要があります。たとえば、PERFDATA スキーマ オブジェクトをターゲット エンドポイントの PERFDATA スキーマに移行する場合、次の変換を使用できます。

```
{
    "rule-type": "transformation",
    "rule-id": "2",
    "rule-name": "2",
    "object-locator": {
        "schema-name": "PERFDATA"
    },
    "rule-target": "schema",
    "rule-action": "rename",
    "value": "PERFDATA"
}
```

変換の詳細については、「JSON を使用するテーブル選択および変換を指定する」をご参照ください。

Oracle ターゲットでテーブルとインデックスのテーブル スペースを変更する

Oracle をターゲットとして使用する場合、AWS DMS はすべてのテーブルとインデックスを、ターゲットのデフォルトのテーブルとインデックステーブルスペースに移行します。たとえば、ソースが Oracle ではないデータベースエンジンだとします。すべてのターゲットテーブルとインデックスは、同じデフォルトのテーブルスペースに移行されます。

この動作をオーバーライドするには、該当するテーブルスペース変換を指定する必要があります。たとえば、テーブルとインデックスをソースのスキーマから名付けられた Oracle ターゲットのテーブルとインデックスのテーブルスペースに移行するとします。この場合、次のような変換を使用できます。ここでは、ソース スキーマは INVENTORY と名付けられ、ターゲットの該当するテーブルとインデックスのテーブル スペースはそれぞれ INVENTORYTBL および INVENTORYIDX と名付けられます。

```
{
   "rule-type": "transformation",
   "rule-id": "3",
   "rule-name": "3",
   "rule-action": "rename",
```

```
    "rule-target": "table-tablespace",
    "object-locator": {
        "schema-name": "INVENTORY",
        "table-name": "%",
        "table-tablespace-name": "%"
    },
    "value": "INVENTORYTBL"
},
{
    "rule-type": "transformation",
    "rule-id": "4
    "rule-name": "4",
    "rule-action": "rename",
    "rule-target": "index-tablespace",
    "object-locator": {
        "schema-name": "INVENTORY",
        "table-name": "%",
        "index-tablespace-name": "%"
    },
    "value": "INVENTORYIDX"
}
```

変換の詳細については、「 JSON を使用するテーブル選択および変換を指定する」をご参照ください。

Oracle がソースとターゲットの両方である場合、Oracle ソースの追加の接続属性 enableHomogenousTablespace=true を設定することで、既存のテーブルまたはインデックステーブルスペースの割り当てを保持できます。詳細については、「のソースとして Oracle を使用する場合のエンドポイント設定 AWS DMS」を参照してください。

レプリケーション インスタンスバージョンのアップグレード

AWS では、新機能とパフォーマンスの強化を含め、定期的に新しいバージョンの AWS DMS レプリケーション エンジン ソフトウェアをリリースしています。レプリケーション エンジン ソフトウェアの各バージョンには、他のバージョンと区別するための独自のバージョン番号があります。レプリケーション インスタンスを新しいバージョンにアップグレードする前に、AWS DMS の本番作業ロードを実行しているレプリケーションインスタンスの既存バージョンを必ずテストします。利用可能なバージョン アップグレードの詳細は、「AWS DMS リリースノート」をご参照ください。

移行コストについて

AWS Database Migration Service で、AWS にデータベースを簡単かつ安全にローコストで移行できます。レプリケーション インスタンスと追加のログストレージに対してのみ料金が発生します。各データベース移行インスタンスには、ほとんどのレプリケーションでスワップ領域、レプリケーション ログ、およびデータキャッシュに十分なストレージが含まれており、インバウンドデータ転送は無料です。

初期ロード時またはピークロード時にさらに多くのリソースが必要になる場合があります。クラウドウォッチ メトリクスを使用して、レプリケーション インスタンスのリソース使用率を綿密に監視できます。その後、使用状況に基づいてレプリケーション インスタンスのサイズをスケールアップおよびスケールダウンできます。

移行コストの見積りの詳細については、以下をご参照ください。

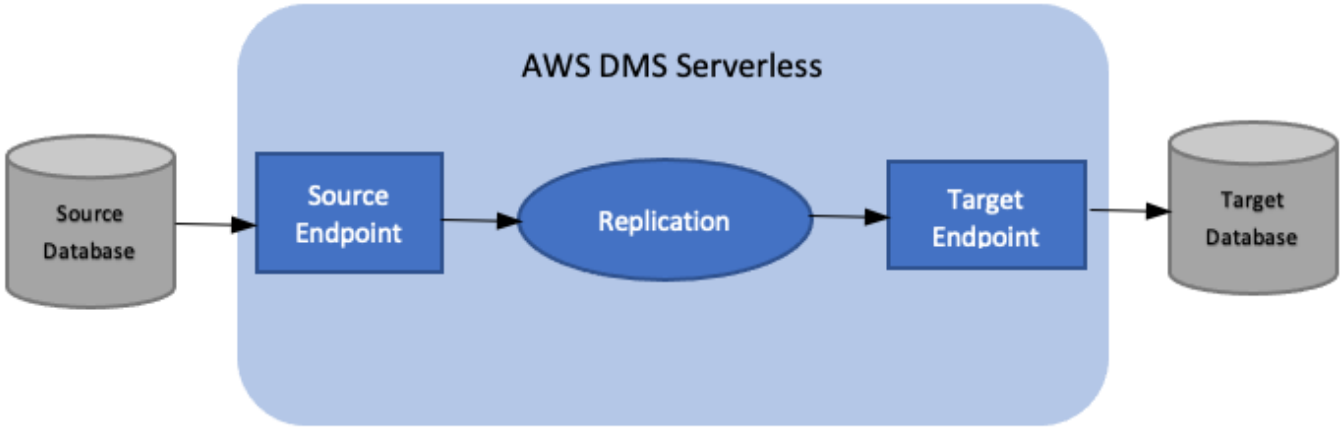
- AWS Database Migration Service 料金表
- AWS 料金計算機

Serverless AWS DMS の使用

AWS DMS サーバーレスは、自動プロビジョニング、スケーリング、組み込みの高可用性、 pay-for-use 請求モデルを提供し、運用の俊敏性を高め、コストを最適化する機能です。サーバーレスという特徴により、容量の見積もり、プロビジョニング、コスト最適化、レプリケーションエンジンのバージョン管理やパッチ適用などのレプリケーションインスタンス管理タスクの必要性が最小限に抑えられます。

AWS DMS Serverless では、 の現在の機能 AWS DMS （ このドキュメントでは AWS DMS 標準と呼びます) と同様に、エンドポイントを使用してソース接続とターゲット接続を作成します。ソースエンドポイントとターゲットエンドポイントを作成した後、特定のレプリケーションの設定を含むレプリケーション設定を作成します。このレプリケーションを起動、停止、変更、または削除して、レプリケーションを管理できます。各レプリケーションには、データベース移行の要件に応じて構成できる設定があります。これらの設定は、JSON ファイルまたは の AWS DMS セクションを使用して指定します AWS Management Console。レプリケーション設定の詳細については、AWS DMS 「エンドポイントの使用」を参照してください。レプリケーションの開始後、 AWS DMS Serverless はソースデータベースに接続し、データベースのメタデータを収集してレプリケーションのワークロードを分析します。このメタデータを使用して、 は必要な容量を AWS DMS 計算してプロビジョニングし、データレプリケーションを開始します。

次の図は、 AWS DMS サーバーレスレプリケーションプロセスを示しています。



> **Note**
>
> AWS DMS Serverless はデフォルトのエンジンバージョンを使用します。デフォルトのエンジンのバージョンの詳細については、「リリースノート」を参照してください。

AWS DMS Serverless の詳細については、以下のトピックを参照してください。

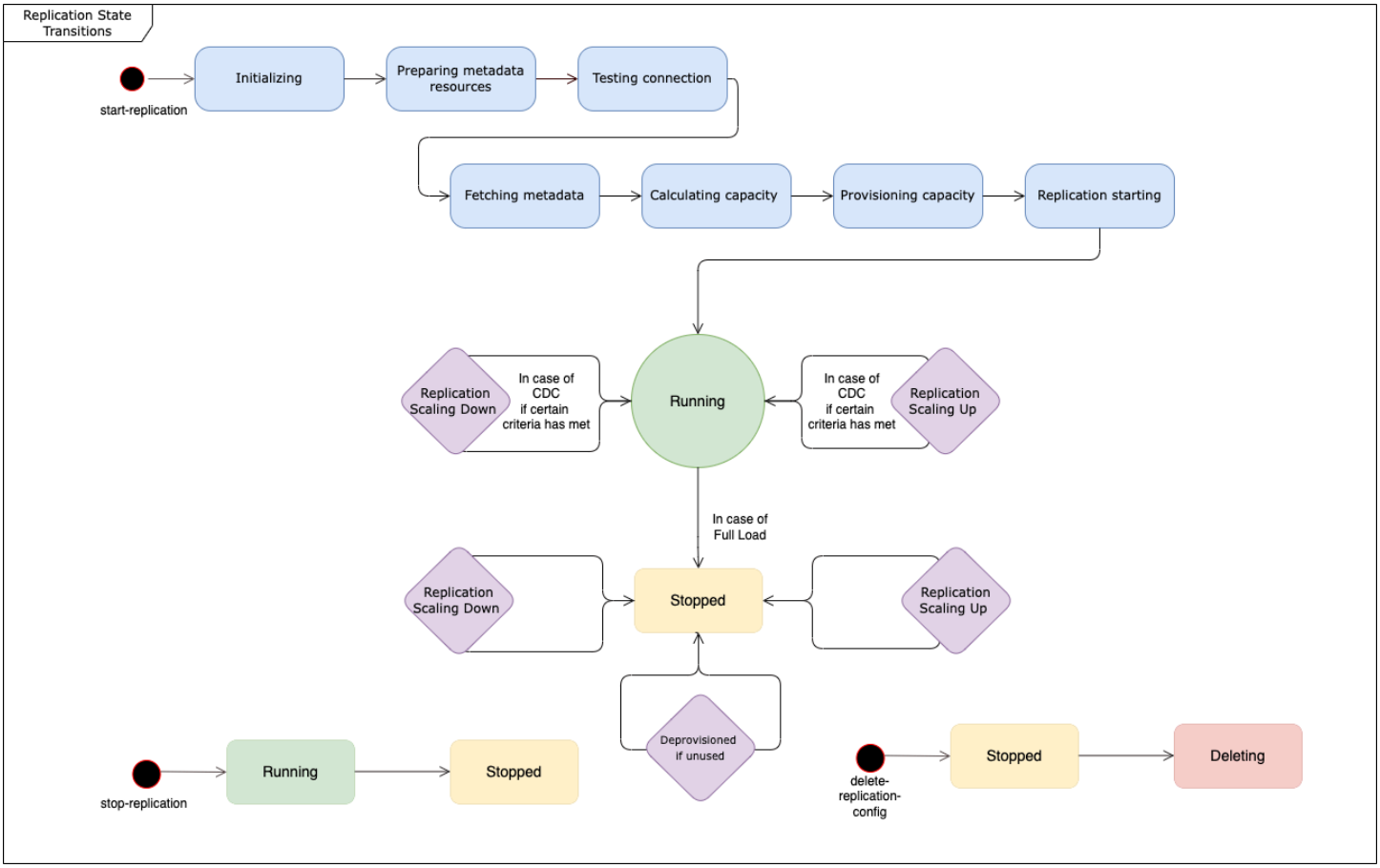
トピック

- AWS DMS サーバーレスコンポーネント
- AWS DMS サーバーレスの制限

AWS DMS サーバーレスコンポーネント

レプリケーションの実行に必要なリソースを管理するために、 AWS DMS Serverless には、サービスによって実行されるさまざまな内部アクションを明らかにする詳細な状態があります。レプリケーションを開始すると、 AWS DMS Serverless は容量負荷を計算して、計算した容量をプロビジョンし、次のとおり、レプリケーションのステータスに応じてデータレプリケーションを開始します。

次の図は、 AWS DMS サーバーレスレプリケーションの状態遷移を示しています。



- レプリケーションを開始した後の最初のステータスは「初期化中」です。このステータスでは、必要なパラメータがすべて初期化されます。
- その直後のステータスには、「メタデータリソースを準備しています」、「接続をテストしています」、「メタデータを取得しています」などがあります。これらの状態では、 AWS DMS Serverless はソースデータベースに接続して、必要な容量を予測するために必要な情報を取得します。
  - レプリケーション状態が接続のテストの場合、 AWS DMS Serverless はソースデータベースとターゲットデータベースへの接続が正常にセットアップされていることを確認します。
  - 「接続をテストしています」の次のレプリケーションステータスは、「メタデータを取得しています」です。ここでは、容量の計算に必要な情報 AWS DMS を取得します。
  - が必要な情報 AWS DMS を取得すると、次の状態は容量の計算です。このステータスの場合、システムはレプリケーションの実行に必要となる基盤のリソースのサイズを計算します。
- 「キャパシティを計算しています」の次に遷移するステータスは、「キャパシティをプロビジョニングしています」です。レプリケーションのステータスがこの状態の場合、 AWS DMS Serverless は基盤となるコンピューティングリソースを初期化します。

- すべてのリソースのプロビジョニングが正常に完了した後のレプリケーションのステータスは、「レプリケーション開始」です。この状態では、 AWS DMS Serverless はデータのレプリケーションを開始します。レプリケーションのフェーズには以下が含まれます。
  - 全ロード： このフェーズでは、DMS はレプリケーションの開始時と同じようにソースデータストアをレプリケートします。
  - CDC (初期): このフェーズでは、DMS はフルロードフェーズ中に発生した変更をソースデータストアにレプリケートします。DMS は、`StopTaskCachedChangesNotApplied`タスク設定が の場合にのみ、このフェーズを実行します`false`。
  - CDC (進行中): 初期 CDC フェーズの後、DMS は変更が発生したときにソースデータベースにレプリケートします。DMS は、`StopTaskCachedChangesApplied`タスク設定が の場合にのみ、最初の CDC フェーズ後にレプリケーションを実行します`false`。
- 最後のステータスは、「実行中」です。実行中ステータスの場合、データレプリケーションが進行中です。
- 停止したレプリケーションは停止状態になります。停止したレプリケーションは、次の状況で再起動できます。
  - DMS がプロビジョニング解除したレプリケーションを再起動することはできません。
  - 停止した CDC 専用または全ロードおよび CDC レプリケーションは、 StartReplicationアクションを使用して再起動できます。コンソールを使用して停止したレプリケーションを再起動することはできません。
  - PostgreSQL をエンジンとして使用する停止したレプリケーションを再起動することはできません。

このトピックには、次のセクションが含まれています。

- サポート対象エンジンバージョン
- サーバーレスレプリケーションの作成
- AWS DMS サーバーレスレプリケーションの変更
- コンピューティング設定
- AWS DMS サーバーレスの自動スケーリングについて
- AWS DMS サーバーレスレプリケーションのモニタリング
- フルロード Oracle から Amazon Redshift への移行のスループットの向上

AWS DMS Serverless の場合、 AWS DMS コンソールの左側のナビゲーションパネルに新しいオプション Serverless レプリケーション があります。サーバーレスレプリケーションを使用すると、レ

プリケーションのインスタンスタイプやタスクの代わりにレプリケーションを指定してレプリケーションを定義できます。さらに、DMS がレプリケーション用にプロビジョンする DMS キャパシティユニット (DCU) の最大値と最小値を指定できます。DCU は、レプリケーションが現在使用している DCU ごとに 2GB の RAM. AWS DMS bills アカウントです。 AWS DMS 料金の詳細については、AWS 「 Database Migration Service の料金」を参照してください。

AWS DMS は、テーブルマッピングとワークロードの予測サイズに基づいて、レプリケーションリソースを自動的にプロビジョニングします。このキャパシティユニットは、指定する最小キャパシティユニット値と最大キャパシティユニット値の範囲内の値となります。

サポート対象エンジンバージョン

AWS DMS Serverless では、サービスがその設定を処理するため、エンジンバージョンを選択および管理する必要はありません。 AWS DMS Serverless は、次のソースをサポートしています。

- Microsoft SQL Server
- PostgreSQL 互換データベース
- MySQL 互換データベース
- MariaDB
- Oracle
- IBM Db2

AWS DMS Serverless は、次のターゲットをサポートしています。

- Microsoft SQL Server
- PostgreSQL
- MySQL 互換データベース
- Oracle
- Amazon S3
- Amazon Redshift
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Managed Streaming for Apache Kafka
- Amazon OpenSearch サービス
- Amazon DocumentDB (MongoDB 互換性)

- Amazon Neptune

AWS DMS Serverless の一部として、 AWS DMS サーバーレスレプリケーションを作成、設定、起動、管理できるコンソールコマンドにアクセスできます。コンソールの [サーバーレスレプリケーション] のセクションを使用して、このようなコマンドを実行するには、次のいずれかを実行する必要があります。

- 新しい AWS Identity and Access Management (IAM) ポリシーと IAM ロールを設定して、そのポリシーをアタッチします。
- テンプレートを使用して AWS CloudFormation 、必要なアクセスを提供します。

AWS DMS Serverless では、サービスにリンクされたロール (SLR) がアカウントに存在する必要があります。 は、このロールの作成と使用 AWS DMS を管理します。必要な SLR があることを確認する方法の詳細については、「AWS DMS Serverless のサービスにリンクされたロール」を参照してください。

サーバーレスレプリケーションの作成

2 つの既存の AWS DMS エンドポイント間でサーバーレスレプリケーションを作成するには、次の手順を実行します。 AWS DMS エンドポイントの作成の詳細については、「」を参照してくださいソースおよびターゲットエンドポイントの作成。

サーバーレスレプリケーションの作成

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. ナビゲーションペインで、[サーバーレスレプリケーション] を選択して、[レプリケーションの作成] をクリックします。
3. [レプリケーションの作成] ページで、次のとおりサーバーレスレプリケーションの設定を指定します。

オプション	アクション
名前	**DMS-replication** などのレプリケーションを識別できる名前を入力する。

オプション	アクション
説明的な Amazon リソースネーム (ARN) - オプション	このオプションのパラメータを使用して、レプリケーションの説明を指定できます。
ソースデータベースエンドポイント	アカウント内の既存のエンドポイントを選択する。AWS DMS Serverless は、 AWS DMS 標準がサポートするエンドポイントタイプのサブセットのみをサポートすることに注意してください。
ターゲットデータベースエンドポイント	アカウント内の既存のエンドポイントを選択する。AWS DMS Serverless は、 AWS DMS 標準がサポートするエンドポイントタイプのサブセットのみをサポートすることに注意してください。
レプリケーションタイプ	要件に基づいて次のレプリケーションタイプから選択する。 • 全ロード: 既存のデータのみを AWS DMS 移行します。 • フルロードおよび変更データキャプチャ (CDC): レプリケーション中に発生する既存のデータと変更を AWS DMS 移行します。 • 変更データキャプチャ (CDC): レプリケーションを開始した後に発生する変更 AWS DMS のみを移行します。

[設定] セクションで、レプリケーションのニーズに応じて設定を行います。

[テーブルマッピング] セクションでは、テーブルマッピングを設定して、レプリケートするデータを選択したりフィルタリングしたりするためのルールを定義します。マッピングを指定する前に、ソースデータベースとターゲットデータベースのデータ型マッピングのドキュメントセクションを確認してください。ソースデータベースとターゲットデータベースのデータ型マッピングの詳細については、 AWS DMS エンドポイントの使用 トピックのソースエンドポイントタイプとターゲットエンドポイントタイプのデータ型セクションを参照してください。

[コンピューティング設定] セクションで、次のとおり設定します。コンピューティング設定の詳細については、「コンピューティング設定」を参照。

オプション	アクション
VPC	既存の VPC を選択する。
サブネットグループ	既存のサブネットグループを選択する。
VPC セキュリティグループ	まだ選択されていない場合はデフォルトを選択する。
AWS KMS キー	適切な KMS キーを選択します。KMS キーの詳細については、AWS Key Management Service 「 API リファレンス」の「キーの作成」を参照してください。
デプロイメント	そのままにする。
アベイラビリティーゾーン	そのままにする。
最小 DMS キャパシティユニット (DCU) - (オプション)	デフォルト値の 1 DCU を使用する場合は空白のままにする。
最大 DMS キャパシティユニット (DCU)	16 DCU を選択する。

[メンテナンス] 設定はそのままにします。

4. [レプリケーションインスタンスを作成] をクリックします。

AWS DMS は、移行を実行するためのサーバーレスレプリケーションを作成します。

AWS DMS サーバーレスレプリケーションの変更

レプリケーションの設定を変更するには、`modify-replication-config` アクションを使用します。変更できるのは、、`CREATED`、`STOPPED`または `FAILED`状態の AWS DMS レプリケーション設定のみです。`modify-replication-config` アクションの詳細については、 AWS Database Migration Service API リファレンスのModifyReplication「Config」を参照してください。

を使用してサーバーレスレプリケーション設定を変更するには AWS Management Console

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. ナビゲーションペインで [サーバーレスレプリケーション] を選択します。
3. 変更するレプリケーションを選択します。次の表では、レプリケーションの現在のステータスに基づいて実行できる変更について説明しています。

設定	説明	許可されるステータス
名前	レプリケーション名を変更できる。8 ~ 16 文字の印刷可能な ASCII 文字 (/、"、@ を除く) を含むレプリケーション名を入力する。名前は、選択した AWS リージョンのアカウントで一意でなければなりません。実行する AWS リージョンやタスクを含めるなど、名前にいくつかの詳細を追加できます。例: **`west2-mysql2mysql-config1`** 。	`ReplicationState` が、`CREATED`、`STOPPED`、または `FAILED` の場合。
ソースデータベースエンドポイント	新しい既存のソースエンドポイントをレプリケーションのソースとして選択する。	`ReplicationState` が `CREATED` または、`ProvisionState` が `null` で `FAILED` の場合。
ターゲットデータベースエンドポイント	新しい既存のターゲットエンドポイントをレプリケーションのターゲットとして選択する。	`ReplicationState` が `CREATED` または、`ProvisionState` が `null` で `FAILED` の場合。
レプリケーションタイプ	サーバーレスレプリケーションのタイプは変更できる。	`ReplicationState` が `CREATED` または、`ProvisionState`

設定	説明	許可されるステータス
		が null で FAILED の場合。
レプリケーション設定	ターゲットテーブル準備モード、LOB 列をレプリケーションに含めるかどうか、最大 LOB サイズ、検証、ログ記録などのレプリケーション設定を変更できる。詳細については、「タスク設定」を参照してください。	ReplicationState が、CREATED、STOPPED、または FAILED の場合。
テーブルマッピング	選択ルールや変換ルールなどのサーバーレスレプリケーションのテーブルマッピング設定を変更できる。詳細については、「テーブルマッピング」を参照してください。	ReplicationState が、CREATED、STOPPED、または FAILED の場合。

設定	説明	許可されるステータス

コンピューティング設定	ネットワーク設定、スケーリング設定、メンテナンス設定など、サーバーレスレプリケーションのコンピューティング設定を変更できる。コンピューティング設定の詳細については、「コンピューティング設定」を参照。	• ReplicationState が CREATED、STOPPED、または FAILED の場合、次のスケーリング設定、メンテナンス設定、ネットワーク設定を変更できる。
		  • MinCapacityUnits
		  • MaxCapacityUnits
		  • MultiAZ
		  • PreferredMaintenanceWindow
		  • VpcSecurityGroupIds
		• ReplicationState が CREATED、または ProvisionState が null で FAILED の場合、次のネットワーク設定とセキュリティ設定を変更できる。
		  • AvailabilityZone
		  • DnsNameServers
		  • KmsKeyId

設定	説明	許可されるステータス
		• ReplicationSubnetGroupId

コンピューティング設定

コンピューティング設定のパラメータまたはコンソールのセクションを使用してレプリケーションのプロビジョニングを設定します。コンピューティング設定オブジェクトには次のフィールドがあります。

オプション	説明
MinCapacity単位	これは、 がプロビジョニングする DMS キャパシティーユニット (DCU) の最小数 AWS DMS です。これは、自動スケーリングでスケールダウンできる最小の DCU でもある。
MaxCapacity単位	レプリケーションの容量予測に応じて、 AWS DMS がプロビジョンできる最大 DMS キャパシティユニット (DCU)。自動スケーリングでスケールアップできる最大 DCU でもある。
KmsKeyID	レプリケーションストレージと接続情報の暗号化に使用する暗号キー。(デフォルト) aws/dms を選択した場合、 はアカウントと に関連付けられたデフォルトの KMS キー AWS DMS を使用します AWS リージョン。説明とアカウント番号が、キーの ARN とともに表示されます。暗号化キーの使用の詳細については、「暗号化キーの設定と AWS KMS アクセス許可の指定」を参照。このチュートリアルでは、[(デフォルト) aws/dms] が選択されたままにする。
ReplicationSubnetGroupId	選択した VPC 内のレプリケーションを作成するレプリケーションサブネットグループ。ソースデータベースが VPC 内にある場合は、レプリケーションの場所として、

オプション	説明
	ソースデータベースを含むサブネットグループを選択する。レプリケーション サブネットグループの詳細については、「レプリケーション サブネットグループの作成」をご参照ください。
VpcSecurityGroupIds	レプリケーションのインスタンスが VPC 内で作成されます。ソースデータベースが VPC 内にある場合は、データベースが配置されている DB インスタンスへのアクセス許可を付与する VPC セキュリティグループを選択する。
PreferredMaintenanceウィンドウ	このパラメータは、システムメンテナンスを実行できる毎週の時間帯を協定世界時 (UTC) で定義する。デフォルトは、 ごとに 8 時間の時間ブロックからランダムに選択された 30 分の時間枠で AWS リージョン、曜日がランダムに発生します。
マルチ AZ	このオプションのパラメータを使用すると、フェイルオーバーをサポートするために別のアベイラビリティーゾーンにレプリケーションのスタンバイレプリカが作成される。変更データキャプチャ (CDC) または進行中のレプリケーションを使用する場合は、このオプションを有効にする必要がある。

AWS DMS サーバーレスの自動スケーリングについて

レプリケーションをプロビジョニングして RUNNING状態になると、 AWS DMS サービスは、変化するワークロードに適応するために基盤となるリソースの容量を管理します。この管理では、次のレプリケーション設定に基づいてレプリケーションリソースをスケールします。

- `MinCapacityUnits`
- `MaxCapacityUnits`

レプリケーションは、使用率のしきい値の上限を一定期間超えるとスケールアップし、容量使用率が長期間にわたり最小容量使用率のしきい値を下回るとスケールダウンします。

> **Note**
>
> サーバーレスレプリケーションは、全ロードの進行中にオートスケールダウンできません。

AWS DMS サーバーレスでの自動スケーリングのチューニング

レプリケーションの自動スケーリングパラメータを調整するには、 を最大値MaxCapacityUnitsに設定し、 がリソースのプロビジョニング AWS DMS を管理することをお勧めします。トランザクション量の急増に対応できるように、自動スケーリングの利点を最大限に活用するには、DCU の最大容量を最大に設定することをお勧めします。料金見積りツールには、レプリケーションで最大 DCU を継続的に使用する場合の最大月額コストが表示されます。実際には使用した容量に対してのみ課金されるため、最大 DCU は実際のコストを示すものではありません。

レプリケーションでリソースをフルキャパシティーで使用していない場合、 はコストを節約するためにリソースのプロビジョニングを AWS DMS 徐々に解除します。ただし、リソースのプロビジョニングとプロビジョニング解除には時間がかかるため、MinCapacityUnits にはレプリケーションワークロードの突発的なスパイクに対応できる値を設定することをお勧めします。これにより、がより高いワークロードレベルの AWS DMS リソースをプロビジョニングしている間、レプリケーションのプロビジョニングが不足しなくなります。

レプリケーションのアンダープロビジョニングにより、最大容量設定がデータ要件に対して低すぎるか、最小容量設定がレプリケーションワークロードの突然の急増の処理には低すぎる場合、CapacityUtilization メトリクスが常に最大値になり、レプリケーションの失敗につながる場合があります。リソースのプロビジョニング不足が原因でレプリケーションが失敗した場合、 はレプリケーションログに out-of-memory イベント AWS DMS を作成します。レプリケーションワークロードの突然の急増が原因で out-of-memory 条件が発生した場合、レプリケーションは自動的にスケーリングされ、再起動されます。

AWS DMS サーバーレスレプリケーションのモニタリング

AWS には、 AWS DMS サーバーレスレプリケーションをモニタリングし、潜在的なインシデントに対応するためのツールがいくつか用意されています。

- AWS DMS サーバーレスレプリケーションメトリクス
- AWS DMS サーバーレスレプリケーションログ

AWS DMS サーバーレスレプリケーションメトリクス

サーバーレスレプリケーションモニタリングには、次の統計の Amazon CloudWatch メトリクスが含まれます。これらの統計はサーバーレスレプリケーションごとにグループ化されています。

メトリクス	単位	説明
CapacityUtilization	割合 (%)	サーバーレスレプリケーションによるメモリ使用率
CDCIncomingChanges	割合 (%)	ターゲットへの適用を待 point-in-time っている の変更イベントの合計数。これは、ソースエンドポイントのトランザクション変更レートの測定と同じではありません。このメトリクスの数値が大きい場合は、通常 AWS DMS 、キャプチャされた変更をタイムリーに適用できず、ターゲットレイテンシーが高くなることを示します。
CDCLatencySource	[秒]	ソース エンドポイントからキャプチャされた最後のイベントと、 AWS DMS インスタンスの現在のシステム タイムスタンプの間の間隔 (秒)。CDC LatencySource は、ソースインスタンスとレプリケーションインスタンス間のレイテンシーを表します。CDC が高いということは、ソースからの変更をキャプチャするプロセスが遅れるLatencySource ことを意味します。継続的なレプリケーションのレイテンシーを特定するには、このメトリクスを CDC とともに表示できますLatencyTarget。CDC LatencySource と CDC の両方LatencyTarget が高い場合は、まず CDCLatencySource を調査します。 ソースとレプリケーションの間にレプリケーションラグがない場合、CDC は LatencySource 0 になります。また、レプリケーションがソースのトランザクションログで次のイベントを読み込もうとしたときに、ソースから最後に読み取ったときと比較して新しいイベントがない場合も、CDCLatencySource がゼロになる可能性があります。この場合、レプリケーションは CDC を 0 LatencySource にリセットします。

メトリクス	単位	説明
CDCLatencyTarget	[秒]	ターゲットのコミットを待機中の最初のイベント タイムスタンプと AWS DMS インスタンスの現在のタイムスタンプの間の間隔 (秒)。ターゲットのレイテンシーは、レプリケーションインスタンスのサーバー時間と、ターゲットコンポーネントに転送された最も古い未確定のイベント ID との差。つまり、ターゲットのレイテンシーは、レプリケーションインスタンスと、適用されても TRG エンドポイントがまだ確認していない最も古いイベントとの間のタイムスタンプの差 (99%)。CDC LatencyTarget が高い場合、ターゲットに変更イベントを適用するプロセスが遅れていることを示します。継続的なレプリケーションのレイテンシーを特定するには、このメトリクスを CDC とともに表示できますLatencySource。CDC LatencyTarget は高いが CDC LatencySource は高くない場合は、次の点を調べます。 • ターゲットにプライマリキーまたはインデックスがありません • ターゲット インスタンスまたはレプリケーション インスタンスでリソースのボトルネックが発生する • レプリケーションインスタンスとターゲットの間にネットワークの問題がある
CDC ThroughputBandwidthターゲット	KB/秒	ターゲットに送信される送信データ (KB/秒)。CDC は、サンプリングポイントで送信された送信データ ThroughputBandwidth を記録します。ネットワークトラフィックが見つからない場合、値は 0 です。CDC は長時間実行トランザクションを発行しないため、ネットワークトラフィックは記録されない場合があります。
CDC ThroughputRowsソース	行数/秒	ソースから受信した 1 秒あたりの行数単位の変更数。

メトリクス	単位	説明
CDC ThroughputRowsターゲット	行数/秒	ターゲットに送信された 1 秒あたりの行数単位の変更数。
FullLoadThroughput Bandwidthターゲット	KB/秒	ターゲットに送信される全ロードによるネットワーク帯域幅 (1 秒あたりの KB 数)。
FullLoadThroughputRowsターゲット	行数/秒	ターゲットに送信される全ロードによる変更 (1 秒あたりの行数)。

AWS DMS サーバーレスレプリケーションログ

Amazon を使用して CloudWatch 、 AWS DMS 移行プロセス中にレプリケーション情報をログに記録できます。レプリケーション設定を選択して、ログ記録を有効にします。

サーバーレスレプリケーションは、レプリケーションの進行状況の可視性を高め、トラブルシューティングを支援するために、ステータスログを CloudWatch アカウントにアップロードします。

AWS DMS は、サーバーレスにリンクされたログを、プレフィックス が付いた専用ロググループにアップロードしますdms-serverless-replication-*<your replication config resource ID>*。このロググループ内には、dms-serverless-replication-orchestrator-*<your replication config resource ID>* というログストリームがあります。このログストリームは、レプリケーションのレプリケーションステータスと、この段階で行われている作業の詳細を示す関連メッセージをレポートします。ログのエントリ例については、以下の「サーバーレスレプリケーションのログの例」を参照してください。

> **Note**
>
> AWS DMS は、レプリケーションを実行するまでロググループまたはストリームを作成しません。レプリケーションのみを作成する場合は、ロググループまたはストリームを作成しAWS DMS ません。

実行済みのレプリケーションのログを表示するには、次のステップを実行します。

1. AWS DMS コンソールを開き、ナビゲーションペインからサーバーレスレプリケーションを選択します。[サーバーレスレプリケーション] ダイアログが開きます。
2. [設定] セクションに移動し、[一般] 列の [サーバーレスログを表示する] を選択します。CloudWatch ロググループが開きます。
3. 「移行タスクログ」セクションを見つけ、 CloudWatch 「ログの表示」を選択します。

レプリケーションが失敗した場合、 はレプリケーション状態が のログエントリとfailed、失敗の理由を説明するメッセージ AWS DMS を作成します。失敗したレプリケーションのトラブルシューティングの最初のステップとして、 CloudWatch ログを確認する必要があります。

> **Note**
>
> AWS DMS Classic と同様に、データ移行自体の進行状況に関するより詳細なログ記録、つまり基盤となるレプリケーションタスクによって出力されるログを有効にするオプションがあります。次の JSON の例のとおり、レプリケーション設定で EnableLogging の Logging フィールドを true に設定することで、このようなログ記録を有効にできます。
>
> ```
> {
>   "Logging": {
>     "EnableLogging": true
>   }
> }
> ```
>
> 有効にした場合、このログはサーバーレスレプリケーションのステータスが running である間、初めて表示されます。このログは、以前のログストリームと同じロググループに表示され、新しいログストリームである dms-serverless-serv-res-id-*{unique identifier}* の下に表示されます。サーバーレスレプリケーションログを解釈する方法については、次のセクションを参照してください。

サーバーレスレプリケーションのログの例

このセクションには、サーバーレスレプリケーションのログエントリの例が記載されています。

例: レプリケーションの起動

サーバーレスレプリケーションを実行すると、 は次のようなログエントリ AWS DMS を作成します。

```
{'replication_state':'initializing', 'message': 'Initializing the replication
 workflow.'}
```

例: レプリケーションの失敗

レプリケーションのエンドポイントのいずれかが正しく設定されていない場合、 は次のようなログエントリ AWS DMS を作成します。

```
{'replication_state':'failed', 'message': 'Test connection failed for endpoint X.',
 'failure_message': 'X'}
```

失敗後のログにこのメッセージが表示されている場合は、特定されたエンドポイントが正常であり、適切に設定されているかを確認します。

フルロード Oracle から Amazon Redshift への移行のスループットの向上

AWS DMS は、Oracle から Amazon Redshift へのフルロード移行のスループットパフォーマンスを大幅に向上させます。DMS は、テーブルマッピングで `custom parallel-load`オプションを指定せずに、テーブルに対してこの機能を自動的に有効にします。カスタマイズされた並列ロードオプションを持つテーブルの場合、DMS サーバーレスは、指定されたテーブルマッピング設定に基づいてテーブルロードを分散します。拡張スループットを使用するには、次の操作を行います。

- パーティションや境界を参照しない選択ルールを指定します。例えば、テーブルマッピングのテーブル設定に が含まれている場合`parallel-load`、DMS Serverless は拡張スループット機能を使用しません。詳細については、「選択ルールと選択アクション」を参照してください。
- `MaxFileSize` と `WriteBufferSize`を 64 MB に設定します。詳細については、「AWS DMS のターゲットとして Amazon Redshift を使用する場合のエンドポイントの設定」を参照してください。
- スパースデータを含む`true`データストアの場合は `CompressCsvFiles`を、高密度データを含むデータストア`false`の場合は に設定することをお勧めします。
- 次のタスク設定を に設定します`0`。
  - `ParallelLoadThreads`
  - `ParallelLoadQueuesPerThread`
  - `ParallelApplyThreads`
  - `ParallelApplyQueuesPerThread`
  - `ParallelLoadBufferSize`

- 並列データ移行をサポートする49には、 MaxFullLoadSubTasksに設定します。
- LOB mode を inline に設定します。詳細については、「AWS DMS タスクでのソースデータベースの LOB サポートの設定」を参照してください。

AWS DMS では、次のレプリケーションのスループットパフォーマンスは向上しません。

- 並列ロードを使用するテーブルとのレプリケーション。詳細については、「選択したテーブルおよびビューさらにコレクションで並列ロードを使用する」を参照してください。
- データ変換ルールを使用したレプリケーション。
- フィルタールールを使用したレプリケーション。
- change-data-type 変換ルールとのレプリケーション。

AWS DMS サーバーレスの制限

AWS DMS サーバーレスには以下の制限があります。

- 変更できるのは、、CREATED、STOPPEDまたは FAILED状態の AWS DMS レプリケーション設定のみです。どの設定をどの条件下で変更できるかの詳細については、「AWS DMS サーバーレスレプリケーションの変更」を参照してください。
- 削除できるのは、、STOPPED、または FAILED状態の AWS DMS レプリケーション設定のみです。
- レプリケーションにつき、100GB の静的に割り当てられたストレージを使用できます。長時間実行されるトランザクションやキャッシュなどの要件により、レプリケーションでこれ以上のメモリが使用される場合は、ワークロードを個別のサーバーレスレプリケーションに分割することをお勧めします。LOB を含むすべてのレプリケーションを別のサーバーレスレプリケーションに配置するなど、ワークロードはテーブルごと、または要件ごとに分割できます。
- レプリケーションインスタンスとは異なり、 AWS DMS サーバーレスレプリケーションには管理タスク用のパブリック IP アドレスはありません。サーバーレスレプリケーションは、コンソールを使用して管理します。
- この AWS DMS サーバーレスリリースでは、 AWS DMS 標準がサポートするすべてのソースエンドポイントタイプとターゲットエンドポイントタイプをサポートしているわけではありません。サポート対象のエンジンタイプのリストについては、「AWS DMS サーバーレスコンポーネント」を参照してください。

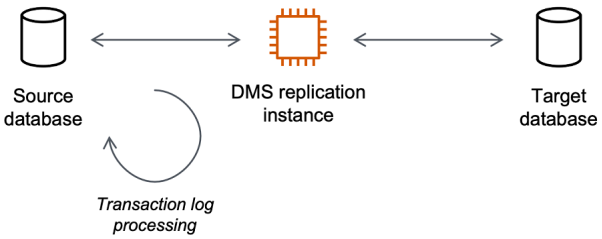
- サーバーレスレプリケーションでは VPC エンドポイントを使用して依存関係にアクセスする必要があります。次のエンドポイントタイプにアクセスするには、VPC エンドポイントを使用する必要があります。
  - Amazon S3
  - Amazon Kinesis
  - AWS Secrets Manager
  - Amazon DynamoDB
  - Amazon Redshift
  - Amazon OpenSearch サービス

  VPC エンドポイントの設定の詳細については、「AWS DMS ソースエンドポイントとターゲットエンドポイントとしての VPC エンドポイントの設定」を参照してください。
- AWS DMS サーバーレスは、選択ルールと変換ルールを持つビューをサポートしていません。
- AWS DMS serverless は、 AWS カスタマーマネージドキーの使用をサポートしていません。AWS DMS serverless は、デフォルトの DMS キーの使用のみをサポートします。詳細については、「でのデータ保護 AWS Database Migration Service」を参照してください。
- DMS Serverless は DB2 エンドポイントの SSL 接続をサポートしていません。

AWS DMS レプリケーションインスタンスの使用

AWS DMS レプリケーション インスタンスを作成すると、 は Amazon VPC サービスに基づいて Virtual Private Cloud (VPC) の Amazon EC2 インスタンスにそのインスタンス AWS DMS を作成します。このレプリケーション インスタンスを使用して、データベース移行を実行します。[Multi-AZ] (マルチ AZ) オプションを選択した場合、レプリケーション インスタンスはマルチ AZ 配置を使用して高可用性およびフェイルオーバー サポートを提供します。

マルチ AZ 配置では、 はレプリケーションインスタンスの同期スタンバイレプリカ AWS DMS を別のアベイラビリティーゾーンに自動的にプロビジョニングして維持します。プライマリレプリケーション インスタンスは、同期的にアベイラビリティーゾーン間でスタンバイレプリカにレプリケートされます。このアプローチでは、データの冗長性が確保されて I/O のフリーズは排除されるため、レイテンシーのスパイクは最小限に抑えられます。



AWS DMS はレプリケーションインスタンスを使用してソースデータストアに接続し、ソースデータを読み取って、ターゲットデータストアが使用するためにデータをフォーマットします。レプリケーション インスタンスもターゲットデータストアにデータをロードします。この処理のほとんどはメモリ内で行われます。ただし、大きいトランザクションではディスクでのバッファリングが必要になることがあります。キャッシュされたトランザクションとログファイルもディスクに書き込まれます。

レ AWS DMS プリケーション インスタンスは、次の AWS リージョンで作成できます。

リージョン名	リージョン	エンドポイント	プロトコル	
米国東部 (オハイオ)	us-east-2	dms.us-east-2.amazonaws.com	HTTPS	
		dms-fips.us-east-2.amazonaws.com	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
米国東部 (バージニア北部)	us-east-1	dms.us-east-1.amazonaws.com dms-fips.us-east-1.amazonaws.com	HTTPS HTTPS	
米国西部 (北カリフォルニア)	us-west-1	dms.us-west-1.amazonaws.com dms-fips.us-west-1.amazonaws.com	HTTPS HTTPS	
米国西部 (オレゴン)	us-west-2	dms.us-west-2.amazonaws.com dms-fips.us-west-2.amazonaws.com	HTTPS HTTPS	
アフリカ (ケープタウン)	af-south-1	dms.af-south-1.amazonaws.com	HTTPS	
アジアパシフィック (香港)	ap-east-1	dms.ap-east-1.amazonaws.com	HTTPS	
アジアパシフィック (ハイデラバード)	ap-south-2	dms.ap-south-2.amazonaws.com	HTTPS	
アジアパシフィック (ジャカルタ)	ap-southeast-3	dms.ap-southeast-3.amazonaws.com	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
アジアパシフィック (メルボルン)	ap-southeast-4	dms.ap-southeast-4.amazonaws.com	HTTPS	
アジアパシフィック (ムンバイ)	ap-south-1	dms.ap-south-1.amazonaws.com	HTTPS	
アジアパシフィック (大阪)	ap-northeast-3	dms.ap-northeast-3.amazonaws.com	HTTPS	
アジアパシフィック (ソウル)	ap-northeast-2	dms.ap-northeast-2.amazonaws.com	HTTPS	
アジアパシフィック (シンガポール)	ap-southeast-1	dms.ap-southeast-1.amazonaws.com	HTTPS	
アジアパシフィック (シドニー)	ap-southeast-2	dms.ap-southeast-2.amazonaws.com	HTTPS	
アジアパシフィック (東京)	ap-northeast-1	dms.ap-northeast-1.amazonaws.com	HTTPS	
カナダ (中部)	ca-central-1	dms.ca-central-1.amazonaws.com	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
カナダ西部 (カルガリー)	ca-west-1	dms.ca-west-1.amazonaws.com	HTTPS	
欧州 (フランクフルト)	eu-central-1	dms.eu-central-1.amazonaws.com	HTTPS	
欧州 (アイルランド)	eu-west-1	dms.eu-west-1.amazonaws.com	HTTPS	
欧州 (ロンドン)	eu-west-2	dms.eu-west-2.amazonaws.com	HTTPS	
ヨーロッパ (ミラノ)	eu-south-1	dms.eu-south-1.amazonaws.com	HTTPS	
欧州 (パリ)	eu-west-3	dms.eu-west-3.amazonaws.com	HTTPS	
欧州 (スペイン)	eu-south-2	dms.eu-south-2.amazonaws.com	HTTPS	
欧州 (ストックホルム)	eu-north-1	dms.eu-north-1.amazonaws.com	HTTPS	
欧州 (チューリッヒ)	eu-central-2	dms.eu-central-2.amazonaws.com	HTTPS	
イスラエル (テルアビブ)	il-central-1	dms.il-central-1.amazonaws.com	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
中東 (バーレーン)	me-south-1	dms.me-south-1.amazonaws.com	HTTPS	
中東 (アラブ首長国連邦)	me-central-1	dms.me-central-1.amazonaws.com	HTTPS	
南米 (サンパウロ)	sa-east-1	dms.sa-east-1.amazonaws.com	HTTPS	
AWS GovCloud (米国東部)	us-gov-east-1	dms.us-gov-east-1.amazonaws.com	HTTPS	
AWS GovCloud (米国西部)	us-gov-west-1	dms.us-gov-west-1.amazonaws.com	HTTPS	

AWS DMS は、米国政府機関や顧客が機密性の高いワークロードをクラウドに移行できるようにする という特別な AWS リージョンをサポートしています。 AWS GovCloud (US) AWS GovCloud (US) は、米国政府の特定の規制およびコンプライアンス要件に対応しています。の詳細については AWS GovCloud (US)、 GovCloud 「 (米国) とは AWS 」を参照してください。

以下で、レプリケーション インスタンスの詳細についてご参照ください。

トピック

- 移行に適した AWS DMS レプリケーションインスタンスの選択
- レプリケーションインスタンス向けの最適なサイズの選択
- レプリケーションエンジンバージョンの操作
- パブリックおよびプライベートレプリケーション インスタンス
- IP アドレス指定とネットワークタイプ

- レプリケーション インスタンスのためのネットワークのセットアップ
- レプリケーション インスタンスのための暗号化キーの設定
- レプリケーション インスタンスの作成
- レプリケーション インスタンスの変更
- レプリケーション インスタンスを再起動する
- レプリケーション インスタンスを削除する
- AWS DMS メンテナンス ウィンドウの操作

移行に適した AWS DMS レプリケーションインスタンスの選択

AWS DMS は Amazon EC2 インスタンスにレプリケーションインスタンスを作成します。 AWS DMS 現在、 はレプリケーションインスタンスの T2, T3, C4, C5, C6i, R4, R5R6i Amazon EC2 インスタンスクラスをサポートしています。

- T2 インスタンスはベースラインを超えるとバーストする機能を備えたベースラインレベルの CPU パフォーマンスを提供するバースト可能なパフォーマンスインスタンスです。ベースラインパフォーマンスとバースト機能は、CPU クレジットにより管理されます。T2 インスタンスはインスタンスのサイズに応じて、設定レートで CPU クレジットを継続的に受け取ります。これらのインスタンスはアイドル状態では CPU クレジットを蓄積し、アクティブなときは CPU クレジットを消費します。

  T2 インスタンスはさまざまな汎用ワークロードに適しています。たとえば、マイクロサービス、低レイテンシーのインタラクティブなアプリケーション、小中規模のデータベース、仮想化デスクトップ、開発、ビルド、ステージ環境、コードリポジトリ、製品プロトタイプなどがあります。
- T3 インスタンスは次世代バースト可能汎用インスタンスタイプです。このインスタンスタイプはいつでも必要な時間だけ CPU 使用率をバーストさせる機能を備えベースラインレベルの CPU パフォーマンスを提供します。T3 インスタンスはコンピューティング、メモリ、ネットワークリソースを提供し、一時的な使用スパイクが発生する CPU 使用率の中程度のアプリケーション向けに設計されています。T3 インスタンスはワークロードがベースラインしきい値を下回って動作している場合、CPU クレジットを累積します。獲得した CPU クレジットはフル CPU コアパフォーマンスで必要に応じて 1 分間バーストさせる機会を T3 インスタンスに提供します。

  T3 インスタンスは必要に応じて `unlimited` モードでいつでもバーストできます。`unlimited` モードの詳細については、「バーストパフォーマンスインスタンスの Unlimited モードでの作業」をご参照ください。

- C4 インスタンスはコンピューティング集約型ワークロードに最適化されており、コンピューティング比あたり低価格で非常にコスト効率の高いパフォーマンスを実現します。これにより、パケット/秒 (PPS) のパフォーマンスが大幅に向上し、ネットワークジッターが低く、ネットワークレイテンシーも低くなります。 AWS DMS 特に Oracle から PostgreSQL への移行などの異種移行やレプリケーションを実行する場合、CPU を大量に消費する可能性があります。C4 インスタンスは、このような状況に適しています。
- C5 インスタンスはアドバンスト コンピューティング集約型ワークロードを実行するためにコンピューティング比あたりの低コストでコスト効率の高いパフォーマンスを実現する次世代インスタンスタイプです。これには、ハイパフォーマンスウェブサーバー、ハイパフォーマンスコンピューティング (HPC)、バッチ処理、広告配信、高スケーラビリティのあるマルチプレイヤー ゲーム、ビデオ エンコーディングなどのワークロードが含まれます。その他のワークロード C5 インスタンスは科学的モデリング、分散分析、マシンおよび深層学習の推論などに適しています。C5 インスタンスは Intel および AMD のプロセッサーの選択で利用できます。
- C6i インスタンスは、さまざまなワークロードに対し、同等の Gen5 インスタンスと比較して最大 15 % 優れたコンピューティングコストパフォーマンスと常時オンのメモリ暗号化を提供します。C6i インスタンスは、バッチ処理、分散分析、ハイパフォーマンスコンピューティング (HPC)、広告配信、高スケーラビリティのマルチプレイヤーゲーム、動画エンコーディングなど、コンピューティング集約型のワークロードに最適です。
- R4 インスタンスはメモリ負荷の大きいワークロード用に最適化されたメモリです。 AWS DMS を使用する高スループット トランザクション システムの継続的な移行やレプリケーションは大量の CPU やメモリ消費につながることがあります。R4 インスタンスは旧世代のインスタンスタイプよりも多くの vCPU あたりのメモリが含まれます。
- R5 インスタンスは次世代のメモリ最適化 Amazon EC2 インスタンスタイプです。R5 インスタンスはハイパフォーマンス データベース、ウェブ スケール分散インメモリキャッシュ、中規模のインメモリデータベース、リアルタイム ビッグデータ分析、その他のエンタープライズアプリケーションなど、メモリを大量に消費するアプリケーションに最適です。を使用した高スループット トランザクションシステムの継続的な移行やレプリケーションでも、大量の CPU とメモリが消費 AWS DMS される可能性があります。
- R6i インスタンスは、さまざまなワークロードに対し、同等の Gen5 インスタンスと比較して最大 15 % 優れたコンピューティングコストパフォーマンスと常時オンのメモリ暗号化を提供します。R6i インスタンスは SAP 認定を受けており、SQL データベースや noSQL データベースなどのワークロード、Memcached や Redis などの分散ウェブスケールインメモリキャッシュ、SAP HANA などのインメモリデータベース、Hadoop クラスターや Spark クラスターなどのリアルタイムビッグデータ分析に最適です。

レプリケーション インスタンスごとに、メモリと vCPU の固有の設定があります。次の表は、各レプリケーション インスタンスタイプの設定を示しています。料金の詳細については「AWS Database Migration Service 料金表のページ」をご参照ください。

汎用レプリケーションインスタンスタイプ

タイプ	vCPU	メモリ (GiB)
dms.t2.micro	1	1
dms.t2.small	1	2
dms.t2.medium	2	4
dms.t2.large	2	8
dms.t3.micro	2	1
dms.t3.small	2	2
dms.t3.medium	2	4
dms.t3.large	2	8

コンピューティング最適化レプリケーションインスタンスタイプ

タイプ	vCPU	メモリ (GiB)
dms.c4.large	2	3.75
dms.c4.xlarge	4	7.5
dms.c4.2xlarge	8	15
dms.c4.4xlarge	16	30
dms.c5.large	2	4
dms.c5.xlarge	4	8

タイプ	vCPU	メモリ (GiB)
dms.c5.2xlarge	8	16
dms.c5.4xlarge	16	32
dms.c5.9xlarge	36	72
dms.c5.12xlarge	48	96
dms.c5.18xlarge	72	144
dms.c5.24xlarge	96	192
dms.c6i.large	2	4
dms.c6i.xlarge	4	8
dms.c6i.2xlarge	8	16
dms.c6i.4xlarge	16	32
dms.c6i.8xlarge	32	64
dms.c6i.12xlarge	48	96
dms.c6i.16xlarge	64	128
dms.c6i.24xlarge	96	192
dms.c6i.32xlarge	128	256

メモリ最適化レプリケーションインスタンスタイプ

タイプ	vCPU	メモリ (GiB)
dms.r4.large	2	15.25
dms.r4.xlarge	4	30.5

タイプ	vCPU	メモリ (GiB)
dms.r4.2xlarge	8	61
dms.r4.4xlarge	16	122
dms.r4.8xlarge	32	244
dms.r5.large	2	16
dms.r5.xlarge	4	32
dms.r5.2xlarge	8	64
dms.r5.4xlarge	16	128
dms.r5.8xlarge	32	256
dms.r5.12xlarge	48	384
dms.r5.16xlarge	64	512
dms.r5.24xlarge	96	768
dms.r6i.large	2	16
dms.r6i.xlarge	4	32
dms.r6i.2xlarge	8	64
dms.r6i.4xlarge	16	128
dms.r6i.8xlarge	32	256
dms.r6i.12xlarge	48	384
dms.r6i.16xlarge	64	512
dms.r6i.24xlarge	96	768
dms.r6i.32xlarge	128	1024

上記の表はすべての AWS DMS レプリケーションインスタンスタイプを一覧表示していますが、リージョンで使用できるタイプは異なる場合があります。各リージョンで利用できるレプリケーションインスタンスタイプを確認するには、次の AWS CLI コマンドを実行します。

```
aws dms describe-orderable-replication-instances --region your_region_name
```

トピック

- 使用するインスタンスクラスの決定
- バーストパフォーマンスインスタンスの Unlimited モードでの作業

使用するインスタンスクラスの決定

どのレプリケーションインスタンスクラスが最適かを判断するために、 が AWS DMS 使用する変更データキャプチャ (CDC) プロセスを見てみましょう。

全ロード + CDC タスク (一括ロードと継続的なレプリケーション) を実行していると想定します。この場合、タスクにはメタデータやその他の情報を格納する独自の SQLite リポジトリがあります。が全ロード AWS DMS を開始する前に、以下の手順を実行します。

- AWS DMS は、ソースエンジンのトランザクションログから移行するテーブルの変更のキャプチャを開始します (これらのキャッシュされた変更を と呼びます) 。全ロードが完了すると、これらのキャッシュされた変更が収集され、ターゲットに適用されます。キャッシュされた変更のボリュームに応じて、これらの変更は、メモリから直接適用できます。ここで、これらの変更から設定しきい値まで収集されます。または、ディスクから適用して、変更がメモリに保持できないときに書き込まれるようにすることもできます。
- キャッシュされた変更が適用されると、デフォルトではターゲットインスタンスでトランザクション適用プロセス AWS DMS が開始されます。

適用されたキャッシュされた変更フェーズと継続的なレプリケーションフェーズでは、 は 2 つのストリームバッファ AWS DMS を使用します。1 つは受信データと送信データ用です。 は、別のメモリバッファであるソートと呼ばれる重要なコンポーネント AWS DMS も使用します。ソーターコンポーネントの 2 つの重要な用途 (他のコンポーネントを含む) は次のとおりです。

- すべてのトランザクションを追跡し、関連するトランザクションのみを送信バッファに転送します。
- これにより、トランザクションはソース上と同じコミットの順番で転送されます。

見てわかるように、このアーキテクチャには、 AWS DMSの CDC 用の 3 つの重要なメモリバッファがあります。これらのバッファのいずれでメモリ負荷が生じた場合、移行でパフォーマンス上の問題が発生し、障害が起きる可能性があります。

1 秒あたりのトランザクション数 (TPS) が多い重いワークロードをこのアーキテクチャに接続すると、R5 インスタンス と R6i インスタンスが提供する追加メモリが役立つことが明らかになります。R5 インスタンスと R6i インスタンスを使用すると、メモリ内に多数のトランザクションが保持されるため、継続的なレプリケーション中のメモリ負荷の問題を回避できます。

バーストパフォーマンスインスタンスの Unlimited モードでの作業

T3 インスタンスなどの unlimited として設定したバーストパフォーマンスインスタンスは、必要に応じた期間にわたり、高い CPU 使用率を保持できます。時間あたりインスタンス料金は、すべての CPU 使用率のスパイクを自動的にカバーできます。24 時間移動ベースでまたはインスタンスの存続期間のいずれか短い方の時間で、インスタンスの平均 CPU 使用率がベースライン以下になった場合、インスタンス料金は自動的にすべての CPU 使用率スパイクをカバーします。

汎用のワークロードではほとんどの場合、unlimited として設定されたインスタンスは追加料金なしで十分なパフォーマンスを提供します。長時間にわたって高い CPU 使用率でインスタンスを実行する場合には、vCPU 時間ごとに均一追加料金が発生します。T3 インスタンスの料金について詳しくは、AWS Database Migration Service の「T3 CPU クレジット」をご参照ください。

T3 インスタンスの unlimited モードの詳細については、「Amazon EC2 ユーザーガイド」の「バーストパフォーマンスインスタンスの無制限モード」を参照してください。 Amazon EC2

> **Important**
>
> AWS 無料利用枠提供の下でdms.t3.microインスタンスをunlimitedモードで使うと、料金が適用される場合があります。特に、24 時間横揺れ周期における平均使用率が、そのインスタンスのベースライン使用率を超過すると料金が発生することがあります。詳細については、「Amazon EC2 ユーザーガイド」の「ベースライン使用率」を参照してください。
> Amazon EC2
> T3 インスタンスは、デフォルトで unlimited として起動します。24 時間の平均 CPU 使用率がベースラインを超えた場合は、余剰クレジットに対して課金されます。場合によっては、T3 スポットインスタンスは unlimited のように起動することがあり、すぐに短時間使用する予定もあると思います。CPU クレジットが発生するアイドル時間なしで実行すると、余剰クレジットの料金が発生します。コストの増加を抑えるには、T3 スポットインスタンスを標準モードで起動することをお勧めします。詳細については、「Amazon EC2 ユーザーガ

イド」の「余剰クレジットには、バーストパフォーマンスインスタンスの料金 、T3 スポットインスタンス 、および標準モードが発生する可能性がありますT3」を参照してください。https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/burstable-performance-instances-standard-mode.html Amazon EC2

レプリケーションインスタンス向けの最適なサイズの選択

適切なレプリケーション インスタンスの選択は、ユースケースのいくつかの要因によって異なります。レプリケーション インスタンスリソースの使用方法を理解しやすいように、次の説明をご参照ください。全ロードと CDC タスクの一般的なシナリオを説明します。

フルロードタスク中、 はテーブルを個別に AWS DMS ロードします。デフォルトでは、一度に 8 つのテーブルがロードされます。 は、フルロードタスク中にソースへの継続的な変更 AWS DMS をキャプチャするため、変更は後でターゲットエンドポイントに適用できます。変更はメモリにキャッシュされます。使用可能なメモリがなくなると、変更はディスクにキャッシュされます。テーブルのフルロードタスクが完了すると、 はキャッシュされた変更をターゲットテーブルに AWS DMS 直ちに適用します。

テーブルのキャッシュされた変更がすべて適用されると、ターゲットエンドポイントはトランザクション上一貫性のある状態になります。この時点で、ターゲットは最後にキャッシュされた変更に関してソースエンドポイントと同期しています。 AWS DMS その後、 はソースとターゲットの間で継続的なレプリケーションを開始します。そのために、 はソーストランザクションログから変更オペレーション AWS DMS を受け取り、トランザクション整合性のある方法でターゲットに適用します。（このプロセスでは、バッチ最適化適用が選択されていないことを前提としています）。 は、可能であれば、レプリケーションインスタンスのメモリを介して進行中の変更を AWS DMS ストリーミングします。それ以外の場合、 はターゲットに適用できるようになるまで、レプリケーション インスタンスのディスクに変更を AWS DMS 書き込みます。

レプリケーション インスタンスが変更処理をどのように処理するか、およびそのプロセスでメモリがどのように使用されるかについて、何らかの制御があります。変更処理のチューニング方法の詳細については、「変更処理のチューニング設定」をご参照ください。

考慮すべき要素

メモリとディスク容量は、ユースケースに適したレプリケーションインスタンスを選択する際の重要な要素です。レプリケーションインスタンスを選択するために分析すべきユースケースの特徴について次に説明します。

- データベースとテーブルのサイズ

  データ量は、フルロードパフォーマンスを最適化するためのタスクの設定を決定するうえで役立ちます。例えば、1 TB スキーマが 2 つある場合、テーブルをパーティション化して 500 GB の 4 つのタスクの分割して、それらを並列で実行できます。レプリケーションインスタンスで使用できる CPU リソースにより、実行できる並列処理は異なります。このため、フルロードパフォーマンスを最適化するには、データベースのサイズとテーブルのサイズを把握することをお勧めします。これにより、実行できるタスクの数を判断できます。

- ラージオブジェクト

  移行の範囲内のデータ型がパフォーマンスに影響を及ぼす場合があります。特にラージオブジェクト (LOB) は、パフォーマンスとメモリ消費に影響します。LOB 値を移行するには、 は 2 ステップのプロセス AWS DMS を実行します。まず、 AWS DMS は LOB 値なしで行をターゲットに挿入します。次に、LOB 値で行 AWS DMS を更新します。このプロセスはメモリに影響を及ぼすため、ソース内の LOB 列を特定し、そのサイズを分析しておくことが重要です。

- ロード頻度とトランザクションのサイズ

  ロード頻度と 1 秒あたりのトランザクション数 (TPS) は、メモリ使用量に影響します。TPS またはデータ操作言語 (DML) のアクティビティ数が多いと、メモリの使用量が上がります。これが発生するのは、変更内容をターゲットに適用するまで DMS はキャッシュするためです。これにより、CDC の実行中にスワッピング (メモリオーバーフローによる物理ディスクへの書き込み) が発生し、レイテンシーが発生します。

- テーブルキーと参照整合性

  テーブルのキーに関する情報により、データの移行に使用する CDC モード (バッチ適用またはトランザクション適用) が定まります。通常、トランザクションの適用はバッチの適用よりも遅くなります。長時間実行されるトランザクションの場合、多くの変更を移行する必要がある可能性があります。トランザクション適用を使用する場合、バッチ適用と比較して変更を保存するためにより多くのメモリが必要になる AWS DMS 場合があります。プライマリキーなしでテーブルを移行すると、一括適用が失敗し、DMS タスクはトランザクション適用モードに変更されます。CDC 中にテーブル間で参照整合性がアクティブな場合、 はデフォルトでトランザクション適用 AWS DMS を使用します。バッチ適用とトランザクション適用との比較の詳細については、「DMS バッチ適用機能を使用して CDC レプリケーションパフォーマンスを向上させるにはどうすればよいですか?」を参照してください。

このようなメトリクスを使用して、レプリケーションインスタンスをコンピューティング最適化またはメモリ最適化する必要があるかを判断します。

一般的な問題

移行中にレプリケーションインスタンスでのリソースの競合の原因となる次のとおりの一般的な問題に直面する場合があります。レプリケーションインスタンスメトリクスの詳細については、「レプリケーションインスタンスのメトリクス」を参照してください。

- レプリケーションインスタンスのメモリが不足すると、データはディスクに書き込まれます。ディスクからの読み取りによってレイテンシーが発生する可能性があります。ただしこれは、十分なメモリを備えたレプリケーションインスタンスのサイズを設定することで回避できます。
- レプリケーションインスタンスに割り当てられるディスクサイズは、必要となるより小さいサイズでも構いません。ディスクサイズは、メモリ内のデータがオーバーフローした場合や、タスクログの保存にも使用されます。最大 IOPS もこれに依存します。
- 複数のタスクや並列処理の高いタスクを実行すると、レプリケーションインスタンスの CPU 消費量に影響します。これにより、タスクの処理が遅くなり、レイテンシーが発生します。

ベストプラクティス

レプリケーションインスタンスのサイジングを行う際は、次の 2 つの最も一般的なベストプラクティスの採用を検討します。詳細については、「AWS Database Migration Service のベストプラクティス」を参照してください。

1. ワークロードのサイジングを行い、コンピューティング集約型であるか、メモリ集中型であるかを判断しておきます。これに基づいて、レプリケーションインスタンスのクラスとサイズを決定できます。
   - AWS DMS はメモリ内の LOBsを処理します。この操作にはかなりの量のメモリが必要です。
   - タスクの数とスレッドの数は CPU 消費に影響します。フルロード操作中は `MaxFullLoadSubTasks` の使用を最大 8 に抑えます。
2. フルロード中のワークロードが高い場合は、レプリケーションインスタンスに割り当てられるディスク容量を増やします。これにより、レプリケーションインスタンスは割り当てられた IOPS を最大限に利用できます。

上記のガイドラインは考えられるすべてのシナリオをカバーしているわけではありません。レプリケーションインスタンスのサイジングを行う際は、特定のユースケースの詳細を考慮することが重要です。

上記のテストにより、CPU とメモリはワークロードによって異なることが明らかになっています。具体的には、LOB はメモリに影響し、タスク数や並列処理は CPU に影響します。移行の実行後、レプリケーションインスタンスのCPU、解放可能なメモリ、空きストレージ、IOPS をモニタリングします。収集したデータに基づいて、必要に応じてレプリケーション インスタンスのサイズを増減することができます。

レプリケーションエンジンバージョンの操作

レプリケーションエンジンは、レプリケーションインスタンスで実行され、指定した移行タスクを実行するコア AWS DMS ソフトウェアです。 AWS は、新しい機能とパフォーマンスの向上により、 AWS DMS レプリケーションエンジンソフトウェアの新しいバージョンを定期的にリリースします。レプリケーションエンジンソフトウェアの各バージョンには、他のバージョンと区別するための独自のバージョン番号があります。

新しいレプリケーションインスタンスを起動すると、特に指定しない限り、最新の AWS DMS エンジンバージョンが実行されます。詳細については、「AWS DMS レプリケーションインスタンスの使用」を参照してください。

現在実行中のレプリケーションインスタンスがある場合は、より新しいエンジンバージョンにアップグレードできます (エンジンバージョンのダウングレードAWS DMS はサポートされていません)。レプリケーションエンジンのバージョンの詳細については、「AWS DMS リリースノート」をご参照ください。

コンソールを使用したエンジンバージョンのアップグレード

を使用してレ AWS DMS プリケーション インスタンスをアップグレードできます AWS Management Console。

コンソールを使用してレプリケーション インスタンスをアップグレードするには

1. https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. ナビゲーションペインで [Replication instances] (レプリケーション インスタンス) を選択します。
3. レプリケーションエンジンを選択し、[Modify] (変更) を選択します。
4. [エンジンバージョン] で アップグレードするバージョン番号を選択して、[変更する] をクリックします。

> **Note**
>
> レプリケーション インスタンスをアップグレードする前に、すべてのタスクを停止することをお勧めします。タスクを停止しない場合、 AWS DMS はアップグレード前にタスクを自動的に停止します。タスクを手動で停止した場合、アップグレードの完了後にタスクを手動で開始する必要があります。レプリケーション インスタンスタイプのアップグレードには数分間かかります。インスタンスが使用可能になると、ステータスは available に変わります。

を使用したエンジンバージョンのアップグレード AWS CLI

レ AWS DMS プリケーション インスタンスは AWS CLI、次のように を使用してアップグレードできます。

を使用してレプリケーション インスタンスをアップグレードするには AWS CLI

1. 次のコマンドを使用して、レプリケーション インスタンスの Amazon リソースネーム (ARN) を確認します。

   ```
   aws dms describe-replication-instances \
   --query "ReplicationInstances[*].
   [ReplicationInstanceIdentifier,ReplicationInstanceArn,ReplicationInstanceClass]"
   ```

   出力で、アップグレードするレプリケーション インスタンスの ARN を書き留めます。たとえば、`arn:aws:dms:us-east-1:123456789012:rep:6EFQQO6U6EDPRCPKLNPL2SCEEY` などです。
2. 次のコマンドを使用して、使用可能なレプリケーション インスタンスバージョンを調べます。

   ```
   aws dms describe-orderable-replication-instances \
   --query "OrderableReplicationInstances[*].[ReplicationInstanceClass,EngineVersion]"
   ```

   出力で、エンジンバージョン番号またはレプリケーション インスタンスクラスに使用できる番号を書き留めます。この情報はステップ 1 の出力に表示されます。
3. 次のコマンドを使ってレプリケーション インスタンスをアップグレードします。

   ```
   aws dms modify-replication-instance \
   --replication-instance-arn arn \
   --engine-version n.n.n
   ```

前述の *arn* を、前のステップで書き留めた実際のレプリケーション インスタンス ARN に置き換えます。

n.n.n を、たとえば、3.4.5 など目的のエンジンバージョン番号に置き換えます。

> **Note**
>
> レプリケーション インスタンスタイプのアップグレードには数分間かかります。次のコマンドを使用して、レプリケーション インスタンスのステータスを表示できます。
>
> ```
> aws dms describe-replication-instances \
> --query "ReplicationInstances[*].
> [ReplicationInstanceIdentifier,ReplicationInstanceStatus]"
> ```
>
> レプリケーション インスタンスが使用可能になると、ステータスは available に変わります。

パブリックおよびプライベートレプリケーション インスタンス

レプリケーションインスタンスがソースデータベースとターゲットデータベースに接続するために使用するについて、パブリック IP アドレスにするかプライベート IP アドレスするかを指定できます。

プライベート レプリケーション インスタンスには、レプリケーション ネットワーク外からアクセスできないるプライベート IP アドレスがあります。ソースデータベースとターゲットデータベースの両方が、レプリケーションインスタンスの仮想プライベートクラウド (VPC) に接続されている同じネットワーク内にある場合は、専用インスタンスを使用します。ネットワークは、仮想プライベートネットワーク (VPN) AWS Direct Connect、または VPC ピアリングを使用して VPC に接続できます。

VPC ピアリング接続は、2 つの VPC 間のネットワーク接続です。これを使用すると、各 VPC のプライベート IP アドレスを使用して、あたかも同じネットワーク内にあるかのようにルーティングできます。VPC ピアリングの詳細については、Amazon VPC ユーザーガイド の「VPC ピアリング」をご参照ください。

パブリックレプリケーションインスタンスは、レプリケーションインスタンスの VPC セキュリティグループ、レプリケーションインスタンスのパブリック IP アドレス、または NAT ゲートウェイの

パブリック IP アドレスを使用できます。これらの接続によって、データ移行に使用するネットワークが形成されます。

IP アドレス指定とネットワークタイプ

AWS DMS は常に Amazon Virtual Private Cloud (VPC) にレプリケーションインスタンスを作成します。VPC を作成する際に、使用する IP アドレス指定 (IPv4 または IPv6、あるいはその両方) を決定できます。その後、レプリケーションインスタンスを作成または変更する際に、デュアルスタックモードを使用して IPv4 アドレスプロトコルまたは IPv6 アドレスプロトコルの使用を指定できます。

IPv4 アドレス

VPC を作成する際、10.0.0.0/16 などのクラスレス ドメイン間ルーティング (CIDR) ブロックの形式で VPC の IPv4 アドレスの範囲を指定できます。サブネットグループは、この CIDR ブロック内の IP アドレスの範囲を定義します。これらの IP アドレスはプライベートまたはパブリックです。

プライベート IPv4 アドレスは、インターネットから到達できない IP アドレスです。プライベート IPv4 アドレスは、レプリケーションインスタンスと同じ VPC 内のその他のリソース(Amazon EC2 インスタンスなど) との間の通信に使用できます。各レプリケーションインスタンスには、VPC 内通信のためのプライベート IP アドレスが備えられています。

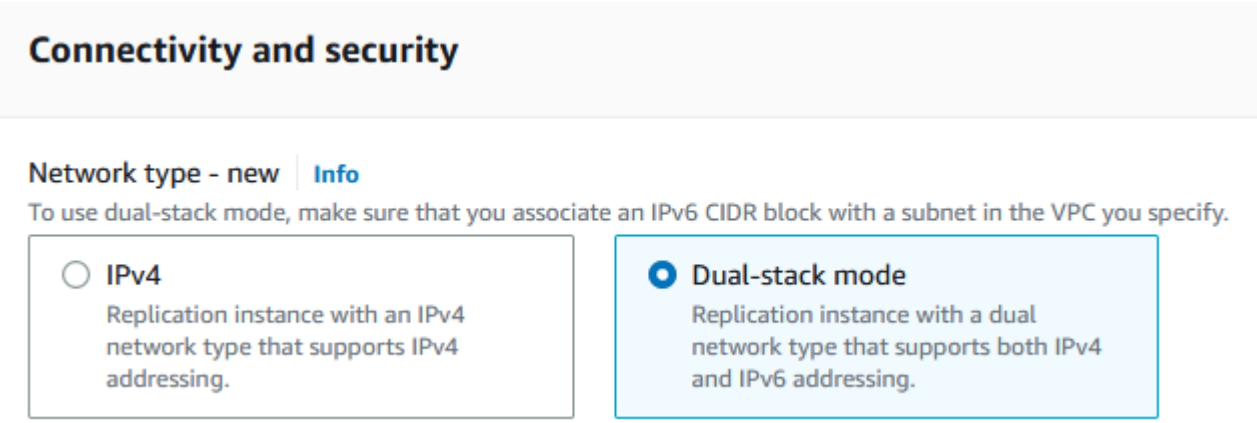
パブリック IP アドレスは、インターネットから到達可能な IPv4 アドレスです。パブリックアドレスは、レプリケーションインスタンスとインターネット上のリソース間の通信に使用できます。レプリケーションインスタンスにパブリック IP アドレスに持たせるかをユーザーは管理できます。

デュアルスタックモードと IPv6 アドレス

IPv6 を介してレプリケーションインスタンスと通信する必要があるリソースがある場合は、デュアルスタックモードを使用します。デュアルスタックモードを使用するには、レプリケーションインスタンスに関連付けるサブネットグループ内の各サブネットに IPv6 CIDR ブロックが関連付けられていることを確認します。この要件を満たすように、新しいレプリケーションサブネットグループを作成するか、既存のレプリケーションサブネットグループを変更することができます。各 IPv6 アドレスは、グローバルに一意です。VPC の IPv6 CIDR ブロックは、Amazon の IPv6 アドレスのプールから自動的に割り当てられます。範囲を自分で選択することはできません。

DMS は、プライベートデュアルスタックモードのレプリケーションインスタンスの IPv6 エンドポイントに対するインターネットゲートウェイアクセスを無効にします。DMS がこれを行うのは、IPv6 エンドポイントをプライベートにして、VPC 内からのみアクセスできるようにするためです。

AWS DMS コンソールを使用してレプリケーションインスタンスを作成または変更し、ネットワークタイプセクションでデュアルスタックモードを指定できます。次の画像は、コンソールの [Network type] (ネットワークの種類) セクションを示しています。



リファレンス

- IPv4 と IPv6 の詳細については、「Amazon VPC ユーザーガイド」の「Amazon VPC の仕組み」を参照してください。
- デュアルスタックモードを使用したレプリケーションインスタンスの作成の詳細については、「レプリケーション インスタンスの作成」を参照してください。
- レプリケーションインスタンスの変更の詳細については、「 レプリケーション インスタンスの変更」を参照してください。

レプリケーション インスタンスのためのネットワークのセットアップ

AWS DMS は常に Amazon VPC に基づいて VPC にレプリケーションインスタンスを作成します。レプリケーション インスタンスがある VPC を指定します。アカウントと AWS リージョンにデフォルトの VPC を使用するか、新しい VPC を作成できます。

レプリケーション インスタンスの VPC に割り当てられた Elastic Network Interface がセキュリティグループに関連付けられていることを確認します。また、このセキュリティグループのルールで、すべてのポートですべてのトラフィックが VPC から出る (egress) であることを確認します。このアプローチでは、エンドポイントで適切な受信ルールが有効になっている場合、レプリケーションインスタンスからソースデータベースエンドポイントとターゲットデータベースエンドポイントへの通信が許可されます。すべてのアドレスへの送信をすべてのポートで許可するデフォルト設定をエンドポイントに使用することをお勧めします。

ソースおよびターゲットエンドポイントは、VPC に接続するか、VPC 内に配置されることにより、VPC 内にあるレプリケーション インスタンスにアクセスします。データベースエンドポイントには、レプリケーション インスタンスからの受信アクセスを許可するネットワークアクセスコントロールリスト (ACL) およびセキュリティグループルール (該当する場合) を含める必要があります。この設定方法は使用するネットワーク構成によって異なります。レプリケーション インスタンスの VPC セキュリティグループ、レプリケーション インスタンスのプライベートまたはパブリック IP アドレス、あるいは NAT ゲートウェイのパブリック IP アドレスを使用することができます。これらの接続によって、データ移行に使用するネットワークが形成されます。

> **Note**
>
> 基盤となるインフラストラクチャの変更により IP アドレスが変更される可能性があるため、VPC CIDR 範囲を使用するか、NAT GW に関連付けられた 伸縮性 IP を介してレプリケーション インスタンスのアウトバウンド トラフィックをルーティングすることをお勧めします。CIDR ブロックを含む VPC の作成方法の詳細については、VPC とサブネットの使用の「Amazon Virtual Private Cloud ユーザーガイド」をご参照ください。Elastic IP アドレスの詳細については、Amazon Elastic Compute Cloud ユーザーガイドの「Elastic IP アドレス」をご参照ください。

データベース移行のネットワーク設定

AWS Database Migration Service では、複数の異なるネットワーク設定を使用できます。データベース移行に使用されるネットワークの一般的な設定を以下に示します。

トピック

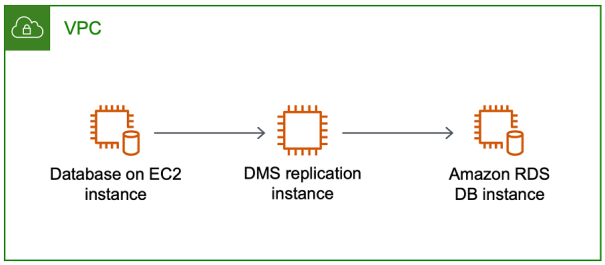
- すべてのデータベース移行コンポーネントが 1 つの VPC にある設定
- 複数の VPC を使用する構成
- 共有 VPC を使用した構成
- AWS Direct Connect または VPN を使用した VPC へのネットワークの設定
- インターネットを使用した VPC へのネットワークの設定
- を使用した VPC 外の RDS DB インスタンスから VPC 内の DB インスタンスへの設定 ClassicLink

現実的であれば、ターゲットエンドポイントと同じリージョン、ターゲットエンドポイントと同じ VPC またはサブネットに DMS レプリケーションインスタンスを作成することをお勧めします。

すべてのデータベース移行コンポーネントが 1 つの VPC にある設定

ソースエンドポイント、レプリケーション インスタンス、ターゲットエンドポイントがすべて同じ VPC にある場合、データベース移行のネットワークが最もシンプルになります。この設定は、ソースとターゲットのエンドポイントが Amazon RDS DB インスタンスまたは Amazon EC2 インスタンスにある場合に適しています。

次の図は、Amazon EC2 インスタンス上のデータベースがレプリケーション インスタンスに接続され、データが Amazon RDS DB インスタンスに移行される設定を示しています。



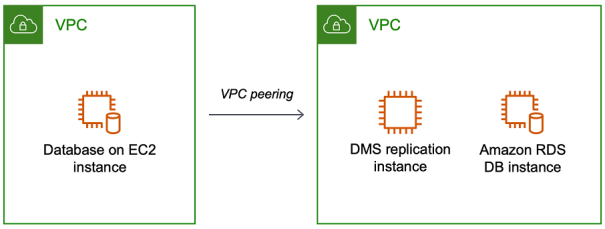
この設定で使用される VPC セキュリティグループは、レプリケーション インスタンスからの受信をデータベースポートで許可する必要があります。これにはいくつか方法があります。レプリケーション インスタンスによって使用されるセキュリティグループにエンドポイントへの入力があることを確認できます。または、レプリケーションインスタンスの VPC CIDR 範囲、NAT ゲートウェイ Elastic IP、またはプライベート IP アドレス (使用している場合) を許可することもできます。ただし、レプリケーション IP アドレスが変更されるとレプリケーションが中断される可能性があるため、レプリケーションインスタンスのプライベート IP アドレスを使用することはお勧めしません。

複数の VPC を使用する構成

ソース エンドポイントとターゲット エンドポイントが別の VPC にある場合、いずれかの VPC でレプリケーション インスタンスを作成できます。これで、VPC ピアリングを使用して 2 つの VPC をリンクできます。

VPC ピア接続は、同じネットワーク内にあるかのように各 VPC のプライベート IP アドレスを使用してルーティングできるようにする、2 つの VPC 間のネットワーキング接続です。独自の VPC 間、別の AWS アカウントの VPCs間、または別の AWS リージョンの VPC との間で VPC ピアリング接続を作成できます。VPC ピアリングの詳細については、Amazon VPC ユーザーガイド の「VPC ピアリング」をご参照ください。

次の図に、VPC ピア接続の設定例を示します。ここでは、VPC 内の Amazon EC2 インスタンスのソースデータベースが VPC ピア接続で VPC と接続されます。この VPC には、レプリケーション インスタンスと Amazon RDS DB インスタンス上のターゲットデータベースが含まれています。



VPC ピアリング接続を実装するには、「Amazon VPC」ドキュメントの「Amazon Virtual Private Cloud の VPC ピア接続を操作する」を参照してください。片方の VPC のルートテーブルにもう一方の VPC の CIDR ブロックが含まれていることを確認します。例えば、VPC A が宛先 10.0.0.0/16 を使用して、VPC B が宛先 172.31.0.0 を使用している場合、VPC A のルートテーブルには 172.31.0.0 が含まれ、VPC B のルートテーブルには 10.0.0.0/16 が含まれている必要があります。詳細については、「Amazon VPC ピアリング接続」ドキュメントの「VPC ピアリング接続のルートテーブルを更新する」を参照してください。

この構成で使用される VPC セキュリティグループは、レプリケーションインスタンスからのデータベースポートでの受信を許可するか、ピアリングされている VPC の CIDR ブロックでの受信を許可する必要があります。

共有 VPC を使用した構成

AWS DMS は、組織内の参加している顧客アカウントと共有されているサブネットを、同じアカウントの通常のサブネットと同様に扱います。以下は、 が VPCs AWS DMS を処理する方法、サブネット、および共有 VPCs。

`ReplicationSubnetGroup` オブジェクトを作成すると、カスタムサブネットまたは VPC で動作するようにネットワークを構成できます。`ReplicationSubnetGroup` を作成する場合、アカウント内の特定の VPC のサブネットを指定できます。指定するサブネットのリストには、別のアベイラビリティーゾーンにある少なくとも 2 つのサブネットが含まれている必要があり、すべてのサブネットが同じ VPC 内にある必要があります。を作成する場合`ReplicationSubnetGroup`、各サブネットは 1 つの AWS DMS VPC にのみリンクされるため、お客様はユーザーに代わって subnets のみを指定します。

または を作成する AWS DMS `ReplicationInstance`ときに AWS DMS `ReplicationConfig`、または `ReplicationInstance` Serverless Replication が動作する `ReplicationSubnetGroup`および/または VPC セキュリティグループを指定できます。指定しない場合、お客様のデフォルト `ReplicationSubnetGroup` (デフォルト VPC 内のすべてのサブネットに指定されていない場合はがユーザーに代わって AWS DMS 作成する) とデフォルトの VPC セキュリティグループ AWS DMS を選択します。

移行は、お客様が指定したアベイラビリティーゾーン、または ReplicationSubnetGroup 内の任意のアベイラビリティーゾーンで実行できます。がレプリケーションインスタンスの作成またはサーバーレスレプリケーションの開始 AWS DMS を試みると、サブネットのアベイラビリティーゾーンがコアサービスアカウントのアベイラビリティーゾーンに変換され、アベイラビリティーゾーンのマッピングが 2 つのアカウント間で同一でなくても、正しいアベイラビリティーゾーンでインスタンスを起動できるようになります。

共有 VPC を使用する場合は、共有 VPC から使用するサブネットにマップする ReplicationSubnetGroup オブジェクトを必ず作成する必要があります。ReplicationInstance または ReplicationConfig を作成する際は、共有 VPC の ReplicationSubnetGroup を指定し、Create リクエストで共有 VPC のために作成した VPC セキュリティグループを指定する必要があります。

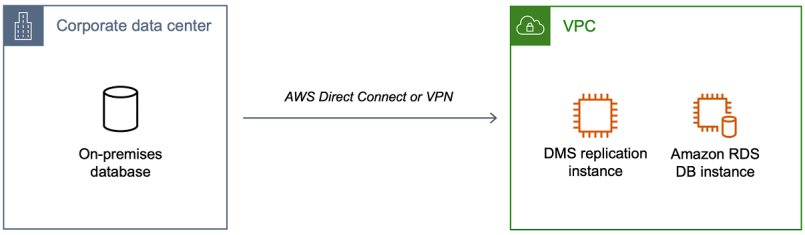
共有 VPC の使用に際して、次の点に注意する必要があります。

- VPC 所有者は、参加者とリソースを共有できませんが、参加者は所有者のサブネットにサービスリソースを作成できます。
- すべてのリソースはアカウント固有であるため、VPC 所有者は参加者が作成したリソース (レプリケーションインスタンスなど) にアクセスできません。ただし、レプリケーションインスタンスを共有 VPC に作成する限り、レプリケーションエンドポイントまたはタスクに適切な権限が付与されていれば、所有アカウントを問わず VPC 内のリソースにアクセスできます。
- リソースはアカウント固有であるため、他の参加者はその他のアカウントが所有するリソースにはアクセスできません。自分のアカウントで共有 VPC に作成されたリソースにアクセスできるように他のアカウントに付与できるようなアクセス許可はありません。

AWS Direct Connect または VPN を使用した VPC へのネットワークの設定

リモートネットワークは、 AWS Direct Connect やソフトウェアまたはハードウェア VPN 接続などのいくつかのオプションを使用して VPC に接続できます。これらのオプションは、内部ネットワークを AWS クラウドに拡張することでモニタリング、認証、セキュリティ、データ、他のシステムなどの既存のオンサイトサービスを統合するためによく使われます。このタイプのネットワーク拡張を使用することで、VPC などの AWSにホストされたリソースにシームレスに接続できます。

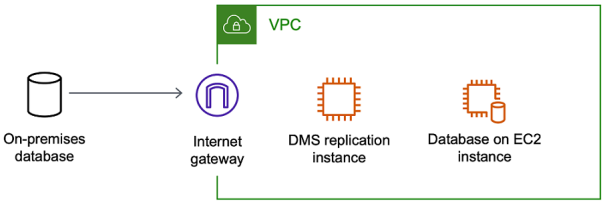
次の図は、ソースエンドポイントが企業データセンターにあるオンプレミスデータベースである設定を示しています。このデータベースは AWS Direct Connect または VPN を使用することで、レプリケーション インスタンスと、Amazon RDS DB インスタンス上のターゲット データベースを含む VPC と接続されます。



この設定では VPC セキュリティグループに、VPC の CIDR 範囲または特定の IP アドレスを送信先とするトラフィックをホストに送信するルーティングルールが必要です。このホストは、VPC からのトラフィックをオンプレミスの VPN にブリッジできる必要があります。この場合、NAT ホストは独自のセキュリティグループ設定を含みます。このような設定の場合、レプリケーションインスタンスの VPC の CIDR 範囲、プライベート IP アドレス、またはセキュリティグループから NAT インスタンスへのトラフィックを許可する必要があります。ただし、レプリケーション IP アドレスが変更されるとレプリケーションが中断される可能性があるため、レプリケーションインスタンスのプライベート IP アドレスを使用することはお勧めしません。

インターネットを使用した VPC へのネットワークの設定

VPN または を使用して AWS リソース AWS Direct Connect に接続しない場合は、インターネットを使用してデータベースを移行できます。この場合 Amazon EC2 インスタンスまたは Amazon RDS DB インスタンスのいずれかに移行できます。この設定では、ターゲットポイントおよびレプリケーション インスタンスを含むインターネットゲートウェイを持つ VPC 内にパブリックレプリケーション インスタンスが必要です。



インターネットゲートウェイを VPC に追加するには、Amazon VPC User Guide の「インターネットゲートウェイのアタッチ」をご参照ください。

VPC ルートテーブルにデフォルトでは VPC に向かわないトラフィックをインターネットゲートウェイに送信するルーティングルールが含まれている必要があります。この設定では、エンドポイントへの接続が、プライベート IP アドレスではなくレプリケーション インスタンスのパブリック IP アドレスから行われているかのように見えます。詳細については、Amazon VPC ユーザーガイド の「VPCルートテーブル」をご参照ください。

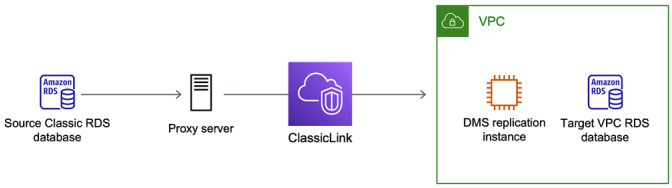
を使用した VPC 外の RDS DB インスタンスから VPC 内の DB インスタンスへの設定 ClassicLink

2022 年 8 月 15 日に、EC2-Classic の提供を終了し h ます。EC2-Classic は、VPC への移行をお勧めします。詳細については、「Amazon EC2 ユーザーガイド」の「EC2-Classic から VPC へ移行」およびブログ記事「EC2-Classic ネットワーキングがリタイア — 準備方法」を参照してください。

VPC 内にない Amazon RDS DB インスタンスを VPC 内の DMS レプリケーションサーバーと DB インスタンスに接続するには、プロキシサーバー ClassicLink で を使用できます。

ClassicLink では、EC2-Classic DB インスタンスを同じ AWS リージョン内のアカウント内の VPC にリンクできます。リンクを作成すると、ソース DB インスタンスはプライベート IP アドレスを使用して VPC 内のレプリケーション インスタンスと通信できます。

VPC のレプリケーションインスタンスは、 を使用して EC2-Classic プラットフォームのソース DB インスタンスに直接アクセスできないため ClassicLink、プロキシサーバーを使用します。プロキシサーバーはソース DB インスタンスを、レプリケーション インスタンスとターゲット DB インスタンスを含む VPC に接続します。プロキシサーバーは ClassicLink を使用して VPC に接続します。プロキシサーバーでのポート転送により、VPC 内にあるソース DB インスタンスとターゲット DB インスタンスの間で通信が可能になります。



AWS Database Migration Service ClassicLink での の使用

VPC 内にない Amazon RDS DB インスタンスを、VPC 内の AWS DMS レプリケーションサーバーと DB インスタンスに接続できます。そのためには、プロキシサーバーで Amazon EC2 ClassicLink を使用できます。

次の手順は、この目的のために ClassicLink を使用する方法を示しています。この手順では、VPC 内にない Amazon RDS ソース DB インスタンスを、 AWS DMS レプリケーションインスタンスとターゲット DB インスタンスを含む VPC に接続します。

- VPC に AWS DMS レプリケーションインスタンスを作成します。(すべてのレプリケーション インスタンスが VPC 内で作成されます)。

- VPC セキュリティグループをレプリケーション インスタンスとターゲット DB インスタンスに関連付けます。2 つのインスタンスが VPC セキュリティグループを共有している場合、デフォルトで相互に通信できます。
- EC2 Classic インスタンスでプロキシサーバーをセットアップします。
- プロキシサーバーと VPC ClassicLink の間で を使用して接続を作成します。
- ソースデータベースとターゲットデータベースの AWS DMS エンドポイントを作成します。
- AWS DMS タスクを作成します。

ClassicLink を使用して、VPC 外の DB インスタンスのデータベースを VPC 内の DB インスタンスのデータベースに移行するには

1. AWS DMS レプリケーションインスタンスを作成し、VPC セキュリティグループを割り当てます。

   a. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。

      AWS Identity and Access Management (IAM) ユーザーとしてサインインしている場合は、にアクセスするための適切なアクセス許可があることを確認してください AWS DMS。データベース移行に必要なアクセス許可の詳細については、「AWS DMSの使用に必要なIAM アクセス許可」をご参照ください。

   b. [Dashboard] ページで、[Replication Instance] を選択します。ステップ 1: AWS DMS コンソールを使用してレプリケーションインスタンスを作成するの手順に従って、レプリケーション インスタンスを作成します。

   c. AWS DMS レプリケーションインスタンスを作成したら、EC2 サービスコンソールを開きます。ナビゲーションペインでC [Network Interfaces] (ネットワーク インターフェース) を選択します。

   d. DMSNetworkInterface を選択し、アクションメニューからセキュリティグループの変更を選択します。

   e. レプリケーション インスタンスとターゲット DB インスタンスに使用するセキュリティグループを選択します。

2. 前のステップのセキュリティグループをターゲット DB インスタンスに関連付けます。

   a. Amazon RDS サービスコンソールを開きます。[Replication engine version] (レプリケーション エンジンバージョン) でバージョン番号を選択し、[Modify] (変更) を選択します。

b. ターゲット DB インスタンスを選択します。[Instance Actions] (インスタンスアクション) では[Modify] (変更) を選択します。

c. [Security Group] (セキュリティグループ) パラメーターの場合、前のステップで使用したセキュリティグループを選択します。

d. [Continue] (続ける) を選択してから、[Modify DB Instance] (DB インスタンス変更) を選択します。

3. ステップ 3: NGINX を使用して EC2 Classic インスタンスでプロキシサーバーを設定します。EC2 Classic インスタンスを起動するには、選択した AMI を使用します。以下の例は、AMI Ubuntu Server 14.04 LTS (HVM) をベースとしています。

   EC2 Classic インスタンスでプロキシサーバーをセットアップするには

   a. EC2 Classic インスタンスに接続し、次のコマンドを使用して NGINX をインストールします。

```
Prompt> sudo apt-get update
Prompt> sudo wget http://nginx.org/download/nginx-1.9.12.tar.gz
Prompt> sudo tar -xvzf nginx-1.9.12.tar.gz
Prompt> cd nginx-1.9.12
Prompt> sudo apt-get install build-essential
Prompt> sudo apt-get install libpcre3 libpcre3-dev
Prompt> sudo apt-get install zlib1g-dev
Prompt> sudo ./configure --with-stream
Prompt> sudo make
Prompt> sudo make install
```

   b. 次のコードを使用して、NGINX デーモンファイル `/etc/init/nginx.conf` を編集します。

```
# /etc/init/nginx.conf – Upstart file

description "nginx http daemon"
author "email"

start on (filesystem and net-device-up IFACE=lo)
stop on runlevel [!2345]
```

```
env DAEMON=/usr/local/nginx/sbin/nginx
env PID=/usr/local/nginx/logs/nginx.pid

expect fork
respawn
respawn limit 10 5

pre-start script
        $DAEMON -t
        if [ $? -ne 0 ]
                then exit $?
        fi
end script

exec $DAEMON
```

c. /usr/local/nginx/conf/nginx.conf に NGINX 設定ファイルを作成します。設定ファイルに、以下の内容を追加します。

```
# /usr/local/nginx/conf/nginx.conf - NGINX configuration file

worker_processes  1;

events {
    worker_connections  1024;
}

stream {
  server {
    listen DB instance port number;
proxy_pass DB instance identifier:DB instance port number;
    }
}
```

d. コマンドラインから、次のコマンドを使用して NGINX を起動します。

```
Prompt> sudo initctl reload-configuration
Prompt> sudo initctl list | grep nginx
```

```
Prompt> sudo initctl start nginx
```

4. プロキシサーバーと、ターゲット DB インスタンスとレプリケーションインスタンスを含むターゲット VPC との間に ClassicLink 接続を作成します。

   a. EC2 コンソールを開き、プロキシサーバーを実行中の EC2 Classic インスタンスを選択します。

   b. アクション で を選択しClassicLink、VPC へのリンク を選択します。

   c. 先に使用したセキュリティグループをこの手順で選択します。

   d. [VPC link] (VPC リンク) を選択します。

5. ステップ 5: の手順を使用して AWS DMS エンドポイントを作成しますステップ 2: ソースエンドポイントとターゲットエンドポイントを指定する。ソース エンドポイントを指定する場合、サーバー名として内部 EC2 DNS ホスト名を使用する必要があります。

6. の手順を使用して AWS DMS タスクを作成しますステップ 3: タスクを作成してデータを移行する。

レプリケーション サブネットグループの作成

データベースの移行に使用するネットワークの一部として、使用する予定の 仮想プライベートクラウド (VPC) のサブネットを指定する必要があります。ここでの VPC は、Amazon VPC サービスに基づいている必要があります。サブネットとは、指定されたアベイラビリティーゾーンにある VPC 内の IP アドレス範囲です。これらのサブネットは、VPC が配置されている AWS リージョンのアベイラビリティーゾーン間で分散できます。

AWS DMS コンソールでレプリケーションインスタンスまたはインスタンスプロファイルを作成するときは、選択したサブネットを使用できます。

レプリケーションサブネットグループを作成し、使用するサブネットを定義できます。少なくとも 2 つのアベイラビリティーゾーンにあるサブネットを指定する必要があります。

レプリケーションサブネットグループを作成する方法

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。

IAM ユーザーとしてサインインしている場合は、 AWS DMSにアクセスするための適切なアクセス許可があることを確認します。データベース移行に必要なアクセス許可の詳細については、「AWS DMSの使用に必要な IAM アクセス許可」をご参照ください。

2. ナビゲーションペインで [サブネットグループ] を選択します。
3. [サブネットグループの作成] を選択します。
4. [レプリケーションサブネットグループの作成] ページで、レプリケーションインスタンスの設定を指定します。次の表で設定について説明します。

オプション	アクション
名前	8 ~ 16 の印刷可能な ASCII 文字 (/、"、@ を除く) を含むレプリケーション サブネットグループ名を入力します。名前は、選択した AWS リージョンのアカウントで一意である必要があります。例えば、実行する AWS リージョンやタスクを含めるなど、名前にインテリジェンスを追加することもできます**DMS-default-VPC** 。
説明	レプリケーション サブネット グループの簡単な説明を入力します。
VPC	データベースの移行に使用する VPC を選択する。VPC では、少なくとも 2 つのアベイラビリティーゾーンに少なくとも 1 つのサブネットが必要であることに注意してください。
サブネットの追加	レプリケーションサブネットグループに含めるサブネットを選択します。少なくとも 2 つのアベイラビリティーゾーンにあるサブネットを選択する必要があります。

5. [サブネットグループの作成] を選択します。

DNS を使用したドメイン エンドポイントの解決

通常、 AWS DMS レプリケーションインスタンスは Amazon EC2 インスタンスのドメインネームシステム (DNS) リゾルバーを使用してドメインエンドポイントを解決します。DNS 解決が必要な場合は、Amazon Route 53 Resolverを使用できます。Route 53 DNS レゾルバーの使用方法の詳細については、「Route 53 レゾルバーの使用開始」をご参照ください。

独自のオンプレミス ネームサーバーを使用して Amazon Route 53 Resolverを使用して特定のエンドポイントを解決する方法については、「 独自のオンプレミスネームサーバーの使用」をご参照ください。

レプリケーション インスタンスのための暗号化キーの設定

AWS DMS は、レプリケーション インスタンスで使用されるストレージとエンドポイント接続情報を暗号化します。DMS は、レプリケーション インスタンスで使用されるストレージを暗号化 AWS KMS key するために、 AWS アカウントに固有の AWS を使用します。この KMS キーは、 AWS Key Management Service () を使用して表示および管理できますAWS KMS。アカウント (aws/dms) でデフォルトの KMS キーあるいは自分で作成するカスタム KMS キーを使用できます。既存の AWS KMS 暗号化キーがある場合は、そのキーを暗号化に使用することもできます。

AWS DMS リソースを暗号化する KMS キー識別子を指定することで、独自の暗号化キーを指定できます。独自の暗号化キーを指定した場合、データベース移行の実行に使用されるユーザーアカウントがそのキーにアクセスできる必要があります。独自の暗号化キーを作成して暗号化キーへのユーザーアクセスを許可する方法の詳細については、AWS KMS 開発者ガイドをご参照ください。

KMS キー識別子を指定しない場合、 AWS DMS はデフォルトの暗号化キーを使用します。KMS は、アカウントの AWS AWS DMS のデフォルトの暗号化キーを作成します。 AWS アカウントには、 AWS リージョンごとに異なるデフォルトの暗号化キーがあります。

AWS DMS リソースの暗号化に使用されるキーを管理するには、 を使用します AWS KMS。ナビゲーションペイン AWS KMS で KMS を検索 AWS Management Console することで、 で を見つけることができます。

AWS KMS は、安全で可用性の高いハードウェアとソフトウェアを組み合わせて、クラウド向けに拡張されたキー管理システムを提供します。を使用すると AWS KMS、暗号化キーを作成し、これらのキーの使用方法を制御するポリシーを定義できます。 は AWS KMS をサポートしているため AWS CloudTrail、キーの使用状況を監査して、キーが適切に使用されていることを確認できます。 AWS KMS キーは、 AWS DMS およびその他のサポートされている AWS サービスと組み合わせて使用

できます。サポート対象 AWS サービスには、Amazon RDS、Amazon S3、Amazon Elastic Block Store (Amazon EBS)、Amazon Redshift が含まれます。

特定の暗号化キーを使用して AWS DMS リソースを作成した場合、それらのリソースの暗号化キーを変更することはできません。 AWS DMS リソースを作成する前に、必ず暗号化キーの要件を確認してください。

レプリケーション インスタンスの作成

データベースを移行する最初のステップは、レプリケーション インスタンスの作成です。レプリケーション インスタンスは、割り当てるタスクを実行してソースデータベースからターゲットデータベースにデータを移行するのに十分なストレージと処理能力を前提とします。このインスタンスの必要なサイズは、移行する必要のあるデータの量、および、インスタンスが実行するタスクにより異なります。レプリケーション インスタンスの詳細については、「AWS DMS レプリケーションインスタンスの使用」をご参照ください。

AWS コンソールを使用してレプリケーションインスタンスを作成するには

1. AWS DMS コンソールのナビゲーションペインでレプリケーションインスタンス を選択し、レプリケーションインスタンスの作成 を選択します。
2. [Create replication instance] ページで、レプリケーションのインスタンス情報を指定します。次の表は、指定できる設定について説明しています。

オプション	アクション
名前	8 ~ 16 の印刷可能な ASCII 文字 (/、"、@ を除く) を含むレプリケーション インスタンス名を入力します。名前は、選択した AWS リージョンのアカウントで一意でなければなりません。例えば、実行する AWS リージョンやタスクなど、名前にインテリジェンスを追加することもできます**`west2-mysql2mysql-instance1`** 。
説明的な Amazon リソースネーム (ARN) - オプション	デフォルトの DMS ARN を上書きするフレンドリ名。作成後に変更することはできない。
説明	レプリケーション インスタンスの簡単な説明を入力します。

オプション	アクション
インスタンスクラス	移行に必要な設定を使用してインスタンスのクラスを選択します。正常に移行を完了するために、インスタンスには十分なストレージ、ネットワーク、処理能力が必要であることに注意してください。移行に最適なインスタンスのクラスを決定する方法に関する詳細については、「AWS DMS レプリケーションインスタンスの使用」をご参照ください。
エンジンバージョン	AWS DMS コンソールでは、サポートされている任意のエンジンバージョンを選択できます。で別のエンジンバージョンを指定しない限り AWS CLI、レプリケーションインスタンスは から最新の非ベータ版の AWS DMS レプリケーションエンジンを実行します AWS CLI。
高可用性	オプションのパラメータを使用して、フェイルオーバーのサポート用に別のアベイラビリティーゾーンにレプリケーション インスタンスのスタンバイレプリカを作成します。変更データキャプチャ (CDC) または継続的なレプリケーションを使用する場合は、このオプションを有効にする必要がある。

オプション	アクション
割り当てられたストレージ (GiB)	ストレージは主に、ログファイルと、キャッシュされたトランザクションで消費されます。キャッシュされたトランザクションでは、ストレージは、キャッシュされたトランザクションをディスクに書き込む必要がある場合にのみ使用されます。したがって、 AWS DMS は大量のストレージを使用しません。例外には次のようなものがあります。 • 膨大なトランザクションをロードする、サイズの大きなテーブル。サイズの大きなテーブルをロードするには時間がかかります。そのため、サイズの大きなテーブルをロードする間、キャッシュされたトランザクションが書き込まれる可能性が高くなります。 • キャッシュされたトランザクションをロードする前に停止するよう設定されているタスク。この場合、すべてのテーブルのロードが完了するまで、すべてのトランザクションがキャッシュされます。この設定では、キャッシュされたトランザクションにより、かなりの量のストレージが消費されることがあります。 • Amazon Redshift にロードされるテーブルを使用する設定になっているタスク。ただし、Amazon Aurora がターゲットのときは、この設定は問題にはなりません。 ほとんどの場合、ストレージのデフォルトの割り当てで十分です。ただしストレージ関連のメトリクスに注意を払うことをお勧めします。デフォルトの割り当てよりも消費量が多い場合は、必ずストレージをスケールアップしてください。

オプション	アクション
ネットワークの種類	DMS は IPv4 アドレス指定プロトコルのネットワークタイプをサポートしており、[デュアルスタック] モードでは、IPv4 と IPv6 の両方のアドレス指定プロトコルのネットワークタイプをサポートする。IPv6 アドレス指定プロトコルのネットワークタイプを介してレプリケーションインスタンスと通信する必要があるリソースがある場合は、[デュアルスタック] モードを使用する。デュアルスタックモードの制限の詳細については、「Amazon Relational Database Service ユーザーガイド」の「デュアルスタックネットワーク DB インスタンスの制限」を参照する。
VPC	使用する VPC を選択します。ソースまたはターゲットデータベースが VPC にある場合、その VPC を選択します。ソースおよびターゲットデータベースが別の VPC にある場合、それらが共にパブリックサブネットにあり、パブリックにアクセス可能であることを確認します。次に、レプリケーション インスタンスを配置する VPC を選択します。レプリケーション インスタンスが、ソース VPC のデータにアクセスできるようにする必要があります。ソースもターゲットデータベースも VPC にない場合は、レプリケーション インスタンスを配置する VPC を選択します。
レプリケーションサブネットグループ	レプリケーション インスタンスを作成する選択した VPC でレプリケーション サブネットグループを選択します。ソースデータベースが VPC にある場合は、レプリケーション インスタンスの場所として、ソースデータベースを含むサブネットグループを選択します。レプリケーション サブネットグループの詳細については、「 レプリケーション サブネットグループの作成」をご参照ください。

オプション	アクション
パブリックアクセス可能	レプリケーション インスタンスをインターネットからアクセス可能にする場合は、このオプションを選択します。デフォルトはパブリックにアクセス可能で、一度オプションを選択すると、レプリケーションインスタンスを作成した後に変更できない。

3. 必要がある場合は、[Advanced (アドバンスト)] タブを選択して、ネットワークおよび暗号化設定の値を設定します。次の表で設定について説明します。

オプション	アクション
アベイラビリティーゾーン	ソースデータベースが配置されているアベイラビリティーゾーンを選択します。
VPC セキュリティグループ	レプリケーションのインスタンスが VPC 内で作成されます。ソースデータベースが VPC にある場合は、データベースが常駐する DB インスタンスへのアクセス権を提供する VPC セキュリティグループを選択します。
KMS キー	使用する暗号化キーを選択して、レプリケーションのストレージと接続情報を暗号化します。(デフォルト) aws/dms を選択すると、アカウントと AWS リージョンに関連付けられたデフォルト AWS Key Management Service (AWS KMS) キーが使用されます。説明とアカウント番号が、キーの ARN とともに表示されます。暗号化キーの使用の詳細については、「暗号化キーの設定と AWS KMS アクセス許可の指定」をご参照ください。

4. [Maintenance] 設定を指定します。次の表で設定について説明します。メンテナンス設定の詳細については、「AWS DMS メンテナンス ウィンドウの操作」をご参照ください。

オプション	アクション
自動バージョンアップグレード	AWS DMS はメジャーバージョンとマイナーバージョンを区別しません。例えば、バージョン 3.4.x から 3.5.x へのアップグレードはメジャーアップグレードとは見なされないため、すべての変更には下位互換性がある。 [自動バージョンアップグレード] が有効になっている場合、DMS はメンテナンスウィンドウ中にレプリケーションインスタンスのバージョンが非推奨となる場合、自動的にアップグレードを行う。 AutoMinorVersionUpgrade を有効にすると、DMS はレプリケーションインスタンスの作成時に現在のデフォルトエンジンバージョンを使用します。例えば、[エンジンバージョン]を現在のデフォルトバージョン以前のバージョン番号に設定すると、DMS はデフォルトバージョンを使用する。 レプリケーションインスタンスの作成時に が有効になっていAutoMinorVersionUpgradeない場合、DMS はエンジンバージョンパラメータで指定されたエンジンバージョンを使用します。
メンテナンスウィンドウ	週 1 回のシステムメンテナンスを実行できる時間帯 (世界標準時 (UTC)) を選択します。 デフォルト: AWS リージョンごとに 8 時間の時間ブロックからランダムに選択された 30 分の時間枠で、ランダムな曜日に発生します。

5. [Create replication instance] (レプリケーションインスタンスの作成) を選択します。

レプリケーション インスタンスの変更

インスタンスクラスの変更やストレージの増量など、レプリケーション インスタンスの設定を変更できます。

レプリケーション インスタンスを変更する際に、変更内容を即時に適用することができます。変更をすぐに適用するには、 AWS Management Consoleで [Apply Immediately] (すぐに適用) オプションを選択します。または、 を呼び出すときに `--apply-immediately`パラメータを使用するか AWS CLI、DMS API を使用する`true`ときに `ApplyImmediately`パラメータを に設定します。

変更の即時適用を選択しない場合、この変更は保留中の変更キューに保存されます。次のメンテナンスウィンドウ実行中に、キューのすべての保留中の変更が適用されます。

> **Note**
>
> 変更の即時適用を選択した場合、保留中の変更キューにあるすべての変更も同様に適用されます。ダウンタイムを必要とする保留中の変更がある場合、[Apply changes immediately] を選択すると予想外のダウンタイムが発生することがあります。

AWS コンソールを使用してレプリケーション インスタンスを変更するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. ナビゲーションペインで [Replication instances] (レプリケーション インスタンス) を選択します。
3. 変更するレプリケーション インスタンスを選択します。次の表は、行うことができる変更を示しています。

オプション	アクション
名前	レプリケーション インスタンスの名前を変更できます。8 ~ 16 の印刷可能な ASCII 文字 (/、"、@ を除く) を含むレプリケーション インスタンス名を入力します。名前は、選択した AWS リージョンのアカウントで一意でなければなりません。例えば、実行する AWS リージョンやタスクなど、名前にインテ

オプション	アクション
	リジェンスを追加することもできます**west2-mysql2mysql-instance1** 。
説明	レプリケーションインスタンスの簡単な説明を変更したり入力したりできる。
インスタンスクラス	インスタンスクラスを変更できます。移行に必要な設定を使用してインスタンスのクラスを選択します。インスタンスクラスを変更すると、レプリケーション インスタンスが再起動されます。この再起動は、次のメンテナンス ウィンドウ中に行われます。または、[Apply changes immediately] (変更を直ちに適用) オプションを選択するとすぐに実行されます。 移行に最適なインスタンスのクラスを決定する方法に関する詳細については、「AWS DMS レプリケーションインスタンスの使用」をご参照ください。
エンジンバージョン	レプリケーション インスタンスで使用されているエンジンバージョンをアップグレードできます。レプリケーションのエンジンバージョンをアップグレードすると、アップグレード中にレプリケーション インスタンスがシャットダウンされます。
マルチ AZ	このオプションを変更して、フェイルオーバーのサポート用に別のアベイラビリティーゾーンにレプリケーション インスタンスのスタンバイレプリカを作成するか、このオプションを削除できます。変更データキャプチャ (CDC) または継続的なレプリケーションを使用する場合は、このオプションを有効にする必要があります。

オプション	アクション
割り当てられたストレージ (GiB)	ストレージは主に、ログファイルと、キャッシュされたトランザクションで消費されます。キャッシュされたトランザクションでは、ストレージは、キャッシュされたトランザクションをディスクに書き込む必要がある場合にのみ使用されます。したがって、 AWS DMS は大量のストレージを使用しません。例外には次のようなものがあります。 • 膨大なトランザクションをロードする、サイズの大きなテーブル。サイズの大きなテーブルをロードするには時間がかかります。そのため、サイズの大きなテーブルをロードする間、キャッシュされたトランザクションが書き込まれる可能性が高くなります。 • キャッシュされたトランザクションをロードする前に停止するよう設定されているタスク。この場合、すべてのテーブルのロードが完了するまで、すべてのトランザクションがキャッシュされます。この設定では、キャッシュされたトランザクションにより、かなりの量のストレージが消費されることがあります。 • Amazon Redshift にロードされるテーブルを使用する設定になっているタスク。ただし、Amazon Aurora がターゲットのときは、この設定は問題にはなりません。 ほとんどの場合、ストレージのデフォルトの割り当てで十分です。ただし、ストレージ関連のメトリクスに注意を払い、デフォルトの割り当てよりも消費量が多い場合はストレージを拡張することをおすすめします。

オプション	アクション
ネットワークの種類	DMS は IPv4 アドレス指定プロトコルのネットワークタイプをサポートしており、[デュアルスタック] モードでは、IPv4 と IPv6 の両方のアドレス指定プロトコルのネットワークタイプをサポートする。IPv6 アドレス指定プロトコルのネットワークタイプを介してレプリケーションインスタンスに通信する必要があるリソースがある場合は、[デュアルスタック] モードを選択する。デュアルスタックモードの制限の詳細については、「Amazon Relational Database Service ユーザーガイド」の「デュアルスタックネットワーク DB インスタンスの制限」を参照する。
VPC セキュリティグループ	レプリケーションのインスタンスが VPC 内で作成されます。ソースデータベースが VPC にある場合は、データベースが常駐する DB インスタンスへのアクセス権を提供する VPC セキュリティグループを選択します。

オプション	アクション
自動バージョンアップグレード	AWS DMS はメジャーバージョンとマイナーバージョンを区別しません。例えば、バージョン 3.4.x から 3.5.x へのアップグレードはメジャーアップグレードとは見なされないため、すべての変更には下位互換性がある。[自動バージョンアップグレード] が有効になっている場合、DMS はメンテナンスウィンドウ中にレプリケーションインスタンスのバージョンが非推奨となる場合、自動的にアップグレードを行う。 [自動バージョンアップグレード] が有効になっている場合、DMS はレプリケーションインスタンスの作成時に現在のデフォルトのエンジンバージョンを使用する。例えば、[エンジンバージョン]を現在のデフォルトバージョン以前のバージョン番号に設定すると、DMS はデフォルトバージョンを使用する。 レプリケーションインスタンスの作成時に [自動バージョンアップグレード] が有効になっていない場合、DMS は [エンジンバージョン] パラメータで指定されたエンジンバージョンを使用する。
メンテナンスウィンドウ	週 1 回のシステムメンテナンスを実行できる時間帯 (世界標準時 (UTC)) を選択します。 デフォルト: AWS リージョンごとに 8 時間の時間ブロックからランダムに選択された 30 分の時間枠で、ランダムな曜日に発生します。

オプション	アクション
Apply changes immediately	行った変更をすぐに適用するには、このオプションを選択します。選択した設定によっては、このオプションを選択すると、レプリケーション インスタンスが直ちに再起動される可能性があります。 AWS DMS が変更を適用している際に [接続のテスト] をクリックすると、エラーメッセージが表示される。AWS DMS がレプリケーション インスタンスに変更を適用したら、接続をもう一度テストを選択します。
Apply changes during next scheduled maintenance window	次に予定されているメンテナンスウィンドウまで DMS による変更の適応を待機させる場合に、このオプションを選択する。

レプリケーション インスタンスを再起動する

AWS DMS レプリケーションインスタンスを再起動して、レプリケーションエンジンを再起動できます。再起動すると、レプリケーション インスタンスは一時的に機能停止になります。その間、インスタンスのステータスは [Rebooting] に設定されます。 AWS DMS インスタンスがマルチ AZ 用に設定されている場合、フェイルオーバーを使用して再起動を実行できます。再起動が完了すると、AWS DMS イベントが作成されます。

AWS DMS インスタンスがマルチ AZ 配置の場合、再起動時に、あるアベイラビリティーゾーンから別の AWS アベイラビリティーゾーンへの計画的なフェイルオーバーを強制的に実行できます。AWS DMS インスタンスの計画的なフェイルオーバーを強制すると、 AWS DMS は、別のアベイラビリティーゾーンのスタンバイインスタンスに自動的に切り替える前に、現在のインスタンスのアクティブな接続を閉じます。計画されたフェイルオーバーで再起動すると、レプリケーション AWS DMS インスタンスクラスのスケーリング時など、インスタンスの計画されたフェイルオーバーイベントをシミュレートできます。

> **Note**
>
> 再起動によって、あるアベイラビリティーゾーンから別のアベイラビリティーゾーンへのフェイルオーバーが強制されると、アベイラビリティーゾーンの変更は数分間反映されない

ことがあります。この遅延は、、 AWS Management Console、および AWS CLI AWS DMS API の呼び出しに表示されます。

再起動時にレプリケーションインスタンス上で移行タスクが実行されている場合、データ損失は発生しません。ただし、タスクは停止して、タスクのステータスがエラー状態に変わります。

移行タスク内のテーブルが一括ロード (フルロードフェーズ) の途中で、まだ開始されていない場合、テーブルはエラー状態になります。ただし、その時点で完了したテーブルは完全な状態で維持されます。フルロードフェーズ中に再起動が発生した場合は、次のいずれかのステップを実行することをお勧めします。

- ステータスが完了であるテーブルをタスクから削除して、残りのテーブルについてタスクを再開する。
- ステータスがエラーのテーブルと保留中のテーブルを対象とする新しいタスクを作成する。

移行タスクのテーブルが継続的なレプリケーションフェーズにある場合、再起動が完了するとタスクが再開されます。

ステータスが Available 状態でない場合、 AWS DMS レプリケーションインスタンスを再起動することはできません。 AWS DMS インスタンスは、以前にリクエストされた変更やメンテナンスウィンドウアクションなど、いくつかの理由で使用できない場合があります。 AWS DMS レプリケーション インスタンスの再起動に必要な時間は、通常短い (5 分未満) です。

AWS コンソールを使用したレプリケーション インスタンスの再起動

レプリケーション インスタンスを再起動するには、 AWS コンソールを使用します。

AWS コンソールを使用してレプリケーション インスタンスを再起動するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. ナビゲーションペインで [Replication instances] (レプリケーション インスタンス) を選択します。
3. 再起動するレプリケーション インスタンスを選択します。
4. [再起動] を選択します。[レプリケーションインスタンスの再起動] ダイアログボックスが開きます。

5. レプリケーションインスタンスをマルチ AZ 配置向けに設定し、別の AWS アベイラビリティーゾーンにフェイルオーバーする場合は [Reboot with planned failover?] チェックボックスをオンにします。
6. [再起動] を選択します。

CLI を使用してレプリケーション インスタンスを再起動する

レプリケーション インスタンスを再起動するには、次のパラメータを指定して コマンドを使用します AWS CLI [reboot-replication-instance](#)。

- `--replication-instance-arn`

Example シンプルな再起動の例

次の AWS CLI 例では、レプリケーションインスタンスを再起動します。

```
aws dms reboot-replication-instance \
--replication-instance-arn arn of my rep instance
```

Example フェイルオーバーによるシンプルな再起動の例

次の AWS CLI 例では、フェイルオーバーを使用してレプリケーション インスタンスを再起動します。

```
aws dms reboot-replication-instance \
--replication-instance-arn arn of my rep instance \
--force-planned-failover
```

API を使用してレプリケーション インスタンスを再起動する

レプリケーション インスタンスを再起動するには、以下のパラメータを指定して AWS DMS API [RebootReplicationInstance](#)アクションを使用します。

- `ReplicationInstanceArn = arn of my rep instance`

Example シンプルな再起動の例

次のコードの例では、レプリケーション インスタンスを再起動します。

```
https://dms.us-west-2.amazonaws.com/
?Action=RebootReplicationInstance
&DBInstanceArn=arn of my rep instance
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140425/us-east-1/dms/aws4_request
&X-Amz-Date=20140425T192732Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=1dc9dd716f4855e9bdf188c70f1cf9f6251b070b68b81103b59ec70c3e7854b3
```

Example フェイルオーバーによるシンプルな再起動の例

次のコード例では、レプリケーションインスタンスを再起動し、別の AWS アベイラビリティーゾーンにフェイルオーバーします。

```
https://dms.us-west-2.amazonaws.com/
?Action=RebootReplicationInstance
&DBInstanceArn=arn of my rep instance
&ForcePlannedFailover=true
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140425/us-east-1/dms/aws4_request
&X-Amz-Date=20140425T192732Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=1dc9dd716f4855e9bdf188c70f1cf9f6251b070b68b81103b59ec70c3e7854b3
```

レプリケーション インスタンスを削除する

AWS DMS レプリケーション インスタンスは、使用が終了したら削除できます。レプリケーション インスタンスを使用する移行タスクがある場合は、レプリケーション インスタンスを削除する前に、タスクを停止して削除する必要があります。

AWS アカウントを閉鎖すると、アカウントに関連付けられているすべての AWS DMS リソースと設定が 2 日後に削除されます。これらのリソースには、すべてのレプリケーション インスタンス、ソースおよびターゲットのエンドポイント設定、レプリケーションタスク、および SSL 証明書が含まれます。2 日後に AWS DMS を再度使用する場合は、必要なリソースを再作成します。

レプリケーションインスタンスが削除基準をすべて満たしていて、長期間ステータスが DELETING のままの場合は、サポートに連絡して問題のトラブルシューティングを行います。

AWS コンソールを使用したレプリケーション インスタンスの削除

レプリケーション インスタンスを削除するには、 AWS コンソールを使用します。

AWS コンソールを使用してレプリケーション インスタンスを削除するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. ナビゲーションペインで [Replication instances] (レプリケーション インスタンス) を選択します。
3. 削除するレプリケーション インスタンスを選択します。
4. [削除] を選択します。
5. ダイアログボックスで、[Delete] を選択します。

CLI を使用したレプリケーション インスタンスの削除

レプリケーション インスタンスを削除するには、次のパラメータを指定して コマンドを使用します AWS CLI delete-replication-instance。

- `--replication-instance-arn`

Example 削除の例

次の AWS CLI 例では、レプリケーションインスタンスを削除します。

```
aws dms delete-replication-instance \
--replication-instance-arn arn of my rep instance
```

API を使用したレプリケーション インスタンスの削除

レプリケーション インスタンスを削除するには、以下のパラメータを指定して AWS DMS API DeleteReplicationInstanceアクションを使用します。

- `ReplicationInstanceArn = arn of my rep instance`

Example 削除の例

次のコードの例では、レプリケーション インスタンスを削除します。

```
https://dms.us-west-2.amazonaws.com/
?Action=DeleteReplicationInstance
&DBInstanceArn=arn of my rep instance
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140425/us-east-1/dms/aws4_request
&X-Amz-Date=20140425T192732Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=1dc9dd716f4855e9bdf188c70f1cf9f6251b070b68b81103b59ec70c3e7854b3
```

AWS DMS メンテナンス ウィンドウの操作

すべての AWS DMS レプリケーションインスタンスには、利用可能なシステム変更が適用される毎週のメンテナンスウィンドウがあります。メンテナンスウィンドウは、変更やソフトウェアのパッチなどが実行されるタイミングをコントロールする機会と考えることができます。

が特定の 1 週間にメンテナンスが必要である AWS DMS と判断した場合、メンテナンスはレプリケーションインスタンスの作成時に選択した 30 分のメンテナンスウィンドウ中に行われます。 は 30 分のメンテナンスウィンドウ中にほとんどのメンテナンス AWS DMS を完了します。ただし、大規模な変更の場合より長い時間がかかる場合があります。

既存の移行タスクにおけるメンテナンスの効果

インスタンスで AWS DMS 移行タスクが実行されている場合、パッチが適用されると次のイベントが発生します。

- 移行タスクのテーブルで継続的な変更フェーズ (CDC) のレプリケートを実行している場合、 AWS DMS は、タスクを一時停止して、パッチの適応後に再開します。その後、パッチが適用されると、移行は中断されたところから再開します。
- AWS DMS が既存のデータを移行する、または既存のデータを移行して進行中の変更タスクをレプリケートする場合、DMS はパッチの適用中に全ロードフェーズにあるすべてのテーブルの移行を停止して再起動します。また、DMS は、パッチの適用中に CDC フェーズにあるすべてのテーブルを停止して、再開します。

メンテナンスウィンドウ設定の変更

メンテナンスウィンドウの時間枠は、 AWS Management Console、、 AWS CLIまたは AWS DMS API を使用して変更できます。

コンソールを使用したメンテナンス ウィンドウ設定の変更

メンテナンスウィンドウの時間枠は、 AWS Management Consoleを使用して変更できます。

コンソールを使用して優先メンテナンスウィンドウを変更するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. ナビゲーションペインで [Replication instances] (レプリケーション インスタンス) を選択します。
3. 変更するレプリケーション インスタンスを選択してから、[Modify] を選択します。
4. [Maintenance] タブを展開し、メンテナンスウィンドウの日時を選択します。
5. [Apply changes immediately] を選択します。
6. [変更] を選択します。

CLI を使用したメンテナンスウィンドウ設定の変更

希望するメンテナンスウィンドウを調整するには、以下のパラメータを指定して コマンドを使用します AWS CLI `modify-replication-instance`。

- `--replication-instance-identifier`
- `--preferred-maintenance-window`

Example

次の AWS CLI 例では、メンテナンスウィンドウを火曜日の午前 4 時 ~ 午前 4 時 30 分に設定します。。

```
aws dms modify-replication-instance \
--replication-instance-identifier myrepinstance \
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

API を使用したメンテナンスウィンドウ設定の変更

希望するメンテナンスウィンドウを調整するには、以下のパラメータを指定して AWS DMS API [ModifyReplicationInstance](#)アクションを使用します。

- `ReplicationInstanceIdentifier = myrepinstance`
- `PreferredMaintenanceWindow = Tue:04:00-Tue:04:30`

Example

次のコード例では、メンテナンスウィンドウを火曜日の午前 4:00 ~ 4:30 UTC に設定します 。

```
https://dms.us-west-2.amazonaws.com/
?Action=ModifyReplicationInstance
&DBInstanceIdentifier=myrepinstance
&PreferredMaintenanceWindow=Tue:04:00-Tue:04:30
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140425/us-east-1/dms/aws4_request
&X-Amz-Date=20140425T192732Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=1dc9dd716f4855e9bdf188c70f1cf9f6251b070b68b81103b59ec70c3e7854b3
```

AWS DMS エンドポイントの使用

エンドポイントは、データストアに関する接続、データストアタイプ、場所情報を提供します。AWSDatabase Migration Serviceはこの情報を使用してデータストアに接続し、ソースエンドポイントからターゲットエンドポイントにデータを移行します。エンドポイント設定を使用して、エンドポイントの追加の接続属性を指定できます。このような設定で、ログ記録、ファイルサイズ、その他のパラメータを制御できます。エンドポイント設定の詳細については、データストアのドキュメントセクションを参照してください。

以下で、エンドポイントの詳細についてご参照ください。

トピック

ソースおよびターゲットエンドポイントの作成

レプリケーションインスタンスの作成時にソースエンドポイントとターゲットエンドポイントを作成することができます。また、レプリケーションインスタンスの作成後にエンドポイントを作成することもできます。ソースおよびターゲットデータストアには、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、Amazon Relational Database Service (Amazon RDS) DB インスタンス、またはオンプレミスデータベースを使用できます。(いずれかのデータストアが AWS サービス上にある必要があります。AWS DMSを使用してオンプレミス データベースから別のオンプレミスデータベースに移行することはできません。)

以下の手順では、AWS DMS コンソール ウィザードを選択したことを前提とします。このステップは、AWS DMS コンソールのナビゲーションペインで [Endpoints] (エンドポイント)、[Create endpoint] (エンドポイントの作成)の順に選択しても実行できます。コンソールウィザードを使用するときは、ソースエンドポイントとターゲットエンドポイントの両方を同じページで作成します。コンソールウィザードを使用しないときは、各エンドポイントを個別に作成します。

AWS コンソールを使用して、ソースまたはターゲット データベース エンドポイントを指定するには

1. [Connect source and target database endpoints] (ソースおよびターゲット データベース エンドポイントの接続) ページで、ソースまたはターゲットデータベースの接続情報を指定します。次の表で設定について説明します。

使用するオプション	この操作を行います
[Endpoint type] (エンドポイントタイプ)	このエンドポイントに対して、ソースエンドポイントまたはターゲットエンドポイントを選択します。
[Select RDS DB Instance] (RDS DB インスタンスの選択)	エンドポイントが Amazon RDS DB インスタンスの場合は、このオプションを選択します。
[Endpoint identifier] (エンドポイント識別子)	エンドポイントを識別するのに使用する名前を入力します。名前に、**`oracle-source`** または **`PostgreSQL-target`** などのエンドポイントの種類を含めることができます。名前はすべてのレプリケーションインスタンスに対して一意である必要があります。
[Source engine] (ソースエンジン) と[Target engine] (ターゲットエンジン)	エンドポイントであるデータベースエンジンのタイプを選択します。

使用するオプション	この操作を行います
[Access to endpoint database] (エンドポイントデータベースへのアクセス)	エンドポイントデータベース認証情報を指定するために使用するオプションを選択します： • [AWS Secrets Manager] を選択する – AWS Secrets Manager で定義されたシークレットを使用して、次のとおり安全に認証情報を提供する。それらに AWS DMS のアクセスを有効にするこれらのシークレットとシークレット アクセス ロールの作成についての詳細は、「シークレットを使用して AWS Database Migration Service エンドポイントにアクセスするには」をご参照ください。 • [Provide access information manually] (アクセス情報を手動で提供する) — 次に示すように、直接入力するクリアテキスト認証情報を使用します。
[AWS Secrets Manager] を選択する	次のシークレット認証情報を設定します。
Secret ID (シークレット ID)	エンドポイント データベースアクセス用に AWS Secrets Manager で作成したシークレットの完全な Amazon リソースネーム (ARN)、部分 ARN、またはフレンドリネームを入力します。
[IAM role] (IAM ロール)	IAM で作成したシークレットアクセスロールの ARN を入力して、お客様の代理でシークレットIDにより識別されるシークレットへの AWS DMS によるアクセスを提供します。サービスロール作成の詳細については、「シークレットを使用して AWS Database Migration Service エンドポイントにアクセスするには」を参照してください。

使用するオプション	この操作を行います
Oracle Automatic Storage Management (ASM) のシークレットID	(Oracle ASM を使用する Oracle ソース エンドポイントの場合のみ) Oracle ASM アクセス用の AWS Secrets Manager に作成したシークレットの完全なAmazonリソースネーム(ARN) 、部分ARN、またはフレンドリ名を入力します。このシークレットは通常、[Secret ID] (シークレットID) で識別されるシークレットと同じサーバー上のOracle ASMにアクセスするために作成されます。
[IAM role for Oracle ASM] (Oracle ASM の IAM ロール)	(Oracle ASM を使用する Oracle ソースエンドポイントの場合のみ) IAM で作成したシークレットアクセスロールの ARN を入力し、お客様の代理で [Secret ID for Oracle automatic storage management (ASM)] (Oracle Automatic Storage Management (ASM) のシークレットID) で識別されるシークレットへの AWS DMS によるアクセスを提供します。
[Provide access information manually] (アクセス情報を手動で提供する)	次のクリアテキスト認証情報を設定します。
[Server name] (サーバー名)	サーバー名を入力します。オンプレミスデータベースの場合、IP アドレスまたはパブリックホスト名にすることができます。Amazon RDS DB インスタンスの場合、DB インスタンスのエンドポイント (DNS 名) とすることができます。たとえば、**`mysqlsrvinst.abcd12345678.us-west-2.rds.amazonaws.com`** とします。
[Port] (ポート)	データベースが使用するポートを入力します。
[Secure Socket Layer (SSL) mode] (Secure Socket Layer (SSL) モード)	このエンドポイントの接続暗号化を有効にする場合は、SSL モードを選択します。選択したモードにより、証明書、およびサーバー証明書情報の提供を求められることがあります。

使用するオプション	この操作を行います
[User name] (ユーザー名)	データ移行を許可するために必要な権限を持つユーザー名を入力します。必要な権限の詳細については、このユーザーガイドの「ソースまたはターゲットデータベースエンジンに対するセキュリティセクション」をご参照ください。
[Password] (パスワード)	必要な権限のあるアカウントのパスワードを入力します。AWS DMS ソース エンドポイントとターゲット エンドポイントのパスワードには、データベースエンジンに応じて文字制限があります。詳細については、以下のテーブルをご参照ください。
[Database name] (データベース名)	特定のデータベースエンジンで、エンドポイントデータベースとして使用するデータベースの名前。

次の表に、リストされたデータベースエンジンのエンドポイントパスワードとシークレットマネージャーシークレットでサポートされていない文字を示します。エンドポイントのパスワードにカンマ (,) を使用する場合は、AWS DMSインスタンスへのアクセス権を認証するため AWS DMS で提供されている Secrets Manager サポートを使用します。詳細については、「シークレットを使用して AWS Database Migration Service エンドポイントにアクセスするには」を参照してください。

このデータベースエンジン用	エンドポイントパスワードとシークレットマネージャーシークレットでは、次の文字はサポートされていません。
すべて	{ }
Microsoft Azure、ソースとしてのみ	;
Microsoft SQL Server	, ;

このデータベースエンジン用	エンドポイントパスワードとシークレットマネージャーシークレットでは、次の文字はサポートされていません。
MySQL、MariaDB、Amazon Aurora MySQL など、MySQL との互換性があります	;
Oracle	,
PostgreSQL、Amazon Aurora PostgreSQL 互換エディション、Amazon Aurora Serverless (Aurora PostgreSQL 互換エディション向けのターゲットしてのみ)	; + %
Amazon Redshift、ターゲットとしてのみ	, ;

2. 必要に応じて、[エンドポイント設定] と [AWS KMS key] を選択します。[Run test] (テストの実行) を選択して、エンドポイントの接続をテストできます。次の表で設定について説明します。

使用するオプション	この操作を行います
エンドポイント設定	追加の接続パラメータをここに入力する。エンドポイント設定の詳細については、[ソースエンジン] または [ターゲットエンジン] のドキュメントセクションを参照する (具体的にはステップ 1)。 Oracle ASM を使用する Oracle ソースエンドポイントの場合、ステップ 1 の [アクセス情報を手動で提供する] を選択すると、Oracle ASM ユーザー認証情報を指定するために、エンドポイント設定で追加の接続属性を入力する必要がある場合がある。このような Oracle ASM エンドポイントの設定の詳細については、「CDC での Oracle LogMiner または AWS DMS Binary Reader の使用」を参照する。

使用するオプション	この操作を行います
AWS KMS key	使用する暗号化キーを選択して、レプリケーションのストレージと接続情報を暗号化します。[(Default) aws/dms] ((デフォルト) aws/dms) を選択する場合、アカウントおよび AWS リージョンに関連付けられたデフォルト AWS Key Management Service (AWS KMS) キーが使用されます。暗号化キーの使用の詳細については、「暗号化キーの設定と AWS KMS アクセス許可の指定」をご参照ください。
[Test endpoint connection (optional)] (エンドポイント接続のテスト (オプション))	VPC およびレプリケーションインスタンス名を追加します。接続をテストするには、[Run test] (テストの実行) を選択します。

データの移行のソース

AWS Database Migration Service (AWS DMS) では、最も一般的なデータエンジンの多くをデータレプリケーションのソースとして使用できます。データベースソースには、Amazon EC2 インスタンスまたはオンプレミスのデータベースで実行されているセルフマネージド型エンジンを使用できます。または、Amazon RDS や Amazon S3 などの AWS サービス上のデータソースも選択できます。

有効なソースすべての一覧については、「Sources for AWS DMS」を参照してください。

トピック

- AWS DMSのソースとしての Oracle データベースの使用
- のソースとしての Microsoft SQL Server データベースの使用 AWS DMS
- AWS DMS のソースとしての Microsoft Azure SQL Database の使用
- AWS DMS のソースとしての Microsoft Azure SQL Managed Instance の使用
- AWS DMS のソースとしての Microsoft Azure Database for PostgreSQL フレキシブル サーバーの使用
- AWS DMS のソースとしての Microsoft Azure Database for MySQL フレキシブル サーバーの使用
- AWS DMS のソースとしての OCI MySQL Heatwave の使用
- AWS DMS でのソースとしての Google Cloud for MySQL の使用
- AWS DMS でのソースとしての Google Cloud for PostgreSQL の使用

- AWS DMS ソースとしての PostgreSQL データベースの使用
- MySQL 互換データベースの AWS DMSのソースとしての使用
- AWS DMS のソースとしての SAP ASE データベースの使用
- AWS DMS のソースとしての MongoDB の使用
- のソースとしての Amazon DocumentDB (MongoDB 互換) の使用 AWS DMS
- のソースとしての Amazon S3 の使用 AWS DMS
- Linux、Unix、Windows 用 IBM Db2、および Amazon RDS データベース (Db2 LUW) をソースとして使用しています AWS DMS
- AWS DMS のソースとしての IBM Db2 for z/OS データベースの使用

AWS DMSのソースとしての Oracle データベースの使用

を使用して、1 つまたは複数の Oracle データベースからデータを移行できます AWS DMS。ソースとして Oracle データベースを使用すると、 AWS DMSが対応しているどのターゲットにもデータを移行できます。

AWS DMS は、次の Oracle データベースエディションをサポートしています。

- Oracle Enterprise Edition
- Oracle Standard Edition
- Oracle Express Edition
- Oracle Personal Edition

がソースとして AWS DMS サポートする Oracle データベースのバージョンについては、「」を参照してくださいのソース AWS DMS。

Secure Sockets Layer (SSL) を使用して、Oracle エンドポイントとレプリケーションインスタンスとの接続を暗号化できます。Oracle エンドポイントで SSL を使用する方法の詳細については、「Oracle エンドポイントでの SSL のサポート」を参照してください。

AWS DMS は、Oracle 透過的データ暗号化 (TDE) を使用して、ソースデータベースに保管中のデータを暗号化します。Oracle ソースエンドポイントで Oracle TDE を使用する方法については、「のソースとして Oracle を使用するためのサポートされている暗号化方法 AWS DMS」をご参照ください。

AWS は、Oracle エンドポイント (および他のすべてのエンドポイントタイプ) での TLS バージョン 1.2 以降の使用をサポートし、TLS バージョン 1.3 以降の使用を推奨します。

Oracle データベースを AWS DMS ソースエンドポイントとして設定するには、次の手順に従います。

1. が Oracle ソースデータベースにアクセス AWS DMS するための適切なアクセス許可を持つ Oracle ユーザーを作成します。
2. 選択した Oracle データベース設定に準拠する Oracle ソース エンドポイントを作成します。 full-load-only タスクを作成するには、それ以上の設定は必要ありません。
3. 変更データキャプチャ (CDC のみまたは全ロードおよび CDC タスク) を処理するタスクを作成するには、Oracle LogMiner または AWS DMS Binary Reader を選択してデータ変更をキャプチャします。 LogMiner またはバイナリリーダーを選択すると、後のアクセス許可と設定オプションの一部が決まります。 LogMiner と Binary Reader の比較については、次のセクションを参照してください。

> **Note**
>
> 全ロードタスク、CDC のみのタスク、および全ロードおよび CDC タスクの詳細については、「[Creating a task] (タスクの作成)」をご参照ください。

Oracle ソースデータベースと の操作の詳細については AWS DMS、以下のセクションを参照してください。

トピック

- CDC での Oracle LogMiner または AWS DMS Binary Reader の使用
- のセルフマネージド型または AWSマネージド型の Oracle ソースデータベースを設定するためのワークフロー AWS DMS Oracle ソースデータベースの設定
- のソースとしてのセルフマネージド Oracle データベースの使用 AWS DMS
- のソースとしての AWSマネージド Oracle データベースの使用 AWS DMS
- のソースとしての Oracle の使用に関する制限 AWS DMS
- Oracle エンドポイントでの SSL のサポート
- のソースとして Oracle を使用するためのサポートされている暗号化方法 AWS DMS
- のソースとして Oracle を使用するためのサポートされている圧縮方法 AWS DMS

- のソースとして Oracle を使用してネストされたテーブルをレプリケートする AWS DMS
- のソースとして Oracle を使用する場合の Oracle ASM への REDO の保存 AWS DMS
- のソースとして Oracle を使用する場合のエンドポイント設定 AWS DMS
- Oracle のソースデータ型

CDC での Oracle LogMiner または AWS DMS Binary Reader の使用

では AWS DMS、Oracle の変更データキャプチャ (CDC) をソースとして実行するときに REDO ログを読み取るには、Oracle LogMiner と AWS DMS Binary Reader の 2 つの方法があります。LogMiner は、オンライン REDO ログとアーカイブ REDO ログファイルを読み取るための Oracle API です。Binary Reader は、raw REDO ログファイルを直接読み取り、解析する AWS DMS メソッドです。これらのメソッドには次の特徴があります。

機能	LogMiner	Binary Reader
設定しやすい	はい	いいえ
ソースシステムのI/Oと CPU への影響軽減	いいえ	はい
CDC パフォーマンスの向上	いいえ	はい
Oracle テーブル クラスターのサポート	はい	いいえ
すべてのタイプの Oracle ハイブリッド列圧縮（HCC）をサポート	はい	部分的 バイナリリーダーは CDC のタスクでクエリローをサポートしていません。他のすべての HCC タイプが完全にサポートされています。
Oracle 12c でのみ LOB カラムのサポート	いいえ (LOB サポートは Oracle 12c の LogMiner	はい

機能	LogMiner	Binary Reader
	では使用できません)。	
LOB カラムにのみ影響を与える UPDATE ステートメントに対応	いいえ	はい
Oracle Transparent データ暗号化 (TDE) に対応	部分的 Oracle を使用する場合 LogMiner、Amazon RDS for Oracle AWS DMS の列レベルでの TDE 暗号化はサポートされていません。	部分的 Binary Reader は、自己管理 Oracle データベースに対してのみ TDE をサポートします。
すべての Oracle 圧縮法に対応	はい	いいえ
XA トランザクションに対応	いいえ	はい
RAC	はい パフォーマンス上の理由や内部 DMS の制限により、推奨されません。	はい 強く推奨

> **Note**
>
> デフォルトでは、 は Oracle LogMiner for (CDC) AWS DMS を使用します。
> AWS DMS は、Oracle ソースデータベースを使用する場合、透過的なデータ暗号化 (TDE) メソッドをサポートします。指定した TDE 認証情報が正しくない場合、 AWS DMS 移行タスクは失敗せず、暗号化されたテーブルの継続的なレプリケーションに影響する可能性があり

ます。TDE 認証情報の指定の詳細については、「のソースとして Oracle を使用するためのサポートされている暗号化方法 AWS DMS」を参照してください。

LogMiner で を使用する主な利点 AWS DMS は次のとおりです。

- LogMiner は、暗号化オプションや圧縮オプションなど、ほとんどの Oracle オプションをサポートしています。Binary Reader では、すべての Oracle オプションがサポートされているわけではありません (特に圧縮とほとんどの暗号化のオプション)。
- LogMiner では、特に Binary Reader の直接アクセス設定や、REDO ログが Oracle Automatic Storage Management (ASM) を使用して管理されている場合と比較して、設定が簡単になります。
- LogMiner は、 で使用するテーブルクラスターをサポートします AWS DMS。Binary Reader はサポートされません。

で Binary Reader を使用する主な利点 AWS DMS は次のとおりです。

- 大量の変更を伴う移行の場合、Oracle ソースデータベースをホストするコンピュータに I/O または CPU が LogMiner ある程度影響する可能性があります。Binary Reader では、ログが複数のデータベース クエリを行うのではなく、直接マイニングされるため、I/O や CPU に影響を与える可能性が高くありません。
- 変更量の多い移行では、通常、Oracle を使用するよりも Binary Reader を使用する場合の CDC パフォーマンスがはるかに優れています LogMiner。
- Binary Reader は、Oracle バージョン 12c の LOBs の CDC をサポートしています。 LogMiner はサポートしていません。

通常、次のいずれかの状況がない限り、Oracle データベースの LogMiner 移行には Oracle を使用します。

- ソース Oracle データベースで複数の移行タスクを実行する必要がある。
- ソース Oracle データベース上の変更のボリュームまたは REDO ログボリュームが多いか、または変更があり、Oracle ASM も使用している場合。

> **Note**
>
> Oracle LogMiner と AWS DMS Binary Reader の使用を切り替える場合は、必ず CDC タスクを再起動してください。

Oracle ソース データベースでの CDC の設定

Oracle ソース エンドポイントが変更データ キャプチャ (CDC) タスクでデータベースに接続するには、特に追加の接続属性を指定する必要があります。これは、全ロードタスクと CDC タスク、または CDC のみのタスクに当てはまります。指定する追加の接続属性は、REDO ログへのアクセスに使用する方法、Oracle LogMiner または AWS DMS Binary Reader によって異なります。

ソース エンドポイントを作成するときに追加の接続属性を指定します。接続属性の設定が複数ある場合は、空白を追加せずにそれぞれをセミコロンで区切ります (例: `oneSetting;thenAnother`)。

AWS DMS LogMiner はデフォルトで を使用します。これを使用するために、追加の接続属性を指定する必要はありません。

Binary Reader を使用してREDO ログにアクセスするには、以下の追加の接続属性を加えます。

```
useLogMinerReader=N;useBfile=Y;
```

次の追加の接続属性形式を使用して、Binary Reader で ASM を使用するサーバーにアクセスします。

```
useLogMinerReader=N;useBfile=Y;asm_user=asm_username;asm_server=RAC_server_ip_address:port_numb
+ASM;
```

ソースエンドポイントの `Password` リクエストパラメータに、Oracle ユーザーパスワードと ASM パスワードの両方をカンマで区切って含める必要があります。

```
oracle_user_password,asm_user_password
```

Oracle ソースが ASM を使用する場合、Binary Reader でハイ パフォーマンスオプションを使用して、スケールが大きいなトランザクション処理を実行できます。これらのオプションには、並列スレッド数 (`parallelASMReadThreads`) および先読みバッファ数 (`readAheadBlocks`) を指定する

追加の接続属性が含まれます。これらの属性を一括設定すると、CDC タスクのパフォーマンスを大幅に改善できます。次に示す設定は、ほとんどの ASM 設定で良好な結果を提供します。

```
useLogMinerReader=N;useBfile=Y;asm_user=asm_username;asm_server=RAC_server_ip_address:port_numb
+ASM;
    parallelASMReadThreads=6;readAheadBlocks=150000;
```

追加の接続属性がサポートする値の詳細については、「のソースとして Oracle を使用する場合のエンドポイント設定 AWS DMS」をご参照ください。

また、 ASM を使用したOracle ソースで CDC タスクのパフォーマンスは、選択した他の設定によって異なります。これらの設定には、 AWS DMS の追加の接続属性と Oracle ソースを設定するための SQL 設定が含まれます。ASM を使用した Oracle ソースの追加接続属性の詳細については、「のソースとして Oracle を使用する場合のエンドポイント設定 AWS DMS」をご参照ください。

また、適切な CDC スタートポイントを選択する必要があります。通常、これを行う場合、CDC を開始する最も早いオープントランザクションをキャプチャするトランザクション処理点を特定します。そうしないと、CDC タスクは以前のオープントランザクションを見逃す可能性があります。Oracle ソースデータベースの場合、Oracle システム変更番号 (SCN) に基づいて CDC ネイティブのスタートポイントを選択して、この最も早いオープントランザクションを識別できます。詳細については、「CDC 開始点を起点とするレプリケーションの実行」を参照してください。

自己管理型 Oracle データベースをソースとして使用するための CDC 設定の詳細については、「Oracle を使用して REDO ログ LogMiner にアクセスするときに必要なアカウント権限」、「AWS DMS Binary Reader を使用して REDO ログにアクセスするときに必要なアカウント権限」や「Oracle ASM で Binary Reader の使用時に必要なアカウント権限」をご参照ください。

が管理する Oracle データベースの CDC をソースとして設定する方法の詳細については、 AWS のRDS for Oracle ソースで Binary Reader を使用するように CDC タスクを設定する AWS DMS「」および「」を参照してくださいAWS DMSの CDC 用 Binary Reader を使用したソースとして Amazon RDS Oracle スタンバイ(リードレプリカ)を使用する。

のセルフマネージド型または AWSマネージド型の Oracle ソースデータベースを設定するためのワークフロー AWS DMS

のセルフマネージド型または AWSマネージド型の Oracle ソースデータベースを設定するためのワークフロー AWS DMS

セルフマネージド型のソースデータベースインスタンスを設定するには、CDC の実行方法に応じて、次のワークフローステップを使用します。

このワークフロー ステップの場合	を使用して CDC を実行する場合は LogMiner、これを実行します。	Binary Reader を使用して CDC を実行する場合は、これを行います
Oracle アカウントの特権を付与します。	「のセルフマネージド Oracle ソースに必要なユーザーアカウント権限 AWS DMS」を参照してください	「のセルフマネージド Oracle ソースに必要なユーザーアカウント権限 AWS DMS」を参照してください
CDC を使用してレプリケーション用のソースデータベースを準備します。	「を使用した CDC 用の Oracle セルフマネージド型ソースデータベースの準備 AWS DMS」を参照してください	「を使用した CDC 用の Oracle セルフマネージド型ソースデータベースの準備 AWS DMS」を参照してください
CDC に必要な追加の Oracle ユーザー権限を付与します。	「Oracle を使用して REDO ログ LogMiner にアクセスするときに必要なアカウント権限」を参照してください	「AWS DMS Binary Reader を使用して REDO ログにアクセスするときに必要なアカウント権限」を参照してください
ASM を持つ Oracle インスタンスの場合は、 CDC のために ASM へのアクセスに必要な追加のユーザーアカウント権限を付与します。	追加のアクションはありません。 は、追加のアカウント権限なしで Oracle ASM AWS DMS をサポートします。	Oracle ASM で Binary Reader の使用時に必要なアカウント権限 を参照してください。
まだ行っていない場合は、LogMiner または CDC 用 Binary Reader を使用するようにタスクを設定します。	「CDC での Oracle LogMiner または AWS DMS Binary Reader の使用」を参照してください	「CDC での Oracle LogMiner または AWS DMS Binary Reader の使用」を参照してください
Oracle スタンバイを CDC のソースとして設定します。	AWS DMS はソースとして Oracle Standby をサポートしていません。	AWS DMSの CDC 用Binary Reader を使用したソースとしてのセルフ管理 Oracle スタンバイの使用 を参照してください。

AWS管理の Oracle ソースデータベースインスタンスを設定するには、次のワークフローステップを使用します。

このワークフロー ステップの場合	を使用して CDC を実行する場合は LogMiner、これを実行します。	Binary Reader を使用して CDC を実行する場合は、これを行います
Oracle アカウントの特権を付与します。	詳細については、「の AWS マネージド Oracle ソースに必要なユーザーアカウント権限 AWS DMS」を参照してください。	詳細については、「の AWS マネージド Oracle ソースに必要なユーザーアカウント権限 AWS DMS」を参照してください。
CDC を使用してレプリケーション用のソースデータベースを準備します。	詳細については、「 の マネージド Oracle AWSソースの設定 AWS DMS」を参照してください。	詳細については、「 の マネージド Oracle AWSソースの設定 AWS DMS」を参照してください。
CDC に必要な追加の Oracle ユーザー権限を付与します。	追加のアカウント権限は必要ありません。	詳細については、「 の RDS for Oracle ソースで Binary Reader を使用するように CDC タスクを設定する AWS DMS」を参照してください。
まだ行っていない場合は、LogMiner または CDC 用 Binary Reader を使用するようにタスクを設定します。	詳細については、「CDC での Oracle LogMiner または AWS DMS Binary Reader の使用」を参照してください。	詳細については、「CDC での Oracle LogMiner または AWS DMS Binary Reader の使用」を参照してください。
Oracle スタンバイを CDC のソースとして設定します。	AWS DMS はソースとして Oracle Standby をサポートしていません。	詳細については、「AWS DMSの CDC 用 Binary Reader を使用したソースとして Amazon RDS Oracle スタンバイ(リードレプリカ)を使用する」を参照してください。

のソースとしてのセルフマネージド Oracle データベースの使用 AWS DMS

セルフ管理データベースは自分で構成および制御するデータベースで、ローカルのオンプレミスデータベース インスタンスまたは Amazon EC2 上のデータベースのいずれかです。次に、 でセルフマネージド Oracle データベースを使用するときに必要な権限と設定について説明します AWS DMS。

のセルフマネージド Oracle ソースに必要なユーザーアカウント権限 AWS DMS

で Oracle データベースをソースとして使用するには AWS DMS、Oracle エンドポイント接続設定で指定された Oracle ユーザーに次の権限を付与します。

> **Note**
>
> 権限を付与する場合、オブジェクトのシノニムではなく、各オブジェクトの実際の名前を使用します。たとえば、下線のある V_$OBJECT を使用します。下線のない V$OBJECT は使用しません。

```
GRANT CREATE SESSION TO db_user;
GRANT SELECT ANY TRANSACTION TO db_user;
GRANT SELECT ON V_$ARCHIVED_LOG TO db_user;
GRANT SELECT ON V_$LOG TO db_user;
GRANT SELECT ON V_$LOGFILE TO db_user;
GRANT SELECT ON V_$LOGMNR_LOGS TO db_user;
GRANT SELECT ON V_$LOGMNR_CONTENTS TO db_user;
GRANT SELECT ON V_$DATABASE TO db_user;
GRANT SELECT ON V_$THREAD TO db_user;
GRANT SELECT ON V_$PARAMETER TO db_user;
GRANT SELECT ON V_$NLS_PARAMETERS TO db_user;
GRANT SELECT ON V_$TIMEZONE_NAMES TO db_user;
GRANT SELECT ON V_$TRANSACTION TO db_user;
GRANT SELECT ON V_$CONTAINERS TO db_user;
GRANT SELECT ON ALL_INDEXES TO db_user;
GRANT SELECT ON ALL_OBJECTS TO db_user;
GRANT SELECT ON ALL_TABLES TO db_user;
GRANT SELECT ON ALL_USERS TO db_user;
GRANT SELECT ON ALL_CATALOG TO db_user;
GRANT SELECT ON ALL_CONSTRAINTS TO db_user;
GRANT SELECT ON ALL_CONS_COLUMNS TO db_user;
GRANT SELECT ON ALL_TAB_COLS TO db_user;
```

```
GRANT SELECT ON ALL_IND_COLUMNS TO db_user;
GRANT SELECT ON ALL_ENCRYPTED_COLUMNS TO db_user;
GRANT SELECT ON ALL_LOG_GROUPS TO db_user;
GRANT SELECT ON ALL_TAB_PARTITIONS TO db_user;
GRANT SELECT ON SYS.DBA_REGISTRY TO db_user;
GRANT SELECT ON SYS.OBJ$ TO db_user;
GRANT SELECT ON DBA_TABLESPACES TO db_user;
GRANT SELECT ON DBA_OBJECTS TO db_user; -- Required if the Oracle version is earlier
 than 11.2.0.3.
GRANT SELECT ON SYS.ENC$ TO db_user; -- Required if transparent data encryption (TDE)
 is enabled. For more information on using Oracle TDE with AWS DMS, see ####### Oracle
 ###################### AWS DMS.
GRANT SELECT ON GV_$TRANSACTION TO db_user; -- Required if the source database is
 Oracle RAC in AWS DMS versions 3.4.6 and higher.
GRANT SELECT ON V_$DATAGUARD_STATS TO db_user; -- Required if the source database is
 Oracle Data Guard and Oracle Standby is used in the latest release of DMS version
 3.4.6, version 3.4.7, and higher.
```

特定のテーブルリストを使用する場合は、レプリケーションテーブルごとに次の追加権限を付与します。

```
GRANT SELECT on any-replicated-table to db_user;
```

検証機能を使用して LOB 列を検証するには、次の追加の権限を付与します。

```
GRANT EXECUTE ON SYS.DBMS_CRYPTO TO db_user;
```

の代わりにバイナリリーダーを使用する場合は、次の追加の権限を付与します LogMiner。

```
GRANT SELECT ON SYS.DBA_DIRECTORIES TO db_user;
```

ビューを公開するには、次の追加の権限を付与します。

```
GRANT SELECT on ALL_VIEWS to dms_user;
```

ビューを公開するには、ソースエンドポイントに追加のexposeViews=true接続属性も追加します。

サーバーレスレプリケーションを使用する場合は、次の追加のアクセス権限を付与します。

```
GRANT SELECT on dba_segments to db_user;
```

サーバーレスレプリケーションの詳細については、「Serverless AWS DMS の使用」を参照してください。

Oracle 独自の移行前評価を使用する場合は、以下のアクセス権限を追加で付与します。

```
GRANT SELECT on gv_$parameter  to dms_user;
GRANT SELECT on v_$instance to dms_user;
GRANT SELECT on v_$version to dms_user;
GRANT SELECT on gv_$ASM_DISKGROUP to dms_user;
GRANT SELECT on gv_$database to dms_user;
GRANT SELECT on dba_db_links to dms_user;
GRANT SELECT on gv_$log_History to dms_user;
GRANT SELECT on gv_$log to dms_user;
GRANT SELECT ON DBA_TYPES TO db_user;
GRANT SELECT ON DBA_USERS to dms_user;
GRANT SELECT ON DBA_DIRECTORIES to dms_user;
```

Oracle 独自の移行前評価の詳細については、「Oracle の評価」を参照してください。

Oracle スタンバイのオープントランザクションを処理するための前提条件

AWS DMS バージョン 3.4.6 以降を使用する場合は、次の手順を実行して Oracle Standby のオープントランザクションを処理します。

1. プライマリデータベースに AWSDMS_DBLINK という名前のデータベースリンクを作成します。*DMS_USER* は、データベースリンクを使用してプライマリデータベースに接続します。このデータベースリンクは、プライマリデータベースで実行されるオープントランザクションをクエリするためにスタンバイインスタンスから実行されることに注意します。次の例を参照してください。

   ```
   CREATE PUBLIC DATABASE LINK AWSDMS_DBLINK
      CONNECT TO DMS_USER IDENTIFIED BY DMS_USER_PASSWORD
      USING '(DESCRIPTION=
               (ADDRESS=(PROTOCOL=TCP)(HOST=PRIMARY_HOST_NAME_OR_IP)(PORT=PORT))
               (CONNECT_DATA=(SERVICE_NAME=SID))
   ```

```
        )';
```

2. 次の例に示されるとおり、*DMS_USER* を使用したデータベースリンクへの接続が確立されていることを確認します。

```
select 1 from dual@AWSDMS_DBLINK
```

を使用した CDC 用の Oracle セルフマネージド型ソースデータベースの準備 AWS DMS

次の手順を実行して、CDC タスクを実行するソースとして自己管理型 Oracle データベースを準備します：

- がソースデータベースのバージョン AWS DMS をサポートしていることの確認.
- ARCHIVELOG モードがオンになっていることを確認する.
- サプリメンタル ロギングの設定.

がソースデータベースのバージョン AWS DMS をサポートしていることの確認

以下のようなクエリを実行して、現在のバージョンの Oracle ソースデータベースが AWS DMSでサポートされていることを確認します。

```
SELECT name, value, description FROM v$parameter WHERE name = 'compatible';
```

ここで name、value、および description は name の値に基づいてクエリされるデータベース内の任意の列です。このクエリがエラーなしで実行された場合、 はデータベースの最新バージョン AWS DMS をサポートし、移行を続行できます。クエリでエラーが発生した場合、 AWS DMS データベースの最新バージョンはサポートされません。移行を続行するには、まず Oracle データベースを でサポートされているバージョンに変換します AWS DMS。

ARCHIVELOG モードがオンになっていることを確認する

Oracle は、ARCHIVELOG モードと NOARCHIVELOG モードの 2 つの異なるモードで実行できます。CDC タスクを実行するには、ARCHIVELOG モードでデータベースを実行します。データベースが ARCHIVELOG モードにあるかどうかを確認するため、次のクエリを実行します。

```
SQL> SELECT log_mode FROM v$database;
```

もし NOARCHIVELOG mode が返されれば、オラクルの命令に従ってデータベースを ARCHIVELOG に設定します。

サプリメンタル ロギングの設定

進行中の変更をキャプチャするには、Oracle ソースデータベースで最小限のサプリメンタルログを有効にする AWS DMS 必要があります。さらに、データベース内のレプリケーションされた各テーブルでサプリメンタル ロギングを有効にする必要があります。

デフォルトでは、 はレプリケートされたすべてのテーブルにPRIMARY KEYサプリメンタルログ AWS DMS を追加します。 AWS DMS がPRIMARY KEYサプリメンタルログを追加できるようにするには、レプリケートされたテーブルごとに次の権限を付与します。

```
ALTER on any-replicated-table;
```

追加の接続属性 AWS DMS を使用して、追加されたデフォルトのPRIMARY KEYサプリメンタルログを無効にすることができますaddSupplementalLogging。詳細については、「のソースとしてOracle を使用する場合のエンドポイント設定 AWS DMS」を参照してください。

プライマリ キー列を参照しない WHERE 句を使用してレプリケーション タスクでテーブルを更新する場合は、必ずサプリメンタル ログを有効にしてください。

サプリメンタル ログをマニュアルでセットアップするには

1. 次のクエリを実行して、データベースのサプリメンタル ログが有効になっていることを確認します。

   ```
   SELECT supplemental_log_data_min FROM v$database;
   ```

   返される結果が YES または IMPLICIT の場合は、データベースのサプリメンタル ログは有効です。

   そうなっていなければ、次のコマンドを実行して、データベースのサプリメンタル ログを有効にします。

   ```
   ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
   ```

2. 必要なサプリメンタル ログがレプリケートされたテーブルごとに追加されていることを確認します。

以下の点を考慮します。

- ALL COLUMNS サプリメンタル ログがテーブルに追加済みなら、その追加は不要です。
- プライマリキーが存在する場合は、プライマリキーのサプリメンタルロギングを追加します。これを行うには、プライマリキーにサプリメンタル ログを追加する形式を使用するか、サプリメンタル ログをデータベースのプライマリキー列に追加します。

```
ALTER TABLE Tablename ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;
```

- プライマリキーが存在せず、テーブルに一意のインデックスが 1 つある場合、一意のインデックスのすべての列をサプリメンタルロギングに追加します。

```
ALTER TABLE TableName ADD SUPPLEMENTAL LOG GROUP LogGroupName
 (UniqueIndexColumn1[, UniqueIndexColumn2] ...) ALWAYS;
```

  SUPPLEMENTAL LOG DATA (UNIQUE INDEX) COLUMNS を使用しても、一意のインデックス列はログに追加されません。
- プライマリキーが存在しず、テーブルに複数の一意のインデックスがある場合、 はアルファベット順に並べられた昇順リストで最初の一意のインデックス AWS DMS を選択します。前の項目と同様に、選択したインデックスの列にサプリメンタル ログを追加する必要があります。

  SUPPLEMENTAL LOG DATA (UNIQUE INDEX) COLUMNS を使用しても、一意のインデックス列はログに追加されません。
- プライマリキーが存在せず、一意のインデックスがない場合は、すべての列にサプリメンタルロギングを追加します。

```
ALTER TABLE TableName ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;
```

  場合によっては、ターゲットテーブルのプライマリキーまたは一意のインデックスは、ソーステーブルのプライマリキーまたは一意のインデックスと異なります。その場合、ターゲットテーブルのプライマリ キーまたは一意のインデックスを構成するソース テーブル列でサプリメンタル ログを手動で追加します。

ターゲットテーブルのプライマリキーを変更する場合、ソースプライマリキーまたは一意のインデックスではなく、ターゲットの一意のインデックスの列にサプリメンタルロギングを追加する必要があります。

テーブルにフィルターまたは変換が定義されている場合は、追加のロギングを有効にする必要があります。

以下の点を考慮します。

- ALL COLUMNS サプリメンタル ログがテーブルに追加済みなら、その追加は不要です。
- テーブルに一意のインデックスやプライマリ キーがある場合、フィルターまたは変換に関係する列ごとにサプリメンタル ログを追加します。ただし、これらの列がプライマリ キーまたは一意のインデックス列と異なる場合にのみ行ってください。
- 変換で列が 1 つしか使用されない場合は、この列をサプリメンタル ログ グループに追加しないでください。たとえば、変換 A+B の場合は、列 A と B の両方にサプリメンタルロギングを追加します。ただし、変換 substring(A,10) の場合、列 A にサプリメンタルロギングを追加しないでください。
- プライマリ キーや一意のインデックス列、フィルタリングまたは変換されるその他の列の両方にサプリメンタル ログを設定するには、USER_LOG_GROUP サプリメンタル ログを追加できます。プライマリ キー/一意のインデックス列、フィルタリングまたは変換されるその他の特定の列の両方にこのサプリメンタル ログを追加します。

  たとえば、TEST.LOGGING という名前のテーブルをプライマリ キー ID と列 NAME によるフィルターでレプリケーションするには、次のようなコマンドを実行して、ロググループのサプリメンタル ログを作成します。

```
ALTER TABLE TEST.LOGGING ADD SUPPLEMENTAL LOG GROUP TEST_LOG_GROUP (ID, NAME) ALWAYS;
```

Oracle を使用して REDO ログ LogMiner にアクセスするときに必要なアカウント権限

Oracle を使用して REDO ログにアクセスするには LogMiner、Oracle エンドポイント接続設定で指定された Oracle ユーザーに次の権限を付与します。

```
GRANT EXECUTE on DBMS_LOGMNR to db_user;
GRANT SELECT on V_$LOGMNR_LOGS to db_user;
GRANT SELECT on V_$LOGMNR_CONTENTS to db_user;
```

```
GRANT LOGMINING to db_user; -- Required only if the Oracle version is 12c or higher.
```

AWS DMS Binary Reader を使用して REDO ログにアクセスするときに必要なアカウント権限

AWS DMS Binary Reader を使用して REDO ログにアクセスするには、Oracle エンドポイント接続設定で指定された Oracle ユーザーに次の権限を付与します。

```
GRANT SELECT on v_$transportable_platform to db_user;   -- Grant this privilege if the
 redo logs are stored in Oracle Automatic Storage Management (ASM) and AWS DMS accesses
 them from ASM.
GRANT CREATE ANY DIRECTORY to db_user;                  -- Grant this privilege to
 allow AWS DMS to use Oracle BFILE read file access in certain cases. This access is
 required when the replication instance doesn't have file-level access to the redo logs
 and the redo logs are on non-ASM storage.
GRANT EXECUTE on DBMS_FILE_TRANSFER to db_user;         -- Grant this privilege to copy
 the redo log files to a temporary folder using the CopyToTempFolder method.
GRANT EXECUTE on DBMS_FILE_GROUP to db_user;
```

Binary Reader は、Oracle ディレクトリを含む Oracle ファイル機能で動作します。各 Oracle ディレクトリオブジェクトには、処理する REDO ログファイルが格納されているフォルダの名前が含まれます。これらの Oracle ディレクトリは、ファイルシステムレベルでは表されません。代わりに、これらは Oracle データベースレベルで作成される論理ディレクトリとなります。これらのビューは、Oracle `ALL_DIRECTORIES` ビューで表示できます。

これらの Oracle ディレクトリ AWS DMS を作成する場合は、前述の権限を付与します。 `CREATE ANY DIRECTORY` は、 `DMS_` プレフィックスを持つディレクトリ名 AWS DMS を作成します。`CREATE ANY DIRECTORY` 権限を付与しない場合は、対応するディレクトリを手動で作成します。このような場合、Oracle ディレクトリを手動で作成すると、Oracle ソースエンドポイントで指定された Oracle ユーザーがこれらのディレクトリを作成したユーザーではない場合があります。このような場合は、`READ on DIRECTORY` 権限も付与します。

Oracle ソースエンドポイントが Active Dataguard Standby (ADG) にある場合は、 AWS データベースブログの「ADG でバイナリリーダーを使用する方法」の投稿を参照してください。

> **Note**
>
> AWS DMS CDC は、自動 REDO トランスポートサービスを使用するように設定されていないアクティブデータ保護スタンバイをサポートしていません。

このような場合、Oracle 管理ファイル (OMF) を使用してログを保存することがあります。または、ソース エンドポイントは ADG にあり、CREATE ANY DIRECTORY 権限を付与できません。このような場合は、 AWS DMS レプリケーションタスクを開始する前に、可能なすべてのログの場所を含むディレクトリを手動で作成します。 AWS DMS が予期した事前に作成されたディレクトリを見つけられない場合、タスクは停止します。また、 AWS DMS は ALL_DIRECTORIES ビューに作成したエントリを削除しないため、手動で削除します。

Oracle ASM で Binary Reader の使用時に必要なアカウント権限

Binary Reader を使用して Automatic Storage Management (ASM) の REDO ログにアクセスするには、Oracle エンドポイント接続設定で指定された Oracle ユーザーに次の権限を付与します。

```
SELECT ON v_$transportable_platform
SYSASM -- To access the ASM account with Oracle 11g Release 2 (version 11.2.0.2) and
 higher, grant the Oracle endpoint user the SYSASM privilege. For older supported
 Oracle versions, it's typically sufficient to grant the Oracle endpoint user the
 SYSDBA privilege.
```

ASM アカウントへのアクセスを検証するには、上記で指定された Oracle バージョンに応じて、コマンド プロンプトを開き、次のいずれかのステートメントを呼び出します。

SYSDBA 特権が必要な場合は、以下を使用します。

```
sqlplus asmuser/asmpassword@+asmserver as sysdba
```

SYSASM 特権が必要な場合は、以下を使用します。

```
sqlplus asmuser/asmpassword@+asmserver as sysasm
```

AWS DMSの CDC 用Binary Reader を使用したソースとしてのセルフ管理 Oracle スタンバイの使用

CDC 用 Binary Reader を使用するときにソースとして Oracle Standby インスタンスを構成するには、次の前提条件からスタートします：

- AWS DMS は現在、Oracle Active Data Guard スタンバイのみをサポートしています。
- Oracle Data Guard の構成で次のものが使用されていることを確認します：
  - REDO データの自動転送用 REDO トランスポート サービス。
  - サービスを適用して、スタンバイデータベースに REDO を自動的に適用します。

上記の要件が満たされていることを確認するには、次のクエリを実行します。

```
SQL> select open_mode, database_role from v$database;
```

クエリの出力で、スタンバイデータベースが読み取り専用モードで開かれており、REDO が自動的に適用されていることを確認します。例:

```
OPEN_MODE             DATABASE_ROLE
--------------------  ----------------
READ ONLY WITH APPLY  PHYSICAL STANDBY
```

CDC 用 Binary Reader を使用するときにソースとして Oracle スタンバイインスタンスを設定するには

1. スタンバイ ログファイルへのアクセスに必要な追加の権限を付与します。

   ```
   GRANT SELECT ON v_$standby_log TO db_user;
   ```

2. AWS Management Console または AWS CLIを使用して、Oracle スタンバイのソースエンドポイントを作成します。エンドポイントを作成する場合、次の追加の接続属性を指定します。

   ```
   useLogminerReader=N;useBfile=Y;
   ```

   > **Note**
   >
   > では AWS DMS、追加の接続属性を使用して、REDO ログの代わりにアーカイブログから移行するかどうかを指定できます。詳細については、「のソースとして Oracle を使用する場合のエンドポイント設定 AWS DMS」を参照してください。

3. アーカイブログの保存先を設定します。

   ASM を使用しない Oracle ソースの DMS Binary Reader は、Oracle ディレクトリを使用してアーカイブの REDO ログにアクセスします。データベースがアーカイブログの宛先として高速リカバリ領域 (FRA) を使用するように設定されている場合、アーカイブ REDO ファイルの場所は一定ではありません。アーカイブ REDO ログが毎日生成されると、YYYY_MM_DD 形式のディレクトリ名を使用して、FRA に新しいディレクトリが作成されます。例:

   ```
   DB_RECOVERY_FILE_DEST/SID/archivelog/YYYY_MM_DD
   ```

DMS が新しく作成された FRA ディレクトリ内のアーカイブ REDOファイルにアクセスする必要があり、プライマリの読み取り/書き込みデータベースがソースとして使用されている場合、DMS は次のとおり新しい Oracle ディレクトリを作成するか、既存の Oracle ディレクトリを置き換えます。

```
CREATE OR REPLACE DIRECTORY dmsrep_taskid AS 'DB_RECOVERY_FILE_DEST/SID/archivelog/
YYYY_MM_DD';
```

スタンバイデータベースがソースとして使用されている場合、データベースは読み取り専用モードであるため、DMS は Oracle ディレクトリを作成または置換できません。ただし、次の追加手順のいずれかを実行できます。

a. Oracle が毎日サブディレクトリを作成しない設定の場合、FRA の代わりに実際のパスを使用するように log_archive_dest_id_1 を変更します。

```
ALTER SYSTEM SET log_archive_dest_1='LOCATION=full directory path'
```

次に、DMS が使用する Oracle ディレクトリオブジェクトを作成します。

```
CREATE OR REPLACE DIRECTORY dms_archived_logs AS 'full directory path';
```

b. 追加のアーカイブログの保存先と、その保存先を指す Oracle ディレクトリオブジェクトを作成します。例:

```
ALTER SYSTEM SET log_archive_dest_3='LOCATION=full directory path';
CREATE DIRECTORY dms_archived_log AS 'full directory path';
```

次に、タスクのソースエンドポイントに追加の接続属性を追加します。

```
archivedLogDestId=3
```

c. DMS で使用する Oracle ディレクトリオブジェクトを手動で事前作成します。

```
CREATE DIRECTORY dms_archived_log_20210301 AS 'DB_RECOVERY_FILE_DEST/SID/
archivelog/2021_03_01';
CREATE DIRECTORY dms_archived_log_20210302 AS 'DB_RECOVERY_FILE_DEST>/SID>/
archivelog/2021_03_02';
...
```

d. 毎日実行され、必要なディレクトリを作成する Oracle スケジューラジョブを作成します。

AWS DMSの CDC のソースとしての Oracle Cloud Infrastructure (OCI) 上のユーザー管理データベースの使用

ユーザー管理データベースとは、仮想マシン (VM)、ベアメタル、または Exadata サーバー上に作成された Oracle データベースなど、ユーザーが設定して管理するデータベースです。または、Oracle Cloud Infrastructure (OCI) などの専用インフラストラクチャ上で実行される、ユーザーが設定して管理するデータベースもあります。次の情報は、 AWS DMSの変更データキャプチャ (CDC) のソースとして OCI 上の Oracle ユーザー管理データベースを使用する際に必要なアクセス権限と設定について説明しています。

OCI でホストされるユーザー管理の Oracleデータベースを変更データキャプチャのソースとして設定するには

1. OCI 上のユーザーマネージドの Oracle ソースデータベースに必要なユーザーアカウントアクセス権限を付与します。詳細については、「Account privileges for a self-managed Oracle source endpoint」を参照してください。
2. Binary Reader を使用して REDO ログにアクセスする際に必要なアカウントアクセス権限を付与します。詳細については、「Account privileges required when using Binary Reader」を参照してください。
3. Oracle Automatic Storage Management (ASM)で Binary Reader を使用する場合に必要なアカウントアクセス権限を追加します。詳細については、「Additional account privileges required when using Binary Reader with Oracle ASM」を参照してください。
4. サプリメンタルロギングをセットアップします。詳細については、「Setting up supplemental logging」を参照してください。
5. TDE 暗号化を設定します。詳細については、「Encryption methods when using an Oracle database as a source endpoint」を参照してください。

Oracle Cloud Infrastructure (OCI) 上の Oracle ソースデータベースからデータをレプリケートする場合、次の制限が適用されます。

制限事項

- DMS は、Oracle を使用して REDO ログにアクセスする LogMiner ことをサポートしていません。
- DMS は Autonomous DB をサポートしていません。

のソースとしての AWSマネージド Oracle データベースの使用 AWS DMS

AWSマネージドデータベースは、Amazon RDS、Amazon Aurora、Amazon S3 などの Amazon サービス上にあるデータベースです。で AWS管理の Oracle データベースを使用するときに設定する必要がある権限と設定を以下に示します AWS DMS。

の AWSマネージド Oracle ソースに必要なユーザーアカウント権限 AWS DMS

Oracle ソース エンドポイント定義で指定された Oracle ユーザーアカウントに、以下の権限を付与します。

> **Important**
>
> *db_user* や *any-replicated-table* などすべてのパラメータ値では、大文字と小文字を区別する識別子を使用して値を指定しない限り、Oracle は値をすべて大文字と見なします。たとえば、CREATE USER *myuser* または CREATE USER MYUSER のように引用符を使用しない *db_user* の値を作成するとします。この場合、Oracle は値をすべて大文字 (MYUSER) として識別して格納します。CREATE USER "MyUser"やCREATE USER 'MyUser'のように引用符を使用すると、Oracle は、指定した大文字と小文字を区別する値を識別して格納します (MyUser)。

```
GRANT CREATE SESSION to db_user;
GRANT SELECT ANY TRANSACTION to db_user;
GRANT SELECT on DBA_TABLESPACES to db_user;
GRANT SELECT ON any-replicated-table to db_user;
GRANT EXECUTE on rdsadmin.rdsadmin_util to db_user;
 -- For Oracle 12c or higher:
GRANT LOGMINING to db_user; - Required only if the Oracle version is 12c or higher.
```

さらに、SELECT と EXECUTE のアクセス許可を次のように Amazon RDS 手順 rdsadmin.rdsadmin_util.grant_sys_object を使用して SYS オブジェクトに付与します。詳細については、「SYS オブジェクトへの SELECT または EXECUTE 権限の付与」をご参照ください。

```
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_VIEWS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_TAB_PARTITIONS', 'db_user',
 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_INDEXES', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_OBJECTS', 'db_user', 'SELECT');
```

```
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_TABLES', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_USERS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_CATALOG', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_CONSTRAINTS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_CONS_COLUMNS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_TAB_COLS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_IND_COLUMNS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_LOG_GROUPS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$ARCHIVED_LOG', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$LOG', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$LOGFILE', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$DATABASE', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$THREAD', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$PARAMETER', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$NLS_PARAMETERS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$TIMEZONE_NAMES', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$TRANSACTION', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$CONTAINERS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_REGISTRY', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('OBJ$', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_ENCRYPTED_COLUMNS', 'db_user',
 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$LOGMNR_LOGS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$LOGMNR_CONTENTS','db_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_LOGMNR', 'db_user', 'EXECUTE');

-- (as of Oracle versions 12.1 and higher)
exec rdsadmin.rdsadmin_util.grant_sys_object('REGISTRY$SQLPATCH', 'db_user', 'SELECT');

-- (for Amazon RDS Active Dataguard Standby (ADG))
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$STANDBY_LOG', 'db_user', 'SELECT');

-- (for transparent data encryption (TDE))

exec rdsadmin.rdsadmin_util.grant_sys_object('ENC$', 'db_user', 'SELECT');

-- (for validation with LOB columns)
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_CRYPTO', 'db_user', 'EXECUTE');

-- (for binary reader)
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_DIRECTORIES','db_user','SELECT');

-- Required when the source database is Oracle Data guard, and Oracle Standby is used
 in the latest release of DMS version 3.4.6, version 3.4.7, and higher.
```

```
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$DATAGUARD_STATS', 'db_user',
 'SELECT');
```

AWS DMS で Amazon RDS アクティブデータガードスタンバイ (ADG) の使用の詳細については、「AWS DMSの CDC 用 Binary Reader を使用したソースとして Amazon RDS Oracle スタンバイ (リードレプリカ)を使用する」をご参照ください。

で Oracle TDE を使用する方法の詳細については AWS DMS、「」を参照してくださいのソースとして Oracle を使用するためのサポートされている暗号化方法 AWS DMS。

Oracle スタンバイのオープントランザクションを処理するための前提条件

AWS DMS バージョン 3.4.6 以降を使用する場合は、次の手順を実行して Oracle Standby のオープントランザクションを処理します。

1. プライマリデータベースに AWSDMS_DBLINK という名前のデータベースリンクを作成します。*DMS_USER* は、データベースリンクを使用してプライマリデータベースに接続します。このデータベースリンクは、プライマリデータベースで実行されるオープントランザクションをクエリするためにスタンバイインスタンスから実行されることに注意します。次の例を参照してください。

   ```
   CREATE PUBLIC DATABASE LINK AWSDMS_DBLINK
      CONNECT TO DMS_USER IDENTIFIED BY DMS_USER_PASSWORD
      USING '(DESCRIPTION=
               (ADDRESS=(PROTOCOL=TCP)(HOST=PRIMARY_HOST_NAME_OR_IP)(PORT=PORT))
               (CONNECT_DATA=(SERVICE_NAME=SID))
             )';
   ```

2. 次の例に示されるとおり、*DMS_USER* を使用したデータベースリンクへの接続が確立されていることを確認します。

   ```
   select 1 from dual@AWSDMS_DBLINK
   ```

の マネージド Oracle AWSソースの設定 AWS DMS

AWSマネージド Oracle データベースを のソースとして使用する前に AWS DMS、Oracle データベースに対して次のタスクを実行します。

- 自動バックアップを有効化します。自動バックアップの有効化の詳細については、Amazon RDS ユーザーガイド の「自動バックアップの有効化」をご参照ください。
- サプリメンタルロギングをセットアップします。
- アーカイブを設定します。Amazon RDS for Oracle DB インスタンスの REDO ログをアーカイブすると AWS DMS 、 は Oracle LogMiner または Binary Reader を使用してログ情報を取得できます。

アーカイブを設定するには

1. `rdsadmin.rdsadmin_util.set_configuration` コマンドを実行してアーカイブをセットアップします。

   たとえば、アーカイブされた REDO ログを 24 時間保持するには、次のコマンドを実行します。

   ```
   exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24);
   commit;
   ```

   > **Note**
   >
   > 変更を反映するにはコミットが必要です。

2. 指定された保持期間中、アーカイブされた REDO ログのための十分な容量がストレージにあることを確認します。たとえば、保持期間が 24 時間の場合は、通常のトランザクション処理時間における累積アーカイブ REDO ログの合計サイズを計算し、その合計に 24 を掛けます。この計算された 24 時間の合計と、使用可能なストレージ容量を比較し、24 時間のトランザクション処理を処理するのに十分なストレージ領域があるかどうかを判断します。

サプリメンタルロギングをセットアップするには

1. データベース レベルでサプリメンタル ログを有効にするには、次のコマンドを実行します。

```
exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD');
```

2. 次のコマンドを実行してプライマリ キー列のサプリメンタル ログを有効にします。

```
exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD','PRIMARY KEY');
```

3. (オプション) テーブル レベルでキーレベルのサプリメンタル ログを有効にします。

   キーレベルのサプリメンタル ログを有効にすると、ソースデータベースで多少のオーバーヘッドが生じます。したがって、テーブルのサブセットのみを移行する場合は、テーブルレベルでのサプリメンタル ログを有効にすることをお勧めします。キーレベルのサプリメンタル ログをテーブル レベルで有効にするには、次のコマンドを使用します。

```
alter table table_name add supplemental log data (PRIMARY KEY) columns;
```

の RDS for Oracle ソースで Binary Reader を使用するように CDC タスクを設定する AWS DMS

CDC 用 Binary Reader を使用して、ソース Amazon RDS for Oracle インスタンスの REDO ログにアクセスする AWS DMS ように を設定できます。

> **Note**
>
> Oracle を使用するには LogMiner、最低限必要なユーザーアカウント権限で十分です。詳細については、「の AWSマネージド Oracle ソースに必要なユーザーアカウント権限 AWS DMS」を参照してください。

AWS DMS Binary Reader を使用するには、バージョンに応じて AWS DMS 、Oracle ソースエンドポイントの追加設定と追加の接続属性を指定します。

Binary Reader のサポートは、Amazon RDS for Oracle の次のバージョンで利用できます。

- Oracle バージョン 12.1.0.2.v11 以降
- Oracle バージョン 12.1.0.2.v7 以降
- Oracle 12.2 – すべてのバージョン
- Oracle 18.0 – すべてのバージョン
- Oracle 19.0 – すべてのバージョン

Binary Reader を使用して CDC を設定するには

1. Amazon RDS for Oracle ソース データベースにマスターユーザーとしてログインし、次のストアド プロシージャを実行してサーバーレベル ディレクトリを作成します。

```
exec rdsadmin.rdsadmin_master_util.create_archivelog_dir;
exec rdsadmin.rdsadmin_master_util.create_onlinelog_dir;
```

2. Oracle ユーザーアカウントに、Oracle ソース エンドポイントにアクセスするための以下の権限を付与します。

```
GRANT READ ON DIRECTORY ONLINELOG_DIR TO db_user;
GRANT READ ON DIRECTORY ARCHIVELOG_DIR TO db_user;
```

3. Amazon RDS Oracle ソースエンドポイントで、次の追加の接続属性を設定します。

   - RDS Oracle バージョン 11.2 と 12.1 では、次のように設定します。

```
useLogminerReader=N;useBfile=Y;accessAlternateDirectly=false;useAlternateFolderForOnline=
oraclePathPrefix=/rdsdbdata/db/{$DATABASE_NAME}_A/;usePathPrefix=/rdsdbdata/
log/;replacePathPrefix=true;
```

   - RDS Oracle バージョン 12.2、18.0、19.0 では、次のように設定します。

```
useLogminerReader=N;useBfile=Y;
```

> **Note**
>
> 複数の属性設定には、セミコロン区切り文字 (;) の後に空白を含めることはできません、例えば `oneSetting;thenAnother`。

CDC タスクの設定の詳細については、「Oracle ソース データベースでの CDC の設定」をご参照ください。

AWS DMSの CDC 用 Binary Reader を使用したソースとして Amazon RDS Oracle スタンバイ(リードレプリカ)を使用する

AWS DMSで CDC 用 Binary Reader を使用する場合は、ソースとして Amazon RDS for Oracle スタンバイを使用するための次の前提条件を確認します:

- Oracle マスターユーザーを使用して、 Binary Reader を設定します。
- 現在、 が Oracle Active Data Guard Standby のみの使用 AWS DMS をサポートしていることを確認してください。

その後、CDC 用 Binary Reader を使用するときに、次の手順に従ってソースとして RDS for Oracle スタンバイを使用します。

CDC 用 Binary Reader を使用するときにソースとして RDS for Oracle スタンバイを設定するには

1. RDS for Oracle のプライマリインスタンスに管理ユーザーとしてサインインします。
2. Amazon RDS ユーザーガイドに記載されている次のストアドプロシージャを実行して、サーバーレベル ディレクトリを作成します。

```
exec rdsadmin.rdsadmin_master_util.create_archivelog_dir;
exec rdsadmin.rdsadmin_master_util.create_onlinelog_dir;
```

3. ステップ 2 で作成したディレクトリを特定します。

```
SELECT directory_name, directory_path FROM all_directories
WHERE directory_name LIKE ( 'ARCHIVELOG_DIR_%' )
        OR directory_name LIKE ( 'ONLINELOG_DIR_%' )
```

たとえば、前述のコードでは、次のようなディレクトリの一覧が表示されます。

DIRECTORY_NAME	DIRECTORY_PATH
ARCHIVELOG_DIR_A	/rdsdbdata/db/ORCL_A/arch
ARCHIVELOG_DIR_B	/rdsdbdata/db/ORCL_B/arch
ONLINELOG_DIR_A	/rdsdbdata/db/ORCL_A/onlinelog
ONLINELOG_DIR_B	/rdsdbdata/db/ORCL_B/onlinelog

4. Oracle スタンバイにアクセスするために使用する Oracle ユーザーアカウントに対する前述のディレクトリに対する Read の権限を許可します。

```
GRANT READ ON DIRECTORY ARCHIVELOG_DIR_A TO db_user;
GRANT READ ON DIRECTORY ARCHIVELOG_DIR_B TO db_user;
GRANT READ ON DIRECTORY ONLINELOG_DIR_A TO db_user;
GRANT READ ON DIRECTORY ONLINELOG_DIR_B TO db_user;
```

5. プライマリ インスタンスでアーカイブ ログ スイッチを実行します。これを行うと、ALL_DIRECTORIES の変更が Oracle スタンバイにも移植されることを確認します。
6. ALL_DIRECTORIES Oracle Standby でクエリを実行して、変更が適用されたことを確認します。
7. マネジメントコンソールまたは AWS Command Line Interface () を使用して、Oracle スタンバイの AWS DMS ソースエンドポイントを作成しますAWS CLI。エンドポイントの作成時に、次の追加の接続属性を指定します。

```
useLogminerReader=N;useBfile=Y;archivedLogDestId=1;additionalArchivedLogDestId=2
```

8. エンドポイントを作成したら、コンソールのエンドポイントの作成ページでエンドポイント接続のテストを使用するか、 AWS CLI test-connection コマンドを使用して接続が確立されていることを確認します。

のソースとしての Oracle の使用に関する制限 AWS DMS

Oracle データベースを AWS DMSのソースとして使用する場合は、以下の制限が適用されます。

- AWS DMS は、 AWS DMS バージョン 3.5.0 以降の Oracle 拡張データ型をサポートしています。
- AWS DMS は長いオブジェクト名 (30 バイト以上) をサポートしていません。
- AWS DMS は関数ベースのインデックスをサポートしていません。
- サプリメンタル ログを管理し、いずれかの列で変換を実行する場合は、すべてのフィールドと列でサプリメンタル ログが有効になっていることを確認してください。サプリメンタル ログの設定の詳細については、以下のトピックをご参照ください：
  - セルフ管理 Oracle ソースデータベースについては、「サプリメンタル ロギングの設定」をご参照ください。
  - マネージド Oracle ソースデータベースについては、 AWS「」を参照してください の マネージド Oracle AWSソースの設定 AWS DMS。
- AWS DMS は、マルチテナントコンテナルートデータベース (CDB$ROOT) をサポートしていません。しかし、Binary Reader を使用した PDB をサポートしています。
- AWS DMS は遅延制約をサポートしていません。
- AWS DMS バージョン 3.5.1 以降では、安全な LOBs は LOB ルックアップを実行することによってのみサポートされます。

- AWS DMS は、サポートされているすべての Oracle バージョン 11 以降の`rename table table-name to new-table-name`構文をサポートします。この構文は Oracle バージョン 10 ソースデータベースではサポートされていません。
- AWS DMS は DDL ステートメント の結果をレプリケートしませんALTER TABLE ADD *column data_type* DEFAULT *default_value*。ターゲットに *default_value* をレプリケートする代わりに、新しい列を NULL に設定します。
- AWS DMS バージョン 3.4.7 以降を使用している場合、パーティションまたはサブパーティションオペレーションに起因する変更をレプリケートするには、DMS タスクを開始する前に次の操作を行います。
  - パーティション分割されたテーブル構造 (DDL) を手動で作成します。
  - DDL が Oracle ソースと Oracle ターゲットの両方で同じであることを確認します。
  - 追加の接続属性 enableHomogenousPartitionOps=true を設定します。

  enableHomogenousPartitionOps の詳細については、「のソースとして Oracle を使用する場合のエンドポイント設定 AWS DMS」を参照してください。また、フルと CDC タスクでは、DMS はキャッシュした変更の一部としてキャプチャしたデータ変更をレプリケートしないことにも注意します。このようなユースケースでは、Oracle ターゲットでテーブル構造を再作成して、該当テーブルをもう一度ロードします。

  AWS DMS バージョン 3.4.7 より前：

  DMS は、パーティション操作またはサブパーティション操作 (ADD、DROP、EXCHANGE、TRUNCATE) によるデータ変更をレプリケートしません。このような更新により、レプリケーション中に次のエラーが発生する可能性があります：
  - ADD オペレーションの場合、追加されたデータの更新と削除により、「影響を受ける行は 0 です」という警告が表示されることがあります。
  - DROP および TRUNCATE オペレーションの場合、新しい挿入によって「重複」エラーが発生する可能性があります。
  - EXCHANGE オペレーションは、「影響を受ける行は 0 です」警告と「重複」エラーの両方が発生する可能性があります。

  パーティションオペレーションまたはサブパーティションオペレーションによる変更をレプリケートするには、対象のテーブルをリロードします。新しい空のパーティションを追加した後、新しく追加されたテーブルに対するオペレーションは、通常どおりターゲットへのレプリケーションとなります。

- AWS DMS 3.4 より前のバージョンでは、ソースで CREATE TABLE ASステートメントを実行した結果生じるターゲットのデータ変更をサポートしていません。ただし、新しいテーブルがターゲットで作成されます。
- AWS DMS は、テーブルメタデータや OBJECT_IDフィールドなど、Oracle DBMS_REDEFINITIONパッケージによって行われた変更をキャプチャしません。
- AWS DMS は、空の BLOB 列と CLOB 列をターゲットNULLの にマッピングします。
- Oracle 11 で変更をキャプチャする場合 LogMiner、文字列長が 1982 を超える CLOB 列の更新は失われ、ターゲットは更新されません。
- 変更データキャプチャ (CDC) AWS DMS 中、プライマリキーとして定義された数値列へのバッチ更新はサポートされていません。
- AWS DMS は特定のUPDATEコマンドをサポートしていません。次の例は、サポートされていない UPDATE コマンドです。

```
UPDATE TEST_TABLE SET KEY=KEY+1;
```

  ここで、TEST_TABLE はテーブル名であり、KEY は、プライマリキーとして定義された数値列です。
- AWS DMS は、LONG 列と LONG RAW 列をロードするためのフル LOB モードをサポートしていません。この代わりに、制限付き LOB モードを使用してこのようなデータ型を Oracle ターゲットに移行できます。限定 LOB モードでは、 は、64 KB を超える LONG または LONG RAW 列に設定したデータを 64 KB に AWS DMS 切り捨てます。
- AWS DMS は、XMLTYPE 列のロードにフル LOB モードをサポートしていません。この代わりに、制限付き LOB モードを使用して XMLTYPE 列を Oracle ターゲットに移行できます。制限付き LOB モードでは、DMS はユーザー定義の「最大 LOB サイズ」変数以上のデータは切り捨てます。「最大 LOB サイズ」の最大推奨値は 100 MB です。
- AWS DMS は、名前にアポストロフィが含まれているテーブルをレプリケートしません。
- AWS DMS は、マテリアライズドビューからの CDC をサポートします。ただし、DMS はその他のビューからの CDC はサポートしていません。
- AWS DMS は、オーバーフローセグメントを持つインデックスで編成されたテーブルの CDC をサポートしていません。
- AWS DMS は、 を enableHomogenousPartitionOpsに設定して参照によってパーティション分割されたテーブルの Drop Partitionオペレーションをサポートしていませんtrue。
- Oracle を使用して REDO ログ LogMiner にアクセスする場合、 には次の制限 AWS DMS があります。

  - Oracle 12 のみの場合、LOB 列に変更をレプリケー AWS DMS トしません。
  - すべての Oracle バージョンで、 AWS DMS XMLTYPEおよび LOB 列に対するUPDATEオペレーションの結果をレプリケートしません。
  - AWS DMS は、Oracle の使用中にレプリケーションで XA トランザクションをサポートしていません LogMiner。
  - Oracle LogMiner は、プラガブルデータベース (PDB) への接続をサポートしていません。PDB に接続するには、Binary Reader を使用して REDO ログにアクセスします。
  - SHRINK SPACE 操作はサポートされていません。
- Binary Reader を使用する場合、 には次の制限 AWS DMS があります。
  - テーブルクラスターはサポートされていません。
  - テーブルレベルでの SHRINK SPACE オペレーションのみがサポートされます。このレベルには、テーブル全体、パーティション、サブパーティションが含まれます。
  - キー圧縮によるインデックス構成表への変更はサポートされません。
  - RAW デバイスでのオンライン REDO ログの実装はサポートされていません。
  - RDS for Oracle では、TDE 暗号化キーのウォレットパスワードの取得がサポートされていないため、Binary Reader は、セルフマネージド型 Oracle データベースに対してのみ TDE をサポートします。
- AWS DMS は、Oracle Automatic Storage Management (ASM) プロキシを使用した Amazon RDS Oracle ソースへの接続をサポートしていません。
- AWS DMS は仮想列をサポートしていません。
- AWS DMS は、ROWID 列に基づくROWIDデータ型またはマテリアライズドビューをサポートしていません。

  AWS DMS は、Oracle マテリアライズドビューの一部をサポートしています。フルロードの場合、DMS は Oracle マテリアライズドビューのフルロードコピーを実行できます。DMS はマテリアライズドビューをベーステーブルとしてターゲットシステムにコピーして、マテリアライズドビューの ROWID 列は無視します。継続的なレプリケーション (CDC) の場合、DMS は変更をマテリアライズドビューのデータへの変更をレプリケート使用としますが、理想的でない結果となる可能性があります。具体的には、マテリアライズドビューが完全に更新されると、DMS はすべての行の個別の削除をレプリケートし、続いてすべての行の個別の挿入をレプリケートします。これはリソースを非常に大量に消費する作業であり、多数の行を含むマテリアライズドビューのパフォーマンスが低下する可能性があります。マテリアライズドビューが高速リフレッシュを実行する継続的なレプリケーションの場合、DMS は高速リフレッシュのデータの変更を処理してレプリケート

しようとします。どちらの場合も、DMS はマテリアライズドビュー内の ROWID 列はスキップします。

- AWS DMS はグローバル一時テーブルをロードまたはキャプチャしません。
- レプリケーションを使用する S3 ターゲットでは、ソース行の更新がすべての列の値を取得できるように、どの列でもサプリメンタル ログを有効にします。以下に例 `alter table yourtablename add supplemental log data (all) columns;` を示します。
- `null` を含む複合一意キーを持つ行の更新はターゲットでレプリケーション不可です。
- AWS DMS は、同じソースエンドポイントでの複数の Oracle TDE 暗号化キーの使用をサポートしていません。各エンドポイントは、TDE 暗号化キー名「`securityDbEncryptionName`」に対して属性を 1 つ、またこのキーの TDE パスワードが 1 つ持つことができます。
- Amazon RDS for Oracle からレプリケートする場合、TDE は暗号化されたテーブルスペースと Oracle の使用でのみサポートされます LogMiner。
- AWS DMS は、複数のテーブル名変更オペレーションを連続してすばやくサポートしていません。
- Oracle 19.0 をソースとして使用する場合、以下の機能は AWS DMS サポートされていません。
  - データガード DML リダイレクト
  - パーティション分割されたハイブリッドテーブル
  - スキーマのみの Oracle アカウント
- AWS DMS は、 `BIN$`または タイプのテーブルまたはビューの移行をサポートしていませんDR$。
- Oracle 18.x 以降、 AWS DMS Oracle Express Edition (Oracle Database XE) からの変更データキャプチャ (CDC) はサポートされていません。
- CHAR 列からデータを移行する場合、DMS は末尾のスペースをすべて切り捨てます。
- AWS DMS は、アプリケーションコンテナからのレプリケーションをサポートしていません。
- AWS DMS は Oracle Flashback データベースと復元ポイントの実行をサポートしていません。これらのオペレーションは Oracle Redo ログファイルの整合性に影響するためです。
- 並列実行オプションを使用したダイレクトロードの `INSERT` プロシージャは、次の場合にはサポートされません。
  - 255 列を超える非圧縮テーブル
  - 行サイズが 8K 以上
  - Exadata HCC テーブル
  - Big Endian プラットフォームで実行されているデータベース

- プライマリキーも一意キーもないソーステーブルでは、ALL COLUMN サプリメンタルロギングを有効にする必要があります。これにより、さらに多くの REDO ログアクティビティが作成され、DMS CDC のレイテンシーが増加する可能性があります。
- AWS DMS は、ソースデータベース内の非表示の列からデータを移行しません。このような列を移行の範囲に含めるには、ALTER TABLE ステートメントを使用して列を表示します。

Oracle エンドポイントでの SSL のサポート

AWS DMS Oracle エンドポイントは、 noneおよび SSL モードの SSL V3 verify-ca をサポートします。Oracle エンドポイントで SSL を使用するには、.pem 証明書ファイルの代わりにエンドポイント用の Oracle ウォレットをアップロードします。

トピック

- Oracle SSL への既存の証明書の使用
- Oracle SSL への自己署名証明書の使用

Oracle SSL への既存の証明書の使用

既存の Oracle クライアントインストールを使用して CA 証明書ファイルから Oracle ウォレットファイルを作成するには、以下の手順を実行します。

AWS DMSで Oracle SSL に既存の Oracle クライアントインストールを使用するには

1. 以下のコマンドを実行して、ORACLE_HOME システム変数を dbhome_1 ディレクトリの場所に設定します。

```
prompt>export ORACLE_HOME=/home/user/app/user/product/12.1.0/dbhome_1
```

2. $ORACLE_HOME/lib を LD_LIBRARY_PATH システム変数に追加します。

```
prompt>export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
```

3. $ORACLE_HOME/ssl_wallet に Oracle Wallet のディレクトリを作成します。

```
prompt>mkdir $ORACLE_HOME/ssl_wallet
```

4. CA 証明書ファイル .pem を ssl_wallet ディレクトリに配置します。Amazon RDS を使用する場合は、Amazon RDS によってホストされる rds-ca-2015-root.pem ルート CA 証明書ファイルをダウンロードできます。このファイルのダウンロード方法については、Amazon RDS ユーザーガイド の「SSL/TLS を使用した DB インスタンス接続の暗号化」をご参照ください。
5. 以下のコマンドを実行して Oracle Wallet を作成します。

```
prompt>orapki wallet create -wallet $ORACLE_HOME/ssl_wallet -auto_login_only
prompt>orapki wallet add -wallet $ORACLE_HOME/ssl_wallet -trusted_cert -cert
   $ORACLE_HOME/ssl_wallet/ca-cert.pem -auto_login_only
```

ここまでの手順を完了したら、ImportCertificate API コールで certificate-wallet パラメータを指定してウォレットファイルをインポートできます。Oracle エンドポイントの作成または変更時に SSL モードとして verify-ca を選択すると、インポートされたウォレット証明書を使用できます。

> **Note**
>
> Oracle ウォレットはバイナリファイルです。 AWS DMS はこれらのファイルをそのまま受け入れます。

Oracle SSL への自己署名証明書の使用

Oracle SSL に自己署名証明書を使用するには、oracle123 の Oracle ウォレットパスワードを次のように仮定して、次のステップを実行します。

で Oracle SSL の自己署名証明書を使用するには AWS DMS

1. 自己署名証明書で使用するディレクトリを作成します。

```
mkdir -p /u01/app/oracle/self_signed_cert
```

2. 前の手順で作成したディレクトリに移動します。

```
cd /u01/app/oracle/self_signed_cert
```

3. ルートキーを作成します。

```
openssl genrsa -out self-rootCA.key 2048
```

4. 前の手順で作成したルートキーを使用して、ルート証明書に自己署名します。

```
openssl req -x509 -new -nodes -key self-rootCA.key
        -sha256 -days 3650 -out self-rootCA.pem
```

次のような、入力パラメーターを使用します。

- `Country Name (2 letter code) [XX]`。例: `AU`
- `State or Province Name (full name) []`。例: `NSW`
- `Locality Name (e.g., city) [Default City]`。例: `Sydney`
- `Organization Name (e.g., company) [Default Company Ltd]`。例: `AmazonWebService`
- `Organizational Unit Name (e.g., section) []`。例: `DBeng`
- `Common Name (e.g., your name or your server's hostname) []`。例: `aws`
- `Email Address []`、例えば:abcd.efgh@amazonwebservice.com

5. Oracle データベース用の Oracle ウォレットディレクトリを作成します。

```
mkdir -p /u01/app/oracle/wallet
```

6. 新しい Oracle ウォレットを作成します。

```
orapki wallet create -wallet "/u01/app/oracle/wallet" -pwd oracle123 -
auto_login_local
```

7. Oracle ウォレットにルート証明書を追加します。

```
orapki wallet add -wallet "/u01/app/oracle/wallet" -pwd oracle123 -trusted_cert
-cert /u01/app/oracle/self_signed_cert/self-rootCA.pem
```

8. Oracle ウォレットの内容のリストを表示します。リストにはルート証明書が含まれます。

```
orapki wallet display -wallet /u01/app/oracle/wallet -pwd oracle123
```

たとえば、次のような表示となることがあります。

```
Requested Certificates:
User Certificates:
Trusted Certificates:
Subject:        CN=aws,OU=DBeng,O= AmazonWebService,L=Sydney,ST=NSW,C=AU
```

9. ORAPKI ユーティリティを使用して証明書署名リクエスト (CSR) を生成します。

```
orapki wallet add -wallet "/u01/app/oracle/wallet" -pwd oracle123
-dn "CN=aws" -keysize 2048 -sign_alg sha256
```

10. 以下のコマンドを実行します。

```
openssl pkcs12 -in /u01/app/oracle/wallet/ewallet.p12 -nodes -out /u01/app/oracle/
wallet/nonoracle_wallet.pem
```

これにより次のような出力が生成されます。

```
Enter Import Password:
MAC verified OK
Warning unsupported bag type: secretBag
```

11. 共通名として「dms」を指定します。

```
openssl req -new -key /u01/app/oracle/wallet/nonoracle_wallet.pem -out certdms.csr
```

次のような、入力パラメーターを使用します。

- Country Name (2 letter code) [XX]。例: AU
- State or Province Name (full name) []。例: NSW
- Locality Name (e.g., city) [Default City]。例: Sydney
- Organization Name (e.g., company) [Default Company Ltd]。例: AmazonWebService
- Organizational Unit Name (e.g., section) []。例: aws

- Common Name (e.g., your name or your server's hostname) []。例: aws
- Email Address []、例えば:abcd.efgh@amazonwebservice.com

これがステップ 4 と同じでないことを確認してください。たとえば、組織単位名を図のように別の名前に変更することで、これを行うことができます。

証明書要求とともに送信する追加属性を入力します。

- A challenge password []。例: oracle123
- An optional company name []。例: aws

12. 証明書の署名を取得します。

```
openssl req -noout -text -in certdms.csr | grep -i signature
```

この投稿の署名キーは sha256WithRSAEncryption となります。

13. 次のコマンドを実行して、証明書 (.crt) ファイルを生成します。

```
openssl x509 -req -in certdms.csr -CA self-rootCA.pem -CAkey self-rootCA.key
-CAcreateserial -out certdms.crt -days 365 -sha256
```

これにより次のような出力が生成されます。

```
Signature ok
subject=/C=AU/ST=NSW/L=Sydney/O=awsweb/OU=DBeng/CN=aws
Getting CA Private Key
```

14. 証明書をウォレットに追加します。

```
orapki wallet add -wallet /u01/app/oracle/wallet -pwd oracle123 -user_cert -cert
 certdms.crt
```

15. ウォレットを見る。エントリが 2 つあるはずです。次のコードが表示されます。

```
orapki wallet display -wallet /u01/app/oracle/wallet -pwd oracle123
```

16. sqlnet.ora ファイル ($ORACLE_HOME/network/admin/sqlnet.ora) を設定します。

```
WALLET_LOCATION =
```

```
   (SOURCE =
     (METHOD = FILE)
     (METHOD_DATA =
       (DIRECTORY = /u01/app/oracle/wallet/)
     )
   )

SQLNET.AUTHENTICATION_SERVICES = (NONE)
SSL_VERSION = 1.0
SSL_CLIENT_AUTHENTICATION = FALSE
SSL_CIPHER_SUITES = (SSL_RSA_WITH_AES_256_CBC_SHA)
```

17. Oracle リスナーを停止します。

```
lsnrctl stop
```

18. listener.ora ファイル ($ORACLE_HOME/network/admin/listener.ora) に SSL のエントリを追加します。

```
SSL_CLIENT_AUTHENTICATION = FALSE
WALLET_LOCATION =
  (SOURCE =
    (METHOD = FILE)
    (METHOD_DATA =
      (DIRECTORY = /u01/app/oracle/wallet/)
    )
  )

SID_LIST_LISTENER =
 (SID_LIST =
  (SID_DESC =
   (GLOBAL_DBNAME = SID)
   (ORACLE_HOME = ORACLE_HOME)
   (SID_NAME = SID)
  )
 )

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = localhost.localdomain)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCPS)(HOST = localhost.localdomain)(PORT = 1522))
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
```

```
    )
  )
```

19. tnsnames.ora ファイル ($ORACLE_HOME/network/admin/tnsnames.ora) を設定します。

```
<SID>=
(DESCRIPTION=
        (ADDRESS_LIST =
                (ADDRESS=(PROTOCOL = TCP)(HOST = localhost.localdomain)(PORT =
 1521))
        )
        (CONNECT_DATA =
                (SERVER = DEDICATED)
                (SERVICE_NAME = <SID>)
        )
)

<SID>_ssl=
(DESCRIPTION=
        (ADDRESS_LIST =
                (ADDRESS=(PROTOCOL = TCPS)(HOST = localhost.localdomain)(PORT =
 1522))
        )
        (CONNECT_DATA =
                (SERVER = DEDICATED)
                (SERVICE_NAME = <SID>)
        )
)
```

20. Oracle リスナーを再起動します。

```
lsnrctl start
```

21. Oracle リスナーの状態を表示します。

```
lsnrctl status
```

22. sqlplus と SSL tnsnames エントリを使用して、localhost からデータベースへの SSL 接続をテストします。

```
sqlplus -L ORACLE_USER@SID_ssl
```

23. SSL を使用して正常に接続したことを確認します。

```
SELECT SYS_CONTEXT('USERENV', 'network_protocol') FROM DUAL;

SYS_CONTEXT('USERENV','NETWORK_PROTOCOL')
--------------------------------------------------------------------------------
tcps
```

24. 現在のディレクトリを自己署名証明書のあるディレクトリに変更します。

```
cd /u01/app/oracle/self_signed_cert
```

25. AWS DMS が使用する新しいクライアント Oracle ウォレットを作成します。

```
orapki wallet create -wallet ./ -auto_login_only
```

26. Oracle ウォレットに自己署名証明書を追加します。

```
orapki wallet add -wallet ./ -trusted_cert -cert self-rootCA.pem -auto_login_only
```

27. AWS DMS が使用する Oracle ウォレットの内容を一覧表示します。リストには自己署名証明書が含まれます。

```
orapki wallet display -wallet ./
```

これにより次のような出力が生成されます。

```
Trusted Certificates:
Subject:        CN=aws,OU=DBeng,O=AmazonWebService,L=Sydney,ST=NSW,C=AU
```

28. 先ほど作成した Oracle ウォレットを にアップロードします AWS DMS。

のソースとして Oracle を使用するためのサポートされている暗号化方法 AWS DMS

次の表に、Oracle ソースデータベースを使用する際に が AWS DMS サポートする透過的データ暗号化 (TDE) メソッドを示します。

REDO ログのアクセス方法	TDE テーブルスペース	TDE 列
Oracle LogMiner	はい	はい

REDO ログのアクセス方法	TDE テーブルスペース	TDE 列
Binary Reader	はい	はい

AWS DMS は、バイナリリーダーを使用する場合、列レベルとテーブルスペースレベルの両方で Oracle TDE をサポートします。で TDE 暗号化を使用するには AWS DMS、まず TDE 暗号化キーと TDE パスワードが保存されている Oracle ウォレットの場所を特定します。次に、Oracle ソース エンドポイントの正しい TDE 暗号化キーとパスワードを特定します。

TDE 暗号化の暗号化キーとパスワードを識別して指定するには

1. 次のクエリを実行して、Oracle データベースホスト上の Oracle 暗号化ウォレットを検索します。

   ```
   SQL> SELECT WRL_PARAMETER FROM V$ENCRYPTION_WALLET;

   WRL_PARAMETER
   --------------------------------------------------------------------------------
   /u01/oracle/product/12.2.0/dbhome_1/data/wallet/
   ```

   ここでは /u01/oracle/product/12.2.0/dbhome_1/data/wallet/ がウォレットの場所です。

2. この値を返すものに応じて、次のいずれかの暗号化オプションを使用してマスター キー ID を取得します。

   a. テーブルレベルまたは列レベルの暗号化の場合は、次のクエリを実行します。

      ```
      SQL> SELECT OBJECT_ID FROM ALL_OBJECTS
      WHERE OWNER='DMS_USER' AND OBJECT_NAME='TEST_TDE_COLUMN' AND
       OBJECT_TYPE='TABLE';

      OBJECT_ID
      ---------------
      81046
      SQL> SELECT MKEYID FROM SYS.ENC$ WHERE OBJ#=81046;

      MKEYID
      ------------
      AWGDC9glSk8Xv+3bVveiVSgAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      ```

ここでは AWGDC9glSk8Xv+3bVveiVSg がマスターキー ID (MKEYID) です。MKEYID の値が得られたら、ステップ 3 に進むことができます。それ以外の場合は、ステップ 2.2 に進みます。

> **Note**
>
> 末尾の文字列 'A' 文字 (AAA...) は値の一部ではありません。

b. テーブルスペース レベルの暗号化の場合は、次のクエリを実行します。

```
SQL> SELECT TABLESPACE_NAME, ENCRYPTED FROM dba_tablespaces;
TABLESPACE_NAME                ENC
------------------------------ ---
SYSTEM                         NO
SYSAUX                         NO
UNDOTBS1                       NO
TEMP                           NO
USERS                          NO
TEST_ENCRYT                    YES
SQL> SELECT name,utl_raw.cast_to_varchar2( utl_encode.base64_encode('01'||
substr(mkeyid,1,4))) ||
 utl_raw.cast_to_varchar2( utl_encode.base64_encode(substr(mkeyid,5,length(mkeyid))))
 masterkeyid_base64
FROM (SELECT t.name, RAWTOHEX(x.mkid) mkeyid FROM v$tablespace t, x$kcbtek x
 WHERE t.ts#=x.ts#)
WHERE name = 'TEST_ENCRYT';

NAME                           MASTERKEYID_BASE64
------------------------------ ----------------------------------
TEST_ENCRYT                    AWGDC9glSk8Xv+3bVveiVSg=
```

ここでは AWGDC9glSk8Xv+3bVveiVSg がマスターキー ID (TEST_ENCRYT) です。ステップ 2.1 と 2.2 の両方が値を返す場合、それらは常に同一です。

末尾の '='文 字は値の一部ではありません。

3. コマンド・ラインから、ソース Oracle データベース ホスト上の暗号化ウォレット エントリをリストします。

```
$ mkstore -wrl /u01/oracle/product/12.2.0/dbhome_1/data/wallet/ -list
Oracle Secret Store entries:
```

```
ORACLE.SECURITY.DB.ENCRYPTION.AWGDC9glSk8Xv+3bVveiVSgAAAAAAAAAAAAAAAAAAAAAAAAAA
ORACLE.SECURITY.DB.ENCRYPTION.AY1mRA8OXU9Qvzo3idU4OH4AAAAAAAAAAAAAAAAAAAAAAAAAA
ORACLE.SECURITY.DB.ENCRYPTION.MASTERKEY
ORACLE.SECURITY.ID.ENCRYPTION.
ORACLE.SECURITY.KB.ENCRYPTION.
ORACLE.SECURITY.KM.ENCRYPTION.AY1mRA8OXU9Qvzo3idU4OH4AAAAAAAAAAAAAAAAAAAAAAAAAA
```

ステップ 2 (AWGDC9glSk8Xv+3bVveiVSg) で確認したマスター キー ID を含むエントリを検索します。このエントリは TDE 暗号化キー名です。

4. 前のステップで検索したエントリの詳細を表示します。

```
$ mkstore -wrl /u01/oracle/product/12.2.0/dbhome_1/data/wallet/ -viewEntry
 ORACLE.SECURITY.DB.ENCRYPTION.AWGDC9glSk8Xv+3bVveiVSgAAAAAAAAAAAAAAAAAAAAAAAAAA
Oracle Secret Store Tool : Version 12.2.0.1.0
Copyright (c) 2004, 2016, Oracle and/or its affiliates. All rights reserved.
Enter wallet password:
ORACLE.SECURITY.DB.ENCRYPTION.AWGDC9glSk8Xv+3bVveiVSgAAAAAAAAAAAAAAAAAAAAAAAAAA
 = AEMAASAASGYs0phWHfNt9J5mEMkkegGFiD4LLfQszDojgDzbfoYDEACv0x3pJC+UGD/
PdtE2jLIcBQcAeHgJChQGLA==
```

ウォレット パスワードを入力して結果を確認します。

ここで、'=' の右にある値は TDE パスワードです。

5. Oracle ソース エンドポイントの TDE 暗号化キー名を指定するには、securityDbEncryptionName 追加の接続属性を設定します。

```
securityDbEncryptionName=ORACLE.SECURITY.DB.ENCRYPTION.AWGDC9glSk8Xv
+3bVveiVSgAAAAAAAAAAAAAAAAAAAAAAAAAA
```

6. Oracle ソースパスワード値の一部としてコンソールでこのキーに関連付けられた TDE パスワードを入力します。パスワード値。TDE パスワード値で終わるコンマ区切りのパスワード値をフォーマットするには、次の順序を使用します。

```
Oracle_db_password,ASM_Password,AEMAASAASGYs0phWHfNt9J5mEMkkegGFiD4LLfQszDojgDzbfoYDEACv0x3
+UGD/PdtE2jLIcBQcAeHgJChQGLA==
```

Oracle データベースの設定に関係なく、この順序でパスワード値を指定します。たとえば、TDE を使用していて Oracle データベースで ASM を使用していない場合、関連するパスワード値をカンマで区切って次の順序で指定します。

```
Oracle_db_password,,AEMAASAASGYs0phWHfNt9J5mEMkkegGFiD4LLfQszDojgDzbfoYDEACv0x3pJC
+UGD/PdtE2jLIcBQcAeHgJChQGLA==
```

指定した TDE 認証情報が正しくない場合、 AWS DMS 移行タスクは失敗しません。ただし、このタスクでは、進行中のレプリケーションの変更をターゲット データベースに対して読み取りまたは適用しません。タスクを開始したら、コンソール移行タスクページでテーブル統計を監視し、変更レプリケーションがなされていることを確認します。

タスクの実行中に DBA によって Oracle データベースの TDE 認証情報値が変更された場合、タスクは失敗します。エラーメッセージには、新しい TDE 暗号化キー名が含まれています。新しい値を指定してタスクを再開するには、前の手順を使用します。

Important

OS レベルのcp、mv、orapki や mkstoreなどのコマンドではASM の場所に保存されているウォレットファイルが破損するため、Oracle 自動ストレージ管理(ASM)の場所で作成された TDE ウォレットを操作することはできません。この制限は、ASM の場所に格納された TDE ウォレット ファイルにのみに固有であり、ローカル OS ディレクトリに格納されている TDE ウォレット ファイルは対象外です。
ASM に格納されている TDE ウォレットを OS レベルのコマンドで操作するには、ローカル キーストアを作成し、次のように ASM キーストアをローカル キーストアにマージします：

1. ローカル キーストアを作成します。

```
ADMINISTER KEY MANAGEMENT create keystore file system wallet location
 identified by wallet password;
```

2. ASM キーストアをローカル キーストアにマージします。

```
ADMINISTER KEY MANAGEMENT merge keystore ASM wallet location identified
 by wallet password into existing keystore file system wallet location
 identified by wallet password with backup;
```

次に、暗号化ウォレット エントリと TDE パスワードを一覧表示するには、ローカル キーストアに対してステップ 3 と 4 を実行します。

のソースとして Oracle を使用するためのサポートされている圧縮方法 AWS DMS

次の表に、Oracle ソースデータベースを使用する際に が AWS DMS サポートする圧縮方法を示します。表に示すように、圧縮のサポートは、Oracle データベースのバージョンと、DMS が Oracle を使用して REDO ログ LogMiner にアクセスするように設定されているかどうかの両方によって異なります。

Version	Basic (ベーシック)	OLTP	HCC (Oracle 11g R2 より)	その他
Oracle 10	いいえ	該当なし	該当なし	いいえ
Oracle 11 以降 – Oracle LogMiner	はい	はい	はい	はい — Oracle でサポートされている圧縮方法 LogMiner。
オラクル 11 以降 — Binary Reader	はい	はい	[Yes](はい) - 詳細については、次の注をご参照ください。	はい

Note

Oracle ソースエンドポイントが Binary Reader を使用するように設定されている場合、HCC 圧縮方法の Query Low レベルは、全ロードタスクに対してのみサポートされます。

のソースとして Oracle を使用してネストされたテーブルをレプリケートする AWS DMS

AWS DMS は、ネストされたテーブルまたは定義された型の列を含む Oracle テーブルのレプリケーションをサポートします。この機能を有効にするには、Oracle ソース エンドポイントに以下の接続属性設定を追加します。

```
allowSelectNestedTables=true;
```

AWS DMS は、Oracle ネストされたテーブルからターゲットテーブルを、一意の制約なしでターゲット上の通常の親テーブルと子テーブルとして作成します。ターゲットの正しいデータにアクセスするには、親テーブルと子テーブルを結合します。これを行うには、まず、ターゲットの子テーブルの NESTED_TABLE_ID 列に一意でないインデックスを手動で作成します。その後、結合 ON 句の NESTED_TABLE_ID 列を、子テーブル名に対応する親列とともに使用できます。さらに、このようなインデックスを作成すると、ターゲットの子テーブルデータが によって更新または削除されるときのパフォーマンスが向上します AWS DMS。例については、「ターゲットの親テーブルと子テーブルの結合の例」をご参照ください。

完全なロードが完了したら、タスクを停止するように設定することをお勧めします。次に、ターゲット上のすべてのレプリケートされた子テーブルに対して一意でないインデックスを作成し、タスクを再開します。

キャプチャされたネストされたテーブルが既存の親テーブルに追加された場合 (キャプチャされたかキャプチャされなかったか)、 はそれを正しく AWS DMS 処理します。ただし、対応するターゲットテーブルの一意でないインデックスは作成されません。この場合、ターゲットの子テーブルが非常に大きくなると、パフォーマンスが影響を受ける可能性があります。このような場合は、タスクを停止し、インデックスを作成してからタスクを再開することをお勧めします。

ネストされたテーブルがターゲットにレプリケートされたら、DBA は親テーブルと対応する子テーブルに対して結合を実行し、データを平坦化します。

Oracle ネストされたテーブルをソースとしてレプリケートするための前提条件

レプリケートされた、すべてのネストされたテーブルについて、親テーブルをレプリケートしてください。テーブル AWS DMS マッピングには、親テーブル (ネストされたテーブル列を含むテーブル) と子テーブル (つまり、ネストされた) の両方を含めます。

ソースとしてサポートされている Oracle ネストされたテーブル型

AWS DMS は、ソースとして次の Oracle ネストされたテーブルタイプをサポートします。

- データ型
- ユーザー定義のオブジェクト

ソースとしての Oracle ネストされたテーブルの AWS DMS サポートの制限

AWS DMS Oracle ネストされたテーブルをソースとしてサポートする場合、 には以下の制限があります。

- AWS DMS は、1 レベルのテーブルネストのみをサポートします。
- AWS DMS テーブルマッピングでは、親テーブルと子テーブルの両方がレプリケーション対象として選択されているかどうかはチェックされません。つまり、子のない親テーブル、または親のない子テーブルを選択する可能性があります。

AWS DMS が Oracle ネストされたテーブルをソースとしてレプリケートする方法

AWS DMS は、次のように親テーブルとネストされたテーブルをターゲットにレプリケートします。

- AWS DMS はソースと同じ親テーブルを作成します。次に、親のネストされた列を RAW(16) として定義し、親のネストされたテーブルへの参照をその NESTED_TABLE_ID 列に含めます。
- AWS DMS は、ネストされたソースと同じ子テーブルを作成しますが、 という名前の列を追加しますNESTED_TABLE_ID。この列は、対応する親のネストされた列と同じ型と値を持ち、同じ意味を持ちます。

ターゲットの親テーブルと子テーブルの結合の例

親テーブルを平坦化するには、次の例に示すように、親テーブルと子テーブルの間で結合を実行します。

1. Type テーブルを作成します。

```
CREATE OR REPLACE TYPE NESTED_TEST_T AS TABLE OF VARCHAR(50);
```

2. 前述のように型 NESTED_TEST_T 列を持つ親テーブルを作成します。

```
CREATE TABLE NESTED_PARENT_TEST (ID NUMBER(10,0) PRIMARY KEY, NAME NESTED_TEST_T)
 NESTED TABLE NAME STORE AS NAME_KEY;
```

3. CHILD.NESTED_TABLE_ID が PARENT.NAME と一致する NAME_KEY 子テーブルとの結合を使用してテーブル NESTED_PARENT_TEST を平坦化します。

```
SELECT … FROM NESTED_PARENT_TEST PARENT, NAME_KEY CHILD WHERE CHILD.NESTED_
TABLE_ID = PARENT.NAME;
```

のソースとして Oracle を使用する場合の Oracle ASM への REDO の保存 AWS DMS

REDO 生成が多い Oracle ソースの場合、REDO を Oracle ASM に保存すると、ASM REDO 読み取りをすべての ASM ノードに分散するように DMS を設定できるため、特に RAC 構成の場合にパフォーマンスが向上します。

この構成を利用するには、asmServer 接続属性を使用します。例えば、以下の接続文字列では、DMS REDO 読み取りが 3 つの ASM ノードに分散されます。

```
asmServer=(DESCRIPTION=(CONNECT_TIMEOUT=8)(ENABLE=BROKEN)(LOAD_BALANCE=ON)(FAILOVER=ON)
(ADDRESS_LIST=
(ADDRESS=(PROTOCOL=tcp)(HOST=asm_node1_ip_address)(PORT=asm_node1_port_number))
(ADDRESS=(PROTOCOL=tcp)(HOST=asm_node2_ip_address)(PORT=asm_node2_port_number))
(ADDRESS=(PROTOCOL=tcp)(HOST=asm_node3_ip_address)(PORT=asm_node3_port_number)))
(CONNECT_DATA=(SERVICE_NAME=+ASM)))
```

NFS を使用して Oracle REDO を保存する場合、適切な DNFS (ダイレクト NFS) クライアントパッチ、特に Oracle のバグ 25224242 に対処するパッチが適用されていることを確認することが重要です。詳細については、Direct NFS クライアント関連のパッチに関する「Recommended Patches for Direct NFS Client」を参照してください。

さらに、NFS の読み取りパフォーマンスを向上するには、次の例のとおり NFS fstab ボリュームの rsize と wsize in の値を増やすことをお勧めします。

```
NAS_name_here:/ora_DATA1_archive /u09/oradata/DATA1 nfs
 rw,bg,hard,nointr,tcp,nfsvers=3,_netdev,
timeo=600,rsize=262144,wsize=262144
```

また、tcp-max-xfer-size値を次のとおり調整します。

```
vserver nfs modify -vserver vserver -tcp-max-xfer-size 262144
```

のソースとして Oracle を使用する場合のエンドポイント設定 AWS DMS

エンドポイントの設定を使用して、追加の接続属性の使用する場合と同様に、ソースの Oracle のデータベースを設定できます。ソースエンドポイントを作成するときはAWS CLI、 AWS DMS コンソールを使用するか、`--oracle-settings '{"EndpointSetting": "value", ...}'`JSON 構文で の `create-endpoint` コマンドを使用して設定を指定します。

次の表は、Oracle をソースとして使用できるエンドポイント設定を説明しています。

名前	説明
`AccessAlternateDirectly`	ソースとして Amazon RDS for Oracle の変更データを Binary Reader でキャプチャするには、この属性を false に設定します。この設定は、DMS インスタンスに対して、指定されたパスプレフィックス置換を使用してファイルへの直接アクセスで REDO ログにアクセスしないように指示します。詳細については、「 の RDS for Oracle ソースで Binary Reader を使用するように CDC タスクを設定する AWS DMS」を参照してください。 デフォルト値: true 有効な値: true/false 例: `--oracle-settings '{"AccessAlternateDirectly": false}'`
`AdditionalArchivedLogDestId`	この属性を、プライマリ スタンバイ セットアップで `ArchivedLogDestId` と設定します。この属性は、Oracle Data Guard データベースをソースとして使用する場合のスイッチオーバーに役立ちます。この場合、 は、変更を読み取るためにアーカイブ REDO ログを取得する送信先を知る AWS DMS 必要があります。これは、スイッチオーバー後、前のプライマリがスタンバイインスタンスになるためです。 AWS DMS では Oracle `RESETLOGS` オプションを使用してデータベースを開くことができますが、必要`RESETLOGS` がない限り を使用しないでください。`RESETLOGS` に関する追加情報については Oracle® データベース Backup およびリカバリ ユーザーガイドの「RMANデータ修復の概念」をご参照ください。

名前	説明
	有効値: アーカイブ先 ID 例: `--oracle-settings '{"AdditionalArchivedLogDestId": 2}'`
`AddSupplementalLogging`	Oracle データベースにテーブルレベルのサプリメンタルロギングをセットアップするには、この属性を設定します。この属性は、テーブルのメタデータに応じて、移行タスク用に選択したすべてのテーブルで次のいずれかを有効にします。 • PRIMARY KEY COLUMNS サプリメンタルロギング • UNIQUE KEY COLUMNS サプリメンタルロギング • ALL COLUMNS サプリメンタルロギング デフォルト値: false 有効な値: true/false 例: `--oracle-settings '{"AddSupplementalLogging": false}'` **Note** このオプションを使用する場合でも、前述のようにデータベースレベルサプリメンタルロギングを有効にする必要があります。

名前	説明
AllowSelectNestedTables	ネストされたテーブルまたは定義された型である列を含む Oracle テーブルのレプリケーションを有効にするには、この属性を true に設定します。詳細については、「のソースとして Oracle を使用してネストされたテーブルをレプリケートする AWS DMS」を参照してください。 デフォルト値: false 有効な値: true/false 例: `--oracle-settings '{"AllowSelectNestedTables": true}'`
ArchivedLogDestId	アーカイブされた REDO ログの宛先 ID を指定します。この値は、v$archived_log ビューの dest_id 列の数値と同じにする必要があります。追加の REDO ログの保存先を使用する場合は、AdditionalArchivedLogDestId 属性を使用して、追加の宛先 ID を指定するようお勧めします。これにより、最初から適切なログにアクセスされるため、パフォーマンスが向上します。 デフォルト値： 1 有効な値: 数値 例: `--oracle-settings '{"ArchivedLogDestId": 1}'`
ArchivedLogsOnly	このフィールドが Y に設定されている場合、 はアーカイブされた REDO ログ AWS DMS にのみアクセスします。アーカイブされた REDO ログが Oracle ASM にのみ保存されている場合、 AWS DMS ユーザーアカウントに ASM 権限を付与する必要があります。 デフォルト値: N 有効な値: Y/N 例: `--oracle-settings '{"ArchivedLogsOnly": Y}'`

名前	説明
asmUsePLSQLArray (ECA のみ)	でソースの変更をキャプチャするときは、この追加の接続属性 (ECA) を使用します BinaryReader。この設定により、DMS は parallelASMReadThread 属性を使用してスレッド数を制御しながら、単一の読み取りスレッドごとに ASM レベルで 50 件の読み取りをバッファできるようになる。この属性を設定すると、 AWS DMS バイナリリーダーは匿名の PL/SQL ブロックを使用して REDO データをキャプチャし、大きなバッファとしてレプリケーションインスタンスに送り返します。これにより、ソースへの往復回数が低減する。これにより、ソースキャプチャのパフォーマンスは大幅に向上するとはいえ、ASM インスタンスの PGA メモリ消費量が増える。メモリターゲットが十分でない場合、安定性の問題が発生する可能性がある。次の式を使用して、単一の DMS タスクによる ASM インスタンスの PGA メモリの使用量合計を見積もることができる。number_of_redo_threads * parallelASMReadThreads * 7 MB デフォルト値: false 有効な値: true/false ECA の例:asmUsePLSQLArray=true;
ConvertTimestampWithZoneToUTC	TIMESTAMP WITH TIME ZONE 列と TIMESTAMP WITH LOCAL TIME ZONE 列のタイムスタンプ値を UTC に変換するには、この属性を true に設定する。デフォルトでは、この属性の値は「false」で、データはソースデータベースのタイムゾーンを使用してレプリケートされる。 デフォルト値: false 有効な値: true/false 例: --oracle-settings '{"ConvertTimestampWithZoneToUTC": true}'

名前	説明
EnableHomogenousPartitionOps	この属性を true に設定すると、Oracle の同種移行のための Oracle パーティションとサブパーティションの DDL 操作をレプリケートできる。 この機能はバージョン 3 AWS DMS .4.7 で導入されたことに注意してください。 デフォルト値: false 有効な値: true/false 例: `--oracle-settings '{"EnableHomogenousPartitionOps": true}'`
EnableHomogenousTablespace	同種のテーブルスペースのレプリケーションを有効にし、ターゲットの同じテーブルスペースで既存のテーブルまたはインデックスを作成するには、この属性を設定します。 デフォルト値: false 有効な値: true/false 例: `--oracle-settings '{"EnableHomogenousTablespace": true}'`

名前	説明
EscapeCharacter	この属性はエスケープ文字に設定する。このエスケープ文字を使うと、単一のワイルドカード文字をテーブルマッピング式で通常の文字のように動作させることができる。詳細については、「テーブルマッピングのワイルドカード」を参照してください。 デフォルト値: Null 有効値: ワイルドカード文字以外の任意の文字 例: `--oracle-settings '{"EscapeCharacter": "#"}'` **Note** escapeCharacter はテーブル名にのみ使用できる。スキーマ名や列名でエスケープ文字は使用できない。
ExposeViews	この属性を使ってビューからデータを 1 回プルできます。これを継続的なレプリケーションに使用することはできません。ビューからデータを抽出すると、そのビューはターゲットスキーマのテーブルとして表示されます。 デフォルト値: false 有効な値: true/false 例: `--oracle-settings '{"ExposeViews": true}'`

名前	説明
ExtraArchivedLogDestIds	1 つ以上のアーカイブされた REDO ログに対する 1 つ以上の送信先の ID を指定します。このような ID は、v$archived_log ビューの dest_id 列の値である。この設定は、セットアップまたは primary-to-single セットアップの追加 ArchivedLogDestId 接続属性で使用します primary-to-multiple-standby 。 この属性は、Oracle Data Guard データベースをソースとして使用するときの切り替えに役立ちます。この場合、 は変更を読み取るためにアーカイブ REDO ログを取得する送信先に関する情報 AWS DMS を必要とします。 AWS DMS これは、前のプライマリのスイッチオーバー後がスタンバイインスタンスであるためです。 有効値:アーカイブ先 ID 例: `--oracle-settings '{"ExtraArchivedLogDestIds": 1}'`
FailTasksOnLobTruncation	`true` に設定されると、LOB 列の実際のサイズが指定された `LobMaxSize` よりも大きい場合、この属性によりタスクは失敗します。 タスクが制限付き LOB モードに設定され、このオプションが `true` に設定されている場合、LOB データを切り捨てるのではなくタスクが失敗します。 デフォルト値: false 有効な値: ブール値 例: `--oracle-settings '{"FailTasksOnLobTruncation": true}'`

名前	説明
`filterTransactionsOfUser` (ECA のみ)	この追加の接続属性 (ECA) を使用して、 を使用するときに Oracle からデータをレプリケートするときに、DMS が指定されたユーザーからのトランザクションを無視できるようにします LogMiner。カンマ区切りのユーザー名の値を渡すことができる。ただし、すべて大文字にする必要がある。 ECA の例:`filterTransactionsOfUser=` *`USERNAME`*`;`
`NumberDataTypeScale`	数値のスケールを指定します。最大 38 のスケールを選択するか、FLOAT には -1、VARCHAR には -2 を選択できます。デフォルトでは、NUMBER データ型が精度 38、スケール 10 に変換されます。 デフォルト値: 10 有効な値: -2〜38 (VARCHAR では -2、FLOAT の場合は -1) 例: `--oracle-settings '{"NumberDataTypeScale": 12}'` **Note** 精度スケールの組み合わせ、-1 (FLOAT) または -2 (VARCHAR) を選択します。DMS は、Oracle がサポートする精度スケールの組み合わせをサポートします。精度が 39 以上の場合は、-2 (VARCHAR) を選択します。Oracle データベースの NumberDataTypeScale 設定は、NUMBER データ型にのみ使用されます (明示的な精度とスケール定義なし)。

名前	説明
OpenTransactionWindow	CDC のみのタスクでオープントランザクションを確認する時間枠を分単位で指定する。 **Note** OpenTransactionWindow を 1 以上に設定した場合、DMS は SCN_TO_TIMESTAMP を使用して SCN 値をタイムスタンプ値に変換します。Oracle Database の制限により、CDC 開始点として古すぎる SCN を指定すると、SCN_TO_TIMESTAMP はORA-08181 エラーで失敗し、CDC のみのタスクを開始することはできません。 デフォルト値: 0 有効値: 0 ~ 240 の整数 例: openTransactionWindow=15;
OraclePathPrefix	Amazon RDS for Oracle をソースとして Binary Reader で変更データをキャプチャするには、この文字列属性を必要な値に設定します。この値は、REDO ログにアクセスするために使用されるデフォルトの Oracle ルートを指定します。詳細については、「の RDS for Oracle ソースで Binary Reader を使用するように CDC タスクを設定する AWS DMS」を参照してください。 デフォルト値： なし 有効な値: /rdsdbdata/db/ORCL_A/ 例: --oracle-settings '{"OraclePathPrefix": "/rdsdbdata/db/ORCL_A/"}'

名前	説明
ParallelASMReadThreads	この属性を設定して、DMS が Oracle 自動ストレージ管理 (ASM) を使用して変更データ キャプチャ (CDC) を実行するように設定するスレッドの数を変更します。2 (デフォルト) から 8 (最大) までの整数値を指定できます。この属性は ReadAheadBlocks 属性とともに使用します。詳細については、「 の RDS for Oracle ソースで Binary Reader を使用するように CDC タスクを設定する AWS DMS」を参照してください。 デフォルト値: 2 有効な値: 2 ~ 8 の整数 例: `--oracle-settings '{"ParallelASMReadThreads": 6;}'`
ReadAheadBlocks	この属性を設定して、 Oracle Automatic Storage Management (ASM) と ASM 以外の NAS ストレージを使用して CDC を実行するように DMS が設定する先読みブロック数を変更する。1,000 (デフォルト) から 200,000 (最大) までの整数値を指定できます。この属性は ParallelASMReadThreads 属性とともに使用します。詳細については、「 の RDS for Oracle ソースで Binary Reader を使用するように CDC タスクを設定する AWS DMS」を参照してください。 デフォルト値: 1000 有効な値: 1,000 ~ 200,000 の整数 例: `--oracle-settings '{"ReadAheadBlocks": 150000}'`
ReadTableSpaceName	true に設定すると、この属性はテーブルスペースのレプリケーションをサポートします。 デフォルト値: false 有効な値: ブール値 例: `--oracle-settings '{"ReadTableSpaceName": true}'`

名前	説明
ReplacePathPrefix	Amazon RDS for Oracle をソースとして Binary Reader で変更データをキャプチャするには、この属性を true に設定します。この設定は、DMS インスタンスに対して、デフォルトの Oracle ルートを UsePathPrefix の設定に置換して REDO ログにアクセスするように指示します。詳細については、「の RDS for Oracle ソースで Binary Reader を使用するように CDC タスクを設定する AWS DMS」を参照してください。 デフォルト値: false 有効な値: true/false 例: `--oracle-settings '{"ReplacePathPrefix": true}'`
RetryInterval	システムがクエリを再送するまで待機する時間の秒数を指定します。 デフォルト値: 5 有効な値: 1 以降の数値 例: `--oracle-settings '{"RetryInterval": 6}'`
SecurityDbEncryptionName	Oracle ソースデータベースの列およびテーブルスペースの透過的データ暗号化 (TDE) に使用されるキーの名前を指定します。Oracle ソースエンドポイントでのこの属性とそれに関連付けられたパスワードの設定の詳細については、「のソースとして Oracle を使用するためのサポートされている暗号化方法 AWS DMS」をご参照ください。 デフォルト値: "" 有効な値: 文字列 例: `--oracle-settings '{"SecurityDbEncryptionName": "ORACLE.SECURITY.DB.ENCRYPTION.Adg8m2dhkU/0v/m5QUaaNJEAAAAAAAAAAAAAAAAAAAAAAAAAAAA"}'`

名前	説明
SpatialSdo2GeoJsonFunctionName	PostgreSQL ターゲットに移行する Oracle バージョン 12.1 以前のソースの場合、この属性を使用して SDO_GEOMETRY 形式を GEOJSON 形式に変換します。 デフォルトでは、 は AWS DMS ユーザーが存在し、アクセス可能である必要があるSDO2GEOJSON カスタム関数を AWS DMS 呼び出します。または、SDOGEOJSON のオペレーションを模倣する独自のカスタム関数を作成し、代わりにそれを呼び出すように SpatialSdo2GeoJsonFunctionName を設定することもできます。 デフォルト値: SDO2GEOJSON 有効な値: 文字列 例: `--oracle-settings '{"SpatialSdo2GeoJsonFunctionName": "myCustomSDO2GEOJSONFunction"}'`
StandbyDelayTime	この属性を使用して、スタンバイ同期の遅延時間を分単位で指定します。ソースが Active Data Guard スタンバイデータベースの場合、この属性を使用して、プライマリ データベースとスタンバイ データベース間のタイムラグを指定します。 では AWS DMS、進行中の変更をレプリケートするためのソースとして Active Data Guard スタンバイインスタンスを使用する Oracle CDC タスクを作成できます。これにより、運用中のアクティブなデータベースに接続する必要がなくなります。 デフォルト値: 0 有効な値: 数値 例: `--oracle-settings '{"StandbyDelayTime": 1}'` 注: DMS 3.4.6、3.4.7 以降を使用している場合、この接続設定の使用は任意となる。DMS 3.4.6 とバージョン 3.4.7 の最新バージョンでは、DMS がスタンバイ遅延時間を計算できるように*dms_user* に V_$DATAGUARD_STATS に対する select アクセス権限が必要となる。

名前	説明
UseAlternateFolderForOnline	Amazon RDS for Oracle をソースとして Binary Reader で変更データをキャプチャするには、この属性を true に設定します。この設定は、DMS インスタンスに対して、指定されたプレフィックス置換を使用してすべてのオンライン REDO ログにアクセスするように指示します。詳細については、「 の RDS for Oracle ソースで Binary Reader を使用するように CDC タスクを設定する AWS DMS」を参照してください。 デフォルト値: false 有効な値: true/false 例: `--oracle-settings '{"UseAlternateFolderForOnline": true}'`
UseBfile	Binary Reader ユーティリティを使用して変更データをキャプチャするには、この属性を Y に設定します。この属性を Y に設定するため、UseLogminerReader を N に設定します。また、 Amazon RDS for Oracle でソースとして Binary Reader を使用するには、追加の属性を設定します。この設定および Oracle 自動ストレージ管理 (ASM) の使用の詳細については、CDC での Oracle LogMiner または AWS DMS Binary Reader の使用 を参照してください 注: この値を追加の接続属性 (ECA) として設定する場合の有効な値は「Y」と「N」となる。この値をエンドポイント設定として設定する場合、有効な値は true と false となる。 デフォルト値: N 有効値: Y または N (この値を ECA として設定する場合)、true または false (この値をエンドポイント設定として設定する場合) 例: `--oracle-settings '{"UseBfile": Y}'`

名前	説明
UseLogminerReader	LogMiner ユーティリティ (デフォルト) を使用して変更データをキャプチャするには、この属性を Y に設定します。 AWS DMS がバイナリファイルとして REDO ログにアクセスするようにする場合は、このオプションを N に設定します。このオプションを N に設定すると、useBfile=Y という設定も追加されます。この設定および Oracle 自動ストレージ管理 (ASM) の使用の詳細については、「CDC での Oracle LogMiner または AWS DMS Binary Reader の使用」をご参照ください。 注: この値を追加の接続属性 (ECA) として設定する場合の有効な値は「Y」と「N」となる。この値をエンドポイント設定として設定する場合、有効な値は `true` と `false` となる。 デフォルト値: Y 有効値: Y または N (この値を ECA として設定する場合)、true または false (この値をエンドポイント設定として設定する場合) 例: `--oracle-settings '{"UseLogminerReader": Y}'`
UsePathPrefix	Amazon RDS for Oracle をソースとして Binary Reader で変更データをキャプチャするには、この文字列属性を必要な値に設定します。この値は、デフォルトの Oracle ルートを置換して REDO ログにアクセスするために使用されるパスプレフィックスを指定します。詳細については、「 の RDS for Oracle ソースで Binary Reader を使用するように CDC タスクを設定する AWS DMS」を参照してください。 デフォルト値： なし 有効な値: /rdsdbdata/log/ 例: `--oracle-settings '{"UsePathPrefix": "  /rdsdbdata/log/"}'`

Oracle のソースデータ型

の Oracle エンドポイントは、ほとんどの Oracle データ型 AWS DMS をサポートしています。次の表は、 の使用時にサポートされる Oracle ソースデータ型 AWS DMS と、 AWS DMS データ型へのデフォルトのマッピングを示しています。

> **Note**
>
> LONG と LONG RAW データ型を除き、Oracle ソースから Oracle ターゲットにレプリケーションする場合 (均質なレプリケーション)、ソースとターゲットのすべてのデータ型が同一になります。ただし、LONG データ型は CLOB にマッピングされ、LONG RAW データ型は BLOB にマップされます。

ターゲットにマッピングされるデータ型を表示する方法については、使用しているターゲットエンドポイントのセクションをご参照ください。

AWS DMS データ型の詳細については、「」を参照してくださいAWS Database Migration Service のデータ型。

Oracle のデータ型	AWS DMS データ型
BINARY_FLOAT	REAL4
BINARY_DOUBLE	REAL8
BINARY	BYTES
FLOAT (P)	精度が 24 以下の場合、REAL4 を使用します。 精度が 24 より高い場合、REAL8 を使用します。
NUMBER (P,S)	位取りが 0 より大きい場合、NUMERIC を使用する。 スケールが 0 の場合 • 精度が 2 以下の場合、INT1 を使用します。 • 精度が 2 より大きく 4 以下の場合、INT2 を使用します。 • 精度が 4 より大きく 9 以下の場合、INT4 を使用します。 • 精度が 9 より大きい場合、NUMERIC を使用します。

Oracle のデータ型	AWS DMS データ型
	• 精度がスケール以上の場合、NUMERIC を使用します。 位取りが 0 未満の場合、REAL8 を使用する。
DATE	DATETIME
INTERVAL_YEAR TO MONTH	STRING (間隔 year_to_month を指定)
INTERVAL_DAY TO SECOND	STRING (間隔 day_to_second を指定)
TIMESTAMP	DATETIME
タイムスタンプ時間帯あり	STRING (timestamp_with_timezone を指定)
タイムスタンプ現地時間帯あり	STRING (timestamp_with_local_ timezone を指定)
CHAR	STRING
VARCHAR2	STRING
NCHAR	WSTRING
NVARCHAR2	WSTRING
RAW	BYTES
REAL	REAL8
BLOB	BLOB このデータ型を で使用するには AWS DMS、特定のタスクで BLOB データ型の使用を有効にする必要があります。 は、プライマリキーを含むテーブルでのみ BLOB データ型 AWS DMS をサポートします。

Oracle のデータ型	AWS DMS データ型
CLOB	CLOB このデータ型を で使用するには AWS DMS、特定のタスクで CLOB データ型の使用を有効にする必要があります。CDC 中、 はプライマリキーを含むテーブルでのみ CLOB データ型 AWS DMS をサポートします。
NCLOB	NCLOB このデータ型を で使用するには AWS DMS、特定のタスクで NCLOB データ型の使用を有効にする必要があります。CDC 中、はプライマリキーを含むテーブルでのみ NCLOB データ型 AWS DMS をサポートします。
LONG	CLOB LONG データ型は、バッチ最適化適用モード (TurboStream CDC モード) ではサポートされていません。 このデータ型を で使用するには AWS DMS、特定のタスクで LOBs の使用を有効にします。 CDC または全ロード中、 はプライマリキーを持つテーブルでのみ LOB データ型 AWS DMS をサポートします。 また、LONG 列をロードするためのフル LOB モードは AWS DMS サポートされていません。この代わりに、制限付き LOB モードを使用して LONG 列を Oracle ターゲットに移行できる。限定 LOB モードでは、 は、64 KB を超える LONG 列に設定したすべてのデータを 64 KB に AWS DMS 切り捨てます。での LOB サポートの詳細については AWS DMS、「」を参照してください。 AWS DMS タスクでのソースデータベースの LOB サポートの設定

Oracle のデータ型	AWS DMS データ型
LONG RAW	BLOB LONG RAW データ型は、バッチ最適化適用モード (TurboStream CDC モード) ではサポートされていません。 このデータ型を で使用するには AWS DMS、特定のタスクで LOBs の使用を有効にします。 CDC または全ロード中、 はプライマリキーを持つテーブルでのみ LOB データ型 AWS DMS をサポートします。 また、LONG RAW 列をロードするためのフル LOB モードは AWS DMS サポートされていません。この代わりに、制限付き LOB モードを使用して LONG RAW 列を Oracle ターゲットに移行できる。限定 LOB モードでは、 は、64 KB を超える LONG RAW 列に設定したデータを 64 KB に AWS DMS 切り捨てます。での LOB サポートの詳細については AWS DMS、「」を参照してください。 AWS DMS タスクでのソースデータベースの LOB サポートの設定
XMLTYPE	CLOB
SDO_GEOMETRY	BLOB (Oracle から Oracle への移行の場合) CLOB (Oracle から PostgreSQL への移行の場合)

以下のデータ型の列とともにソースとして使用されている Oracle テーブルはサポートされておらず、レプリケートすることができません。これらのデータ型の列をレプリケートすると NULL 列が発生します。

- BFILE
- ROWID
- REF
- UROWID
- ユーザー定義のデータ型
- ANYDATA

- VARRAY

> **Note**
>
> 仮想列はサポートされていません。

Oracle 空間データ型の移行

空間データは、空間内のオブジェクトまたは場所のジオメトリ情報を識別します。Oracle データベースでは、空間オブジェクトのジオメトリ説明は型 SDO_GEOMETRY のオブジェクトに保存されます。このオブジェクト内では、ジオメトリ説明は、ユーザー定義テーブルの 1 つの列の 1 つの行に格納されます。

AWS DMS は、Oracle ソースから Oracle または PostgreSQL ターゲットへの Oracle タイプ SDO_GEOMETRY の移行をサポートしています。

を使用して Oracle 空間データ型を移行する場合は AWS DMS、次の考慮事項に注意してください。

- Oracle ターゲットに移行する場合は、型情報を含む USER_SDO_GEOM_METADATA エントリを手動で転送してください。
- Oracle ソースエンドポイントから PostgreSQL ターゲットエンドポイントに移行すると、 はターゲット列 AWS DMS を作成します。これらの列には、2D ディメンションとゼロ (0) に等しい空間参照識別子 (SRID) を持つデフォルトのジオメトリおよび地理的型情報があります。例: `GEOMETRY, 2, 0`。
- PostgreSQL ターゲットに移行する Oracle バージョン 12.1 以前のソースの場合、`SDO2GEOJSON` 関数または `spatialSdo2GeoJsonFunctionName` 追加の接続属性を使用して `SDO_GEOMETRY` オブジェクトを `GEOJSON` 形式に変換します。詳細については、「のソースとして Oracle を使用する場合のエンドポイント設定 AWS DMS」を参照してください。
- AWS DMS は、フル LOB モードの Oracle 空間列移行のみをサポートします。制限付き LOB モードまたはインライン LOB モードは AWS DMS サポートされていません。LOB モードの詳細については、「AWS DMS タスクでのソースデータベースの LOB サポートの設定」を参照。
- は Oracle Spatial Columns を移行するためのフル LOB モード AWS DMS のみをサポートしているため、列のテーブルにはプライマリキーと一意のキーが必要です。プライマリキーと一意のキーがないテーブルは移行でスキップされる。

のソースとしての Microsoft SQL Server データベースの使用 AWS DMS

を使用して、1 つまたは複数の Microsoft SQL Server データベースからデータを移行します AWS DMS。ソースとして SQL Server データベースを使用すると、別の SQL Server データベース、または AWS DMS サポートされている他のデータベースのいずれかにデータを移行できます。

がソースとして AWS DMS サポートする SQL Server のバージョンについては、「」を参照してくださいのソース AWS DMS。

ソース SQL Server データベースは、ネットワーク内のどのコンピュータにもインストールできます。選択したタスクのタイプに応じてソースデータベースに対する適切なアクセス権限のある SQL Server アカウントは、そのデータベースを AWS DMS で使用するために必要です。このアカウントには `view definition` と `view server state` のアクセス権限が必要です。この許可は、次のコマンドを使用して追加します。

```
grant view definition to [user]
grant view server state to [user]
```

AWS DMS は、SQL Server の名前付きインスタンスからのデータの移行をサポートしています。ソースエンドポイントを作成するとき、サーバー名では次の表記を使用できます。

```
IPAddress\InstanceName
```

たとえば、正しいソースエンドポイントサーバー名を以下に示します。ここでは、名前の最初の部分はサーバーの IP アドレス、2 番目の部分は SQL Server インスタンス名 (この例では SQLTest) です。

```
10.0.0.25\SQLTest
```

また、SQL Server の名前付きインスタンスがリッスンするポート番号を取得し、それを使用して AWS DMS ソースエンドポイントを設定します。

> **Note**
>
> ポート 1433 は、Microsoft SQL Server のデフォルトです。ただし、SQL Server をスタートするたびに変化する動的ポート、およびファイアウォール経由で SQL Server に接続するために使用される特定の静的ポート番号もよく使用されます。そのため、 AWS DMS ソース

エンドポイントを作成するときに、SQL Server の名前付きインスタンスの実際のポート番号を知りたいと考えています。

SSL を使用して、SQL Server エンドポイントとレプリケーションインスタンスとの接続を暗号化できます。SQL Server エンドポイントで SSL を使用する方法の詳細については、「での SSL の使用 AWS Database Migration Service」をご参照ください。

SQL Server ソースデータベースと の使用の詳細については AWS DMS、以下を参照してください。

トピック

- のソースとして SQL Server を使用する場合の制限 AWS DMS
- 全ロードのみのタスクに対する許可
- SQL Server のソースからの継続的なレプリケーション (CDC) を使用するための前提条件
- オンプレミスまたは Amazon EC2 上のセルフマネージド型 SQL Server のデータ変更のキャプチャ
- Cloud SQL Server DB インスタンスでの継続的なレプリケーションのセットアップ
- のソースとして Amazon RDS for SQL Server を使用する場合の推奨設定 AWS DMS
- SQL Server でサポートされている圧縮方法
- セルフマネージド SQL Server AlwaysOn 可用性グループの使用
- のソースとして SQL Server を使用する場合のセキュリティ要件 AWS Database Migration Service
- のソースとして SQL Server を使用する場合のエンドポイント設定 AWS DMS
- SQL Server のソースデータ型

のソースとして SQL Server を使用する場合の制限 AWS DMS

SQL Server データベースを AWS DMS のソースとして使用する場合は、以下の制限が適用されます。

- 列のアイデンティティプロパティは、ターゲットデータベース列に移行されません。
- SQL Server エンドポイントは、スパース列でのテーブルの使用をサポートしていません。
- Windows 認証はサポートされていません。
- SQL Server の計算済みフィールドの変更はレプリケーションされていません。
- 一時テーブルはサポートされていません。

- SQL Server パーティション切り替えはサポートされていません。
- WRITETEXT ユーティリティと UPDATETEXT ユーティリティを使用する場合、ソースデータベースに適用されたイベントはキャプチャ AWS DMS されません。
- 次のデータ操作言語 (DML) パターンはサポートされていません。

```
SELECT * INTO new_table FROM existing_table
```

- SQL Server をソースとして使用している場合、列レベルの暗号化はサポートされていません。
- AWS DMS は、SQL Server 2008 または SQL Server 2008 R2 をソースとするサーバーレベルの監査をサポートしていません。これは、SQL Server 2008 および 2008 R2 に関する既知の問題が原因です。例えば、次のコマンドを実行すると、 は失敗 AWS DMS します。

```
USE [master]
GO
ALTER SERVER AUDIT [my_audit_test-20140710] WITH (STATE=on)
GO
```

- SQL Server をソースとして使用する場合、ジオメトリ列は完全 LOB モードではサポートされません。代わりに、制限付き LOB モードを使用するか、インライン LOB モードを使用するように `InlineLobMaxSize` タスク設定を行います。
- レプリケーションタスクで Microsoft SQL Server のソースデータベースを使用する場合、このタスクを削除しても、SQL Server Replication Publisher 定義は削除されません。Microsoft SQL Server システム管理者がそれらの定義を Microsoft SQL Server から削除する必要があります。
- スキーマバインドおよび non-schema-bound ビューからのデータの移行は、全ロードのみのタスクでサポートされています。
- sp_rename を使用したテーブルの名前の変更はサポートされていません (たとえば、`sp_rename 'Sales.SalesRegion', 'SalesReg;`))。
- sp_rename を使用した列の名前の変更はサポートされていません (たとえば、`sp_rename 'Sales.Sales.Region', 'RegID', 'COLUMN';`)。
- AWS DMS では、列のデフォルト値を設定および設定解除する変更処理はサポートされていません (`ALTER TABLE` ステートメントで `ALTER COLUMN SET DEFAULT`句を使用) 。
- AWS DMS では、列の NULL 可能性を設定するための変更処理はサポートされていません (`ALTER TABLE`ステートメントで `ALTER COLUMN [SET|DROP] NOT NULL`句を使用) 。
- SQL Server 2012 および SQL Server 2014 では、可用性グループで DMS レプリケーションを使用する場合、ディストリビューション データベースを可用性グループに配置できません。SQL 2016 では、マージ、双方向、または peer-to-peer レプリケーショントポロジで使用されるディス

トリビューションデータベースを除き、ディストリビューションデータベースを可用性グループに配置することができます。

- パーティション分割されたテーブルの場合、パーティションごとに異なるデータ圧縮設定はサポート AWS DMS されません。
- SQL Server の空間データ型 (GEOGRAPHY および GEOMETRY) に値を挿入するときは、SRID (空間リファレンス系識別子) プロパティを無視するか、別の数値を指定できます。空間データ型のテーブルをレプリケートする場合、 は SRID をデフォルトの SRID (GEOMETRY の場合は 0、GEOGRAPHY の場合は 4326) AWS DMS に置き換えます。
- データベースが MS-REPLICATION または MS-CDC 用に設定されていない場合でも、プライマリキーを持たないテーブルをキャプチャできますが、INSERT/DELETE DML イベントのみがキャプチャされます。UPDATE イベント TRUNCATE TABLE イベントは無視されます。
- Columnstore インデックスはサポートされていません。
- メモリ最適化テーブル (インメモリ OLTP を使用) はサポートされていません。
- 複数の列で構成されるプライマリ キーを持つテーブルをレプリケーションする場合、全ロード中にプライマリ キー列の更新はサポートされません。
- 遅延永続化はサポートされていません。
- RDS がバックアップを実行する方法が原因で、`readBackup0nly=Y` エンドポイント設定 (追加の接続属性) は RDS for SQL Server ソースインスタンスでは機能しません。
- RDS ユーザーは SQL Server ストアド プロシージャ (`sp_repldone`) を実行するアクセス権がないため、Amazon RDS SQL Server ソースインスタンスで `EXCLUSIVE_AUTOMATIC_TRUNCATION` は動作しません。
- AWS DMS は、切り捨てコマンドをキャプチャしません。
- AWS DMS は、高速データベース復旧 (ADR) が有効になっているデータベースからのレプリケーションをサポートしていません。
- AWS DMS は、単一のトランザクション内でのデータ定義言語 (DDL) およびデータ操作言語 (DML) ステートメントのキャプチャをサポートしていません。
- AWS DMS は、データ層アプリケーションパッケージ (DACTAK) のレプリケーションをサポートしていません。
- プライマリキーまたは一意のインデックスが関連し、複数のデータ行を更新する UPDATE ステートメントについては、ターゲットデータベースに変更を適用すると競合が発生する可能性があります。これは例えば、ターゲットデータベースが単一の UPDATE ステートメントではなく INSERT や DELETE のステートメントとして更新を適用する場合に発生する可能性があります。バッチ最適化の適用モードでは、テーブルが無視されることがあります。トランザクション

適用モードでは、UPDATE 操作により制約違反が発生する可能性があります。この問題を回避するには、関連するテーブルをもう一度ロードします。または、例外の適用コントロールテーブル (dmslogs.awsdms_apply_exceptions) で問題のあるレコードを検出して、ターゲットデータベースで手動で編集することもできます。詳細については、「変更処理のチューニング設定」を参照してください。

- AWS DMS では、テーブルとスキーマのレプリケーションはサポートされていません。この際、名前には次のセットの特殊文字が含まれます。

  ```
  \\ -- \n \" \b \r ' \t ;
  ```

- データマスキングはサポートされていません。 は、マスキングせずにマスクされたデータを AWS DMS 移行します。
- AWS DMS は、プライマリキーを持つ最大 32,767 個のテーブルと、テーブルごとに最大 1,000 列をレプリケートします。これは、 がレプリケートされたテーブルごとに SQL Server レプリケーション記事 AWS DMS を作成し、SQL Server レプリケーション記事にはこれらの制限があるためです。
- 変更データキャプチャ (CDC) を使用する場合、一意のインデックスを構成するすべての列を NOT NULL として定義する必要があります。この要件が満たされない場合、SQL Server システムエラー 22838 が発生します。

バックアップトランザクションログにアクセスするときには、次の制限が適用されます。

- 暗号化バックアップはサポートされていません。
- URL または Windows Azure に保存されたバックアップはサポートされていません。
- AWS DMS は、代替共有フォルダからのファイルレベルでのトランザクションログのバックアップの直接処理をサポートしていません。

全ロードのみのタスクに対する許可

全ロード専用タスクを実行するには、次の許可が必要です。 AWS DMS はdms_userログインを作成しないことに注意してください。SQL Server のログインの作成については、「Microsoft SQL Server でのデータベースユーザーの作成」を参照してください。

```
USE db_name;

                CREATE USER dms_user FOR LOGIN dms_user;
                ALTER ROLE [db_datareader] ADD MEMBER dms_user;
```

```
GRANT VIEW DATABASE STATE to dms_user ;

USE master;

GRANT VIEW SERVER STATE TO dms_user;
```

SQL Server のソースからの継続的なレプリケーション (CDC) を使用するための前提条件

オンプレミスまたは Amazon EC2 上のセルフマネージド型 SQL Server データベース、または Amazon RDS や Microsoft Azure SQL マネージドインスタンスなどのクラウドデータベースでは、継続的なレプリケーション (変更データキャプチャ、CDC) を使用できます。

AWS DMS のソースとして SQL Server データベースを使用して継続的なレプリケーションを使用する場合は、次の要件が特別に適用されます。

- SQL Server を完全バックアップ用に設定し、データのレプリケートの開始前にバックアップを実行する必要があります。
- 復旧モデルを [Bulk logged] または [Full] に設定する必要があります。
- 複数のディスクへの SQL Server のバックアップはサポートされていません。異なるディスク上の複数のファイルにデータベースバックアップを書き込むようにバックアップが定義されている場合、 AWS DMS はデータを読み取れず、 AWS DMS タスクは失敗します。
- セルフ管理 SQL Server ソースの場合、DMS CDC タスクで使用されたソース データベースの SQL Server レプリケーション パブリッシャ定義は、そのタスクを削除しても削除されません。SQL Server システム管理者がそれらの定義をセルフマネージド型ソースの SQL Server から削除する必要があります。
- CDC 中に、 は、変更を読み取るために SQL Server トランザクションログのバックアップを検索 AWS DMS する必要があります。 AWS DMS は、ネイティブ形式ではないサードパーティーのバックアップソフトウェアを使用して作成された SQL Server トランザクションログのバックアップをサポートしていません。ネイティブフォーマットのトランザクションログバックアップをサポートするには、サードパーティーのバックアップソフトウェアを使用して作成されている場合は、ソース エンドポイントへの `use3rdPartyBackupDevice=Y` の接続属性を追加します。
- セルフマネージド型 SQL Server ソースの場合、SQL Server は新しく作成されたテーブルの変更を、公開されるまでキャプチャしない点に注意してください。テーブルが SQL Server ソースに追加されると、 はパブリケーションの作成 AWS DMS を管理します。ただし、この処理には数分かかることがあります。この遅延中に新たに作成されたテーブルに行われたオペレーションは、ターゲットにキャプチャまたはレプリケーションされません。

- AWS DMS 変更データキャプチャでは、SQL Server で完全なトランザクションログ記録を有効にする必要があります。SQL Server でフルトランザクションログを有効にするには、MS-REPLICATION または CHANGE DATA CAPTURE (CDC) を有効にします。
- MS CDC キャプチャジョブがこのような変更を処理するまで、SQL Server の tlog エントリは再利用対象としてマークされません。
- CDC オペレーションはメモリ最適化テーブルに対してはサポートされていません。この制限は、SQL Server 2014 (この機能が最初に導入されたバージョン) 以降に適用されます。
- AWS DMS 変更データキャプチャには、デフォルトで Amazon EC2 またはオンプレミス SQL サーバー上のディストリビューションデータベースがソースとして必要です。このため、プライマリキーがあるテーブルの MS レプリケーションを設定する際は、ディストリビューターを有効にしていることを確認します。

オンプレミスまたは Amazon EC2 上のセルフマネージド型 SQL Server のデータ変更のキャプチャ

Microsoft SQL Server ソースデータベースからの変更をキャプチャするには、データベースが完全バックアップ用に構成されていることを確認してください。データベースをフルリカバリモードまたは一括ログモードで構成します。

セルフマネージド SQL Server ソースの場合、 は以下 AWS DMS を使用します。

MS レプリケーション

プライマリ キーを持たないテーブルの変更をキャプチャします。これは、ソース SQL Server インスタンスの AWS DMS エンドポイントユーザーに sysadmin 権限を付与することで自動的に設定できます。または、このセクションの手順に従ってソースを準備し、 AWS DMS エンドポイントの sysadmin 権限を持たないユーザーを使用することもできます。

MS-CDC

プライマリ キーを持たないテーブルの変更をキャプチャします。MS-CDC は、すべてのテーブルのデータベースレベルで個別に有効にします。

継続的なレプリケーション (CDC) に SQL Server データベースをセットアップする場合、次のいずれかの操作を実行できます：

- sysadmin ロールを使用する継続的なレプリケーションをセットアップします
- sysadmin ロールを使用しない継続的なレプリケーションをセットアップします。

セルフマネージド型 SQL Server での継続的なレプリケーションのセットアップ

このセクションには、sysadmin ロールを使用する場合と使用しない場合の、セルフマネージド型 SQL Server での継続的なレプリケーションの設定に関する情報が記載されています。

トピック

- セルフマネージド型 SQL Server での継続的なレプリケーションのセットアップ: sysadmin ロールを使用
- スタンドアロン SQL Server での継続的なレプリケーションのセットアップ: sysadmin ロールなし

セルフマネージド型 SQL Server での継続的なレプリケーションのセットアップ: sysadmin ロールを使用

AWS DMS SQL Server の 継続的レプリケーションでは、プライマリ キーを持つテーブルにはネイティブ SQL Server レプリケーションを使用し、プライマリ キーを持たないテーブルには変更データ キャプチャ (CDC) を使用します。

継続的レプリケーションを設定する前に、「SQL Server のソースからの継続的なレプリケーション (CDC) を使用するための前提条件」をご参照ください。

プライマリキーを持つテーブルの場合、 AWS DMS 通常、ソースで必要なアーティファクトを設定できます。ただし、セルフ管理 SQL Server ソース インスタンスの場合、最初に、SQL Server のディストリビューションを手動で設定する必要があります。その後、sysadmin 権限を持つ AWS DMS ソースユーザーは、プライマリキーを持つテーブルのパブリケーションを自動的に作成できます。

ディストリビューションがすでに設定されているかどうかを確認するには、以下のコマンドを実行します。

```
sp_get_distributor
```

列ディストリビューションの結果が NULL の場合、ディストリビューションは設定されていません。ディストリビューションを設定するには、次の手順を使用します。

ディストリビューションを設定するには

1. SQL Server Management Studio (SSMS) ツールを使用して SQL Server ソースデータベースに接続します。

2. [Replication] (レプリケーション) フォルダのコンテキスト (右クリック) メニューを開き、[Configure Distribution] (ディストリビューション設定) を選択します。ディストリビューションの構成ウィザードが開きます。
3. ウィザードに従ってデフォルト値を入力し、ディストリビューションを作成します。

CDC をセットアップするには

AWS DMS バージョン 3.4.7 以降では、読み取り専用レプリカを使用していない場合、データベースとすべてのテーブルに MS CDC を自動的に設定できます。この機能を使用するには、`SetUpMsCdcForTables` ECA を true に設定します。ECA の詳細については、「エンドポイント設定」を参照してください。

3.4.7 より AWS DMS 前のバージョンの の場合、またはソースとしての読み取り専用レプリカの場合は、次の手順を実行します。

1. プライマリ キーがないテーブルの場合、データベースの MS-CDC をセットアップします。そのためには、sysadmin ロールが割り当てられたアカウントを使用し、次のコマンドを実行します。

```
use [DBname]
EXEC sys.sp_cdc_enable_db
```

2. 次に、ソーステーブルごとに MS-CDC を設定します。一意キーはあるがプライマリ キーがないテーブルごとに、次のクエリを実行して MS-CDC を設定します。

```
exec sys.sp_cdc_enable_table
@source_schema = N'schema_name',
@source_name = N'table_name',
@index_name = N'unique_index_name',
@role_name = NULL,
@supports_net_changes = 1
GO
```

3. プライマリ キーと一意キーの両方がないテーブルごとに、次のクエリを実行して MS-CDC を設定します。

```
exec sys.sp_cdc_enable_table
@source_schema = N'schema_name',
@source_name = N'table_name',
@role_name = NULL
```

```
GO
```

特定のテーブルの MS-CDC をセットアップする方法の詳細については、 SQL Server のドキュメントをご参照ください。

スタンドアロン SQL Server での継続的なレプリケーションのセットアップ: sysadmin ロールなし

sysadmin ロールを使用せずにスタンドアロン SQL Server で継続的なレプリケーションを設定する方法については、「スタンドアロン SQL Server での継続的なレプリケーションのセットアップ: sysadmin ロールなし」を参照してください。

Cloud SQL Server DB インスタンスでの継続的なレプリケーションのセットアップ

このセクションでは、クラウドでホストされている SQL Server のデータベースインスタンスで CDC をセットアップする方法について説明します。クラウドでホストされる SQL Serverインスタンスは、Amazon RDS for SQL Server、Azure SQL Manged Instance、またはその他のマネージド型 Cloud SQL Server インスタンス上で実行されるインスタンスです。各データベースタイプでの継続的なレプリケーションの制限については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

継続的レプリケーションを設定する前に、SQL Server のソースからの継続的なレプリケーション (CDC) を使用するための前提条件 をご参照ください。

セルフ管理 SQL Server ソースとは異なり、Amazon RDS for SQL Server では MS レプリケーションはサポートされません。したがって、 AWS DMS はプライマリ キーの有無にかかわらずテーブルに MS-CDC を使用する必要があります。

Amazon RDS は、ソース SQL Server インスタンスで進行中の変更に が AWS DMS 使用するレプリケーションアーティファクトを設定するための sysadmin 権限を付与しません。次の手順に従って、(管理ユーザーアクセス権限を使用して) Amazon RDS インスタンスの MS-CDC をオンにする必要があります。

Cloud SQL Server DB インスタンスで MS-CDC を有効にするには

1. 次のクエリのいずれかをデータベース レベルで実行します。

   RDS for SQL Server DB インスタンスの場合は、次のクエリを使用します。

   ```
   exec msdb.dbo.rds_cdc_enable_db 'DB_name'
   ```

Azure SQL マネージド DB インスタンスの場合は、次のクエリを使用します。

```
USE DB_name
GO
EXEC sys.sp_cdc_enable_db
GO
```

2. プライマリ キーがあるテーブルごとに、次のクエリを実行して MS-CDC を有効にします。

```
exec sys.sp_cdc_enable_table
@source_schema = N'schema_name',
@source_name = N'table_name',
@role_name = NULL,
@supports_net_changes = 1
GO
```

一意キーはあるがプライマリ キーがないテーブルごとに、次のクエリを実行して MS-CDC を有効にします。

```
exec sys.sp_cdc_enable_table
@source_schema = N'schema_name',
@source_name = N'table_name',
@index_name = N'unique_index_name',
@role_name = NULL,
@supports_net_changes = 1
GO
```

プライマリキーと一意キーの両方がないテーブルごとに、次のクエリを実行して MS-CDC を有効にします。

```
exec sys.sp_cdc_enable_table
@source_schema = N'schema_name',
@source_name = N'table_name',
@role_name = NULL
GO
```

3. 次のコマンドを使用して、ソースで変更を使用できる保持期間を設定します。

```
use dbname
EXEC sys.sp_cdc_change_job @job_type = 'capture' ,@pollinginterval = 86399
```

```
exec sp_cdc_stop_job 'capture'
exec sp_cdc_start_job 'capture'
```

パラメータ @pollinginterval は秒単位で測定され、推奨値 86399 に設定されます。つまり、@pollinginterval = 86399 の場合、トランザクションログは 86,399 秒 (約 1 日) の変更を保持します。手順 exec sp_cdc_start_job 'capture' によって設定が開始されます。

> **Note**
>
> SQL Server の一部のバージョンでは、pollinginterval が 3599 秒以上に設定されている場合、値はデフォルトの 5 秒にリセットされます。この場合、 が T-Log エントリを読み取る前に、T-Log AWS DMS エントリは消去されます。この既知の問題の影響を受ける SQL Server のバージョンを確認するには、「このマイクロソフト KB 記事」をご参照ください。

マルチ AZ で Amazon RDS を使用している場合は、フェイルオーバー時に適切な値を持つようにセカンダリも設定してください。

```
exec rdsadmin..rds_set_configuration 'cdc_capture_pollinginterval' , 86399
```

SQL Server ソースへの継続的な変更をキャプチャする AWS DMS レプリケーションタスクが 1 時間以上停止する場合は、次の手順を使用します。

AWS DMS レプリケーションタスク中に保持期間を維持するには

1. 次のコマンドを使用して、トランザクションログを切り捨てるジョブを停止します。

   ```
   exec sp_cdc_stop_job 'capture'
   ```

2. AWS DMS コンソールでタスクを検索し、タスクを再開します。
3. [Monitoring] (モニタリング) タブを選択し、CDCLatencySource メトリクスを選択します。
4. CDCLatencySource メトリクスが 0 (ゼロ) に等しく、そのままの場合、次のコマンドを使用して、トランザクション ログ切り捨てジョブを再開します。

   ```
   exec sp_cdc_start_job 'capture'
   ```

必ず SQL Server トランザクションログを切り捨てるジョブをスタートしてください。そうしないと、SQL Server インスタンスのストレージがいっぱいになる可能性があります。

Cloud SQL Server DB インスタンスでの継続的なレプリケーションの制限

- AWS DMS は、アクティブなトランザクションログのみを使用した継続的なレプリケーション (CDC) をサポートします。CDC ではバックアップログを使用できません。
- イベントをアクティブトランザクションログからバックアップ ログに移動したり、アクティブトランザクションログから切り捨てたりすると、イベントが損失する可能性があります。

のソースとして Amazon RDS for SQL Server を使用する場合の推奨設定 AWS DMS

Amazon RDS for SQL Server をソースとして使用する場合、キャプチャジョブはパラメータ `maxscans` と `maxtrans` に依存しています。このようなパラメータは、キャプチャがトランザクションログに対して実行するスキャンの最大数と、スキャンごとに処理されるトランザクション数を制御します。

データベースでは、トランザクション数が `maxtrans*maxscans` の場合、`polling_interval` 値を増やすとアクティブなトランザクションログレコードが蓄積されてしまう可能性があります。これにより、トランザクションログのサイズが増大する可能性があります。

AWS DMS は MS-CDC キャプチャジョブに依存しないことに注意してください。MS-CDC キャプチャジョブは、トランザクションログエントリを処理済みとしてマークします。これにより、トランザクションログのバックアップジョブはトランザクションログからエントリを削除できます。

トランザクションログのサイズと MS-CDC ジョブの正常な実行はモニタリングすることをお勧めします。MS-CDC ジョブが失敗すると、トランザクションログが過度に増加し、 AWS DMS レプリケーションが失敗する可能性があります。MS-CDC キャプチャジョブのエラーは、ソースデータベースの `sys.dm_cdc_errors` 動的管理ビューを使用してモニタリングできます。トランザクションログのサイズのモニタリングには、`DBCC SQLPERF(LOGSPACE)` 管理コマンドを使用します。

MS-CDC によるトランザクション ログの増加に対処するには

1. データベース`Log Space Used %`の AWS DMS が からレプリケートされていることを確認し、継続的に増加することを確認します。

   ```
   DBCC SQLPERF(LOGSPACE)
   ```

2. トランザクションログのバックアッププロセスをブロックしている要因を特定します。

```
Select log_reuse_wait, log_reuse_wait_desc, name from sys.databases where name =
 db_name();
```

`log_reuse_wait_desc` 値が `REPLICATION` と等しい場合、ログバックアップの保持は MS-CDC のレイテンシーが原因です。

3. `maxtrans` と `maxscans` パラメータの値を増やして、キャプチャジョブが処理するイベント数を増やします。

```
EXEC sys.sp_cdc_change_job @job_type = 'capture' ,@maxtrans = 5000, @maxscans = 20
exec sp_cdc_stop_job 'capture'
exec sp_cdc_start_job 'capture'
```

この問題に対処するには、 `maxscans`と の値を設定します。`maxtrans`これは、`maxtrans*maxscans`がソースデータベースからレ AWS DMS プリケートするテーブルに対して毎日生成されるイベントの平均数と等しくなります。

このようなパラメータを推奨値よりも高く設定すると、キャプチャジョブはトランザクションログ内のすべてのイベントを処理します。このようなパラメータを推奨値より低く設定すると、MS-CDC のレイテンシーが増加し、トランザクションログが増大します。

ワークロードの変化により生成されるイベントの数が変化するため、`maxtrans` と `maxscans`の適切な値を特定することが困難である場合があります。この場合、MS-CDC のレイテンシーのモニタリングを設定することをお勧めします。詳細については、SQL Server ドキュメントの「プロセスを監視する」を参照してください。その後、モニタリング結果に基づいて`maxtrans` と `maxscans` を動的に設定します。

AWS DMS タスクを再開または続行するために必要なログシーケンス番号 (LSNsがタスクで見つからない場合、タスクが失敗し、完全なリロードが必要になることがあります。

> **Note**
>
> AWS DMS を使用して RDS for SQL Server ソースからデータをレプリケートする場合、Amazon RDS インスタンスの停止イベント後にレプリケーションを再開しようとすると、エラーが発生する可能性があります。これは、SQL Server エージェントプロセスが停止または開始のイベントの後に再起動する際、キャプチャジョブプロセスを再起動するためです。これにより、MS-CDC のポーリング間隔がバイパスされます。

このため、MS-CDC キャプチャジョブ処理よりもトランザクションボリュームの低いデータベースでは、 が停止した場所から再開 AWS DMS する前に、データが処理またはレプリケートおよびバックアップとしてマークされ、次のエラーが発生する可能性があります。

```
[SOURCE_CAPTURE ]E: Failed to access LSN '0000dbd9:0006f9ad:0003' in
 the backup log sets since BACKUP/LOG-s are not available. [1020465]
 (sqlserver_endpoint_capture.c:764)
```

この問題を軽減するには、`maxtrans` と `maxscans` の値を上記の推奨のとおりに設定します。

SQL Server でサポートされている圧縮方法

AWS DMS での SQL Server 圧縮方法のサポートについては、次の点に注意します。

- AWS DMS は、SQL Server バージョン 2008 以降で行/ページ圧縮をサポートしています。
- AWS DMS は Vardecimal ストレージ形式をサポートしていません。
- AWS DMS は、スパース列と列構造圧縮をサポートしていません。

セルフマネージド SQL Server AlwaysOn 可用性グループの使用

SQL Server の Always On 可用性グループは、データベースミラーリングに代わるエンタープライズレベルの代替方法を提供する高可用性と災害復旧ソリューションです。

では AWS DMS、単一のプライマリまたはセカンダリ可用性グループのレプリカから変更を移行できます。

プライマリ可用性グループレプリカの使用

プライマリ可用性グループを のソースとして使用するには AWS DMS、次の手順を実行します。

1. 可用性レプリカ内のすべての SQL Server インスタンスでディストリビューションオプションを有効にします。詳細については、「セルフマネージド型 SQL Server での継続的なレプリケーションのセットアップ」を参照してください。
2. AWS DMS コンソールで、SQL Server ソースデータベース設定を開きます。[サーバー名] には、可用性グループリスナーのために設定したドメインネームサービス (DNS) 名または IP アドレスを指定します。

AWS DMS タスクを初めて開始する場合、開始に通常よりも時間がかかることがあります。このような速度の低下は、テーブルアーティクルの作成が可用性グループサーバーによりレプリケートされるために発生します。

セカンダリ可用性グループレプリカの使用

セカンダリ可用性グループを のソースとして使用するには AWS DMS、次の手順を実行します。

1. 個々のレプリカへの接続には、 AWS DMS ソースエンドポイントユーザーが使用するものと同じ認証情報を使用します。
2. AWS DMS レプリケーションインスタンスが既存のすべてのレプリカの DNS 名を解決できることを確認し、それらに接続します。次の SQL クエリを使用して、すべてのレプリカの DNS 名を取得できます。

```
select ar.replica_server_name, ar.endpoint_url from sys.availability_replicas ar
JOIN sys.availability_databases_cluster adc
ON adc.group_id = ar.group_id AND adc.database_name = '<source_database_name>';
```

3. ソースエンドポイントを作成する際、エンドポイントの [サーバー名] またはエンドポイントのシークレットの [サーバーアドレス] として可用性グループリスナーの DNS 名を指定します。可用性グループリスナーの詳細については、SQL Server ドキュメントの「可用性グループリスナーとは」を参照してください。

   パブリック DNS サーバーまたはオンプレミスの DNS サーバーのいずれかを使用して、可用性グループリスナー、プライマリレプリカ、セカンダリレプリカを解決できます。オンプレミスの DNS サーバーを使用するには、Amazon Route 53 Resolver を設定します。詳細については、「独自のオンプレミスネームサーバーの使用」を参照してください。
4. 次の追加の接続属性をソースエンドポイントに追加します。

追加の接続属性	値	メモ
`applicationIntent`	`ReadOnly`	この ODBC 設定がないと、レプリケーションタスクはプライマリ可用性グループのレプリカにルーティングされる。詳細については、SQL Server ドキュメントの「 SQL Server Native Client の HADR サポート」を参照。

追加の接続属性	値	メモ
multiSubnetFailover	yes	詳細については、SQL Server ドキュメントの「SQL Server Native Client の HADR サポート」を参照。
alwaysOnSharedSyncedBackupIsEnabled	false	詳細については、「のソースとして SQL Server を使用する場合のエンドポイント設定 AWS DMS」を参照してください。
activateSafeguard	false	詳細については、「制限事項」を参照してください。
setUpMsCdcForTables	false	詳細については、「制限事項」を参照してください。

5. 可用性グループ内のすべてのレプリカでディストリビューションオプションを有効にします。すべてのノードをディストリビューターリストに追加します。詳細については、「ディストリビューションを設定するには」を参照してください。
6. プライマリ読み取り/書き込みレプリカで次のクエリを実行して、データベースの公開を有効にします。このクエリはデータベースに対して 1 回だけ実行します。

```
sp_replicationdboption @dbname = N'<source DB name>', @optname = N'publish', @value
 = N'true';
```

制限事項

セカンダリ可用性グループレプリカを使用する場合の制限は次のとおりです。

- AWS DMS は、読み取り専用の可用性グループレプリカをソースとして使用する場合の Safeguard をサポートしていません。詳細については、「のソースとして SQL Server を使用する場合のエンドポイント設定 AWS DMS」を参照してください。
- AWS DMS は、読み取り専用の可用性グループレプリカをソースとして使用する場合、setUpMsCdcForTables追加の接続属性をサポートしません。詳細については、「のソースとして SQL Server を使用する場合のエンドポイント設定 AWS DMS」を参照してください。

- AWS DMS は、バージョン 3.4.7 以降の継続的なレプリケーション (変更データキャプチャ、または CDC) のソースデータベースとして、セルフマネージドセカンダリ可用性グループレプリカを使用できます。Cloud SQL Server のマルチ AZ リードレプリカはサポートされていません。の以前のバージョンを使用する場合は AWS DMS、CDC のソースデータベースとしてプライマリ可用性グループのレプリカを使用してください。

他のノードへのフェイルオーバー

エンドポイントApplicationIntentの追加接続属性を に設定するとReadOnly、 AWS DMS タスクは読み取り専用ルーティングの優先度が最も高い読み取り専用ノードに接続します。その後、最も優先度の高い読み取り専用ノードが使用できない場合は、可用性グループ内のその他の読み取り専用ノードにフェイルオーバーします。を設定しない場合ApplicationIntent、 AWS DMS タスクは可用性グループのプライマリ (読み取り/書き込み) ノードにのみ接続します。

のソースとして SQL Server を使用する場合のセキュリティ要件 AWS Database Migration Service

AWS DMS ユーザーアカウントには、少なくとも接続先のソース SQL Server データベースのdb_ownerユーザーロールが必要です。

のソースとして SQL Server を使用する場合のエンドポイント設定 AWS DMS

追加の接続属性の使用と同様、エンドポイントの設定を使用して、ソースの SQL Server データベースを設定できます。 AWS DMS コンソールを使用するか、 の create-endpoint コマンドを JSON --microsoft-sql-server-settings '{"*EndpointSetting*": *"value"*, *...*}' 構文で使用してAWS CLI、ソースエンドポイントを作成するときに設定を指定します。

次の表は、ソースとして SQL Server を使用できるエンドポイント設定を説明しています。

名前	説明
ActivateSafeguard	この属性は Safeguard をオンまたはオフにする。SafeguardPolicy の詳細については、次を参照。 デフォルト値: true 有効な値: {false, true}

名前	説明
	例: `'{"ActivateSafeguard": true}'`
`AlwaysOnSharedSynchedBackupIsEnabled`	この属性は、Always On 可用性グループクラスターの一部としてホストされている SQL Server ソースデータベースから移行 AWS DMS するときの の動作を調整します。 AWS DMS では、Always On クラスターで実行するように設定された SQL Server ソースデータベースのサポートが強化されています。この場合、 AWS DMS は、ソースデータベース インスタンスがホストされているノード以外の Always On クラスタ内のノードからトランザクション バックアップが実行されているかどうかを追跡しようとします。移行タスクの起動時に、 はクラスター内の各ノードへの接続 AWS DMS を試みますが、いずれかのノードに接続できない場合は失敗します。 Always On クラスター内のすべてのノードでトランザクションバックアップを AWS DMS ポーリングする必要がある場合は、この属性を に設定しますfalse。 デフォルト値: `true` 有効な値: `true` または `false` 例: `'{"AlwaysOnSharedSynchedBackupIsEnabled": false}'`
`"ApplicationIntent": "readonly"`	この ODBC ドライバー属性設定により、SQL Server はレプリケーションタスクを最も優先度の高い読み取り専用ノードにルーティングする。この設定を行わないと、SQL Server はレプリケーションタスクをプライマリ読み取り/書き込みノードにルーティングする。

名前	説明
EnableNonSysadminWrapper	sysadmin ユーザーを使用せずにスタンドアロン SQL Server 上で継続的なレプリケーションを設定する場合、このエンドポイント設定を使用する。このパラメータは、 AWS DMS バージョン 3.4.7 以降でサポートされています。スタンドアロン SQL Server での継続的なレプリケーションの設定については、「スタンドアロン SQL Server での継続的なレプリケーションのセットアップ: sysadmin ロールなし」を参照。 デフォルト値: false 有効な値: true、false 例: '{"EnableNonSysadminWrapper": true}'
ExecuteTimeout	この追加の接続属性 (ECA) を使用して、SQL Server インスタンスのクライアントステートメントのタイムアウトを秒単位で設定する。デフォルト値は 60 秒です。 例: '{"ExecuteTimeout": 100}'
FatalOnSimpleModel	true に設定する場合、SQL Server データベースの復旧モデルが simple に設定されていると、このパラメータは致命的エラー発生させる。 デフォルト値: false 有効な値: true または false 例: '{"FatalOnSimpleModel": true}'
ForceLobLookup	インライン LOB で LOB ルックアップを強制する。 デフォルト値: false 有効な値: true、false 例: '{"ForceLobLookup": false}'

名前	説明
`"MultiSubnetFailover": "Yes"`	この ODBC ドライバ属性によって、可用性グループのフェールオーバー時に DMS が新しいプライマリに接続しやすくなります。この属性は、接続が切断されているか、リスナーの IP アドレスが正しくない状況向けに設計されています。このような状況では、 は可用性グループリスナーに関連付けられているすべての IP アドレスへの接続 AWS DMS を試みます。
`ReadBackupOnly`	この属性を使用するには、sysadmin 権限が必要です。この属性が に設定されている場合Y、継続的レプリケーション中にはトランザクションログのバックアップからのみ変更が AWS DMS 読み取られ、アクティブなトランザクションログファイルからは読み取られません。このパラメータを Y に設定すると、完全ロードおよび継続的なレプリケーションタスク中に、アクティブなトランザクションのログファイルの拡張を制御することができます。ただし、これによって継続的なレプリケーションに一部のソースレイテンシーが生じることがあります。 有効な値: `N` または `Y`。デフォルトは `N` です。 例: `'{"ReadBackupOnly": Y}'` 注意:このパラメータは、RDS がバックアップを実行する方法であるため、Amazon RDS SQL Server ソースインスタンスでは機能しません。

名前	説明
SafeguardPolicy	最適なパフォーマンスを得るために、 AWS DMS は、アクティブなトランザクションログ (TLOG) から未読の変更をすべてキャプチャしようとします。ただし、切り捨てが原因で、アクティブな TLOG に未読の変更がすべて含まれていない場合がある。これが発生すると、 はログバックアップ AWS DMS にアクセスして欠落している変更をキャプチャします。ログバックアップにアクセスする必要性を最小限に抑えるために、 は次のいずれかの方法を使用して切り捨て AWS DMS を防止します。 1. RELY_ON_SQL_SERVER_REPLICATION_AGENT (データベースでトランザクションを開始する) : これは のデフォルトです AWS DMS。 この設定を使用する場合、 AWS DMS では レプリケーションのマークが付けられたトランザクションを AWS DMS がアクティブな TLOG から移動できるように、SQL Server ログのリーダーエージェントが実行されている必要がある。ログリーダーエージェントが実行されていないと、アクティブな TLOG がフルになり、問題が解決されるまでソースデータベースが読み取り専用モードに切り替わる可能性があることに注意する。以外の目的でデータベースで Microsoft レプリケーションを有効にする必要がある場合は AWS DMS、この設定を選択する必要があります。 この設定を使用すると、 は という名前のテーブルを作成してログのバックアップ読み取り AWS DMS を最小限に抑えawsdms_truncation_safeguard 、データベース内のオープントランザクションを模倣して TLOG が切り捨てるのを防ぎます。これにより、5 分間 (デフォルト)、データベースがイベントを切り捨ててバックアップログに移動することを回避できる。メンテナンスジョブが失敗する可能性があるため、このテーブルがいずれのメンテナンスプランにも含まれ

名前	説明
	ていないことを確認する。Start Transactions データベースオプションで設定されたタスクがない場合は、テーブルを削除しても問題ない。 2. EXCLUSIVE_AUTOMATIC_TRUNCATION （単一のタスク**sp_repldone** で排他的に を使用）: この設定を使用すると、 AWS DMS は、 ready for truncation を使用してログエントリを としてマークするレプリケーション エージェントプロセスを完全に制御しますsp_repldone 。この設定では、 は RELY_ON_SQL_SERVER_REPLICATION_AGENT (デフォルト) 設定と同様にダミートランザクションを使用し AWS DMS ません。この設定は、MS レプリケーションがソースデータベース AWS DMS 以外の目的で使用されていない場合にのみ使用できます。また、この設定を使用する場合、データベースにアクセスできる AWS DMS タスクは 1 つだけです。同じデータベースに対して並列 AWS DMS タスクを実行する必要がある場合は、 を使用しますRELY_ON_SQL_SERVER_REPLICATION_AGENT 。 • この設定では、データベース内でログリーダーエージェントを停止する必要がある。タスクの開始時に Log Reader エージェントが実行されている場合、AWS DMS タスクは強制的に停止します。タスクを開始する前にログリーダーエージェントを手動で停止することもできる。 • この方法を MS-CDC で使用する場合、[MS-CDC キャプチャ] ジョブと [MS-CDC クリーンアップ] ジョブを停止して無効にする必要がある。 • AWS DMS はリモートマシンにアクセスできないため、Microsoft SQL Server Migration ジョブがリモートディストリビューターマシンで実行されている場合、この設定は使用できません。 • Amazon RDS ユーザーは sp_repldone ストアドプロシージャを実行するアクセス権がないため、A

名前	説明
	mazon RDS for SQL Server ソースインスタンスでは EXCLUSIVE_AUTOMATIC_TRUNCATION は機能しない。 • sysadmin ロールを使用せずに SafeguardPolicy を EXCLUSIVE_AUTOMATIC_TRUNCATION に設定する場合は、dbo.syscategories オブジェクトと dbo.sysjobs オブジェクトに対するアクセス権限を dmsuser ユーザーに付与する必要がある。 デフォルト値: RELY_ON_SQL_SERVER_REPLICATION_AGENT 有効な値: {EXCLUSIVE_AUTOMATIC_TRUNCATION , RELY_ON_SQL_SERVER_REPLICATION_AGENT } 例: '{"SafeguardPolicy": "EXCLUSIVE_AUTOMATIC_TRUNCATION"}'
SetUpMsCdcForTables	この属性は、ソースデータベースと MS-Replication が有効になっていないタスクマッピング内のテーブルの MS-CDC を有効にする。この値を true に設定すると、ソースデータベースで sp_cdc_enable_db ストアドプロシージャが実行され、ソースデータベースで MS-Replication が有効になっていないタスクの各テーブルで sp_cdc_enable_table ストアドプロシージャが実行される。ディストリビューションの有効化の詳細については、「セルフマネージド型 SQL Server での継続的なレプリケーションのセットアップ」を参照。 有効な値: {true, false} 例: '{"SetUpMsCdcForTables": true}'

名前	説明
TlogAccessMode	CDC データをフェッチするために使用されるモードを示す。 デフォルト値: PreferTlog 有効値: BackupOnly 、PreferBackup 、PreferTlog 、TlogOnly 例: '{"TlogAccessMode": "PreferTlog"}'
Use3rdPartyBackupDevice	この属性がYに設定されている場合, AWS DMS は、サードパーティのトランザクション ログ バックアップがネイティブ形式で作成されている限り処理します。

SQL Server のソースデータ型

のソースとして SQL Server を使用するデータ移行では、ほとんどの SQL Server データ型 AWS DMS がサポートされます。次の表は、 の使用時にサポートされる SQL Server ソースデータ型 AWS DMS と AWS DMS 、データ型からのデフォルトマッピングを示しています。

ターゲットにマッピングされるデータ型を表示する方法については、使用しているターゲットエンドポイントのセクションをご参照ください。

AWS DMS データ型の詳細については、「」を参照してくださいAWS Database Migration Service のデータ型。

SQL Server のデータ型	AWS DMS データ型
BIGINT	INT8
BIT	BOOLEAN
DECIMAL	NUMERIC
INT	INT4
MONEY	NUMERIC

SQL Server のデータ型	AWS DMS データ型
NUMERIC (p,s)	NUMERIC
SMALLINT	INT2
SMALLMONEY	NUMERIC
TINYINT	UINT1
REAL	REAL4
FLOAT	REAL8
DATETIME	DATETIME
DATETIME2 (SQL Server 2008 以降)	DATETIME
SMALLDATETIME	DATETIME
DATE	DATE
TIME	TIME
DATETIMEOFFSET	WSTRING
CHAR	STRING
VARCHAR	STRING

SQL Server のデータ型	AWS DMS データ型
VARCHAR(max)	CLOB TEXT でこのデータ型を使用するには AWS DMS、特定のタスクで CLOB データ型の使用を有効にする必要があります。 SQL Server テーブルの場合、SQL Server の LOB 列の値を変更しない UPDATE ステートメントに対しても、 はターゲットの LOB 列 AWS DMS を更新します。 CDC 中、 はプライマリ キーを含むテーブルでのみ CLOB データ型 AWS DMS をサポートします。
NCHAR	WSTRING
NVARCHAR (長さ)	WSTRING

SQL Server のデータ型	AWS DMS データ型
NVARCHAR (最大)	NCLOB NTEXT でこのデータ型を使用するには AWS DMS、特定のタスク SupportLobs で の使用を有効にする必要があります。Lob サポートの有効化に関する詳細については、AWS DMS タスクでのソースデータベースの LOB サポートの設定 をご参照ください。 SQL Server テーブルの場合、SQL Server の LOB 列の値を変更しない UPDATE ステートメントに対しても、 はターゲットの LOB 列 AWS DMS を更新します。 CDC 中、 はプライマリ キーを含むテーブルでのみ CLOB データ型 AWS DMS をサポートします。
BINARY	BYTES
VARBINARY	BYTES

SQL Server のデータ型	AWS DMS データ型
VARBINARY (最大)	BLOB IMAGE SQL Server テーブルの場合、SQL Server の LOB 列の値を変更しない UPDATE ステートメントに対しても、 はターゲットの LOB 列 AWS DMS を更新します。 でこのデータ型を使用するには AWS DMS、特定のタスクで BLOB データ型の使用を有効にする必要があります。 AWS DMS は、プライマリキーを含むテーブルでのみ BLOB データ型をサポートします。
TIMESTAMP	BYTES
UNIQUEIDENTIFIER	STRING
HIERARCHYID	SQL Server ターゲットエンドポイントにレプリケートする場合は、HIERARCHYID を使用します。 他のすべてのターゲットエンドポイントにレプリケートする場合は、WSTRING (250) を使用します。

SQL Server のデータ型	AWS DMS データ型
XML	NCLOB SQL Server テーブルの場合、SQL Server の LOB 列の値を変更しない UPDATE ステートメントに対しても、 はターゲットの LOB 列 AWS DMS を更新します。 でこのデータ型を使用するには AWS DMS、特定のタスクで NCLOB データ型の使用を有効にする必要があります。 CDC 中、 はプライマリ キーを含むテーブルでのみ NCLOB データ型 AWS DMS をサポートします。
GEOMETRY	このデータ型をサポートするターゲットエンドポイントにレプリケートする場合は、GEOMETRY を使用します。 このデータ型をサポートしないターゲットエンドポイントにレプリケートする場合は、CLOB を使用します。
GEOGRAPHY	このデータ型をサポートするターゲットエンドポイントにレプリケートする場合は、GEOGRAPHY を使用します。 このデータ型をサポートしないターゲットエンドポイントにレプリケートする場合は、CLOB を使用します。

AWS DMS は、次のデータ型のフィールドを含むテーブルをサポートしていません。

- CURSOR
- SQL_VARIANT
- TABLE

> **Note**
>
> ユーザー定義のデータ型は、基本型に従ってサポートされます。たとえば、DATETIME をベースとするユーザー定義のデータ型は DATETIME データ型として扱われます。

AWS DMS のソースとしての Microsoft Azure SQL Database の使用

AWS DMS では、Microsoft Azure SQL Database をソースとして SQL Server とほぼ同じ方法で使用できます。AWS DMS は、オンプレミスまたは Amazon EC2 インスタンスで実行されている SQL Server でサポートされているのと同じデータベースバージョンをソースとしてサポートします。

詳細については、「のソースとしての Microsoft SQL Server データベースの使用 AWS DMS」を参照してください。

> **Note**
>
> AWS DMS は、Azure SQL Database を使用した変更データキャプチャ (CDC) 操作はサポートしていません。

AWS DMS のソースとしての Microsoft Azure SQL Managed Instance の使用

AWS DMS では、Microsoft Azure SQL Managed Instance をソースとして SQL Server とほぼ同じ方法で使用できます。AWS DMS は、オンプレミスまたは Amazon EC2 インスタンスで実行されている SQL Server でサポートされているのと同じデータベースバージョンをソースとしてサポートします。

詳細については、「のソースとしての Microsoft SQL Server データベースの使用 AWS DMS」を参照してください。

AWS DMS のソースとしての Microsoft Azure Database for PostgreSQL フレキシブル サーバーの使用

AWS DMS では、Microsoft Azure Database for PostgreSQL フレキシブルサーバーを PostgreSQL とほとんど同じ方法でソースとして使用できます。

AWS DMS がソースとしてサポートする Microsoft Azure Database for PostgreSQL フレキシブルサーバーのバージョンについては、「のソース AWS DMS」を参照してください。

論理レプリケーションとデコードのための Microsoft Azure for PostgreSQL フレキシブルサーバーのセットアップ

データベースの移行中に、Microsoft Azure Database for PostgreSQL フレキシブルサーバーの論理レプリケーションとデコード機能を使用できます。

論理デコードには、DMS は `test_decoding` または `pglogical` プラグインのどちらかを使用します。`pglogical` プラグインがソースのPostgreSQL データベースで利用できる場合、DMS は `pglogical` を使用してレプリケーションスロットを作成します。それ以外の場合は、`test_decoding` プラグインを使用します。

Microsoft Azure for PostgreSQL フレキシブルサーバーを DMS のソースエンドポイントとして設定するには、次の手順を実行します。

1. ポータルの [Server Parameters] ページを開きます。
2. `wal_level` サーバーパラメータを `LOGICAL` に設定します。
3. `pglogical` の拡張機能を使用する場合は、`shared_preload_libraries` パラメータと `azure.extensions` パラメータを `pglogical` に設定します。
4. `max_replication_slots` パラメータは、同時に実行する予定の DMS タスクの最大数に設定します。Microsoft Azure でのこのパラメータのデフォルト値は 10 です。このパラメータの最大値は PostgreSQL インスタンスの使用可能なメモリにより異なります。メモリ 1 GB あたり 2 ~ 8 のレプリケーションスロットを使用できます。
5. `max_wal_senders` パラメータを 1 以上の値に設定します。`max_wal_senders` パラメータは、実行可能な同時タスクの数を設定します。デフォルト値は 10 です。
6. `max_worker_processes` パラメータ値を 16 以上に設定します。この設定以外の場合は、次のようなエラーが表示されることがあります。

   ```
   WARNING: out of background worker slots.
   ```

7. 変更を保存します。サーバーを再起動して変更を適用します。
8. PostgreSQL インスタンスが接続するリソースからのネットワークトラフィックを許可していることを確認します。
9. 次のコマンドを使用して、既存のユーザーにレプリケーションのアクセス許可を付与するか、レプリケーションのアクセス許可を持つ新しいユーザーを作成します。

- 次のコマンドを使用して、既存のユーザーにレプリケーションのアクセス許可を付与します。

```
ALTER USER <existing_user> WITH REPLICATION;
```

- 次のコマンドを使用して、レプリケーションのアクセス許可を持つ新しいユーザーを作成します。

```
CREATE USER aws_dms_user PASSWORD 'aws_dms_user_password';
GRANT azure_pg_admin to aws_dms_user;
ALTER ROLE aws_dms_user REPLICATION LOGIN;
```

PostgreSQL を使用した論理レプリケーションの詳細については、次のトピックを参照してください。

- 論理レプリケーションを使用した変更データ キャプチャ (CDC) の有効化
- ネイティブ CDC スタートポイントを使用して PostgreSQL ソース エンドポイントの CDC ロードを設定するには
- 「Azure Database for PostgreSQL のドキュメント」の「 Azure Database for PostgreSQL - フレキシブル サーバーでの論理レプリケーションと論理デコード」

AWS DMS のソースとしての Microsoft Azure Database for MySQL フレキシブル サーバーの使用

AWS DMS では、Microsoft Azure Database for MySQL フレキシブルサーバーを MySQL とほとんど同じ方法でソースとして使用できます。

AWS DMS がソースとしてサポートする Microsoft Azure Database for MySQL フレキシブルサーバーのバージョンについては、「のソース AWS DMS」を参照してください。

AWS DMS でカスタマーマネージドの MySQL 互換データベースを使用する方法の詳細については、「自己管理型のMySQL互換データベースをソースとして使用する AWS DMS」を参照してください。

AWS Database Migration Service で Azure MySQL を ソースとして使用する場合の制限

- Azure MySQL フレキシブルサーバーのシステム変数 `sql_generate_invisible_primary_key` のデフォルト値は `ON` です。サーバーは、明示的なプライマリキーなしで作成されたテーブルに、生成された不可視プライマリキー (GIPK) を自動的に追加します。AWS DMS は、GIPK 制約のある MySQL テーブルについて継続的なレプリケーションをサポートしていません。

AWS DMS のソースとしての OCI MySQL Heatwave の使用

AWS DMS では、OCI MySQL Heatwave を MySQL とほとんど同じ方法でソースとして使用できます。OCI MySQL Heatwave をソースとして使用するには、いくつかの追加の設定変更が必要です。

AWS DMS がソースとしてサポートする OCI MySQL Heatwave のバージョンについては、「のソース AWS DMS」を参照してください。

論理レプリケーションのための OCI MySQL Heatwave のセットアップ

OCI MySQL Heatwave インスタンスを DMS のソースエンドポイントとして設定するには、次を実行します。

1. OCI Console にサインインして、左上隅にあるメインのハンバーガーメニュー (≡) を開きます。
2. [データベース]、[DB Systems] の順に選択します。
3. [設定] メニューを開きます。
4. [設定の作成] をクリックします。
5. 設定名 (など) を入力します。例えば、**`dms_configuration`** と入力します。
6. 現在作業中の OCI MySQL Heatwave インスタンスのシェイプを選択します。シェイプは、インスタンスの [DB system configuration:Shape] セクションの下にある [DB system configuration] プロパティタブで確認できます。
7. [User variables] セクションで、`binlog_row_value_options` システム変数を選択します。デフォルト値は `PARTIAL_JSON` です。この値を消去します。
8. [作成] ボタンをクリックします。
9. OCI MySQLHeatwave インスタンスを開き、[編集] ボタンをクリックします。
10. [設定] セクションで、[Change configuration] ボタンをクリックして、ステップ 4 で作成したシェープの設定を選択します。

11. 変更が有効になると、インスタンスの論理レプリケーションの準備が整います。

AWS DMS でのソースとしての Google Cloud for MySQL の使用

AWS DMS では、Google Cloud for MySQL を MySQL とほとんど同じ方法でソースとして使用できます。

AWS DMS がソースとしてサポートする GCP MySQL のバージョンについては、「のソース AWS DMS」を参照してください。

詳細については、「MySQL 互換データベースの AWS DMSのソースとしての使用」を参照してください。

> **Note**
>
> AWS DMS バージョン 3.4.6 以降では、GCP MySQL 8.0 がソースとしてサポートされます。
> AWS DMSは GCP for MySQL `verify-full` インスタンスの SSL モードをサポートしていません。
> GCP MySQL のセキュリティ設定 `Allow only SSL connections` は、サーバーとクライアントの両方の証明書の検証が必要となるため、サポートされていません。AWS DMS では、サーバー証明書の検証のみがサポートされています。
> AWS DMS は、`binlog_checksum` データベースフラグのデフォルトの GCP CloudSQL for MySQL 値の `CRC32` をサポートしています。

AWS DMS でのソースとしての Google Cloud for PostgreSQL の使用

AWS DMS では、Google Cloud for PostgreSQL をセルフマネージド型 PostgreSQL データベースとほとんど同じ方法でソースとして使用できます。

AWS DMS がソースとしてサポートする GCP PostgreSQL のバージョンについては、「のソース AWS DMS」を参照してください。

詳細については、「AWS DMS ソースとしての PostgreSQL データベースの使用」を参照してください。

論理レプリケーションとデコードのための Google Cloud for PostgreSQL のセットアップ

データベースの移行中に、Google Cloud SQL for PostgreSQL の論理レプリケーションとデコード機能を使用できます。

論理デコードには、DMS は 次のプラグインのどちらかを使用します。

- `test_decoding`
- `pglogical`

`pglogical` プラグインがソースのPostgreSQL データベースで利用できる場合、DMS は `pglogical` を使用してレプリケーションスロットを作成します。それ以外の場合は、`test_decoding` プラグインを使用します。

AWS DMS での論理デコード機能の使用に際して、次の点に注意する必要があります。

1. Google Cloud SQL for PostgreSQL の場合、`cloudsql.logical_decoding` フラグを on に設定して、論理デコード機能を有効にします。
2. `pglogical` を有効にするには、`cloudsql.enable_pglogical` フラグを on に設定して、データベースを再起動します。
3. 論理デコード機能を使用するには、REPLICATION 属性を持つ PostgreSQL ユーザーを作成します。`pglogical` 拡張機能を使用する場合、ユーザーには `cloudsqlsuperuser` ロールが必要です。`cloudsqlsuperuser` ロールを持つユーザーを作成するには次を実行します。

   ```
   CREATE USER new_aws_dms_user WITH REPLICATION
   IN ROLE cloudsqlsuperuser LOGIN PASSWORD 'new_aws_dms_user_password';
   ```

   既存のユーザーにこの属性を設定するには、次を実行します。

   ```
   ALTER USER existing_user WITH REPLICATION;
   ```

4. `max_replication_slots` パラメータは、同時に実行する予定の DMS タスクの最大数に設定します。Google Cloud SQL では、このパラメータのデフォルト値は 10 です。このパラメータの最大値は PostgreSQL インスタンスの使用可能なメモリにより異なります。メモリ 1 GB あたり 2~8 のレプリケーションスロットを使用できます。

PostgreSQL を使用した論理レプリケーションの詳細については、次のトピックを参照してください。

- 論理レプリケーションを使用した変更データ キャプチャ (CDC) の有効化
- ネイティブ CDC スタートポイントを使用して PostgreSQL ソース エンドポイントの CDC ロードを設定するには
- 「CLOUD SQL FOR POSTGRESQL のドキュメント」の「論理レプリケーションとデコードを設定する」

AWS DMS ソースとしての PostgreSQL データベースの使用

を使用して、1 つまたは複数の PostgreSQL データベースからデータを移行できます AWS DMS。PostgreSQL データベースをソースとして使用する場合、別の PostgreSQL データベースまたはサポートされているその他の データベースのいずれかにデータを移行できます。

がソースとして AWS DMS サポートする PostgreSQL のバージョンについては、「」を参照してくださいのソース AWS DMS。

AWS DMS は、次のタイプのデータベースに対して PostgreSQL をサポートしています。

- オンプレミスのデータベース
- Amazon EC2 インスタンスでのデータベース
- Amazon RDS DB インスタンス上のデータベース
- Amazon Aurora PostgreSQL 互換エディションに基づく DB インスタンス上のデータベース
- Amazon Aurora PostgreSQL 互換 Serverless エディションを基盤とする DB インスタンス上のデータベース

> **Note**
>
> DMS は Amazon Aurora PostgreSQL—Serverless V1 についてはフルロードのソースとしてのみサポートしています。ただし、Amazon Aurora PostgreSQL—Serverless V2 は、フルロード、フルロード + CDC、CDC のみのタスクのソースとして使用することができます。

AWS DMS 使用する バージョン
使用可能なバージョンを使用します AWS DMS 。
AWS DMS バージョン 3.4.3 以降を使用します。
AWS DMS バージョン 3.4.7 以降を使用します。
AWS DMS バージョン 3.5.1 以降を使用します。
AWS DMS バージョン 3.5.3 以降を使用します。

Secure Sockets Layer (SSL) を使用して、PostgreSQL エンドポイントとレプリケーション インスタンスとの接続を暗号化できます。PostgreSQL エンドポイントで SSL を使用する方法の詳細については、「での SSL の使用 AWS Database Migration Service」をご参照ください。

ソースとして PostgreSQL を使用する場合の追加セキュリティ要件として、指定されるユーザーアカウントは PostgreSQL データベースの登録済みユーザーでなければなりません。

PostgreSQL データベースを AWS DMS ソースエンドポイントとして設定するには、次の手順を実行します。

- PostgreSQL ソースデータベース AWS DMS へのアクセスを提供する適切なアクセス許可を持つ PostgreSQL ユーザーを作成します。

> **Note**
>
> - PostgreSQL ソースデータベースが自己管理型の場合は詳細については、「でのソースとしてのセルフマネージド PostgreSQL データベースの使用 AWS DMS」をご参照ください。

- PostgreSQL ソースデータベースが Amazon RDS によって管理されている場合詳細については、「DMS ソースとしての AWSマネージド PostgreSQL データベースの使用」をご参照ください。

- 選択した PostgreSQL データベース設定に準拠する PostgreSQL ソース エンドポイントを作成します。
- テーブルを移行するタスクまたはタスクのセットを作成します。

  full-load-only タスクを作成するには、それ以上のエンドポイント設定は必要ありません。

  変更データキャプチャのタスク(CDC のみ、または全ロードおよび CDC タスク)を作成する前に、「セルフマネージド PostgreSQL データベースをソースとして使用して CDC を有効にする AWS DMS」または「で AWSマネージド PostgreSQL DB インスタンスで CDC を有効にする AWS DMS」をご参照ください。

トピック

- でのソースとしてのセルフマネージド PostgreSQL データベースの使用 AWS DMS
- DMS ソースとしての AWSマネージド PostgreSQL データベースの使用
- 論理レプリケーションを使用した変更データ キャプチャ (CDC) の有効化
- ネイティブ CDC スタートポイントを使用して PostgreSQL ソース エンドポイントの CDC ロードを設定するには
- を使用した PostgreSQL から PostgreSQL への移行 AWS DMS
- を使用した Babelfish for Amazon Aurora PostgreSQL からの移行 AWS DMS
- PostgreSQL ソースデータベースからの AWS DMS アーティファクトの削除
- DMS ソースとして PostgreSQL データベースを使用する場合の追加設定
- MapBooleanAsBoolean PostgreSQL エンドポイント設定の使用
- PostgreSQL を DMS ソースとして使用する場合のエンドポイント設定と追加の接続属性 (ECAs)
- DMS のソースとして PostgreSQL データベースを使用する場合の制限
- PostgreSQL のソースデータ型

でのソースとしてのセルフマネージド PostgreSQL データベースの使用 AWS DMS

セルフマネージド型の PostgreSQL データベースをソースとして使用すると、別の PostgreSQL データベース、または でサポートされている他のターゲットデータベースのいずれかにデータを移行で

きます AWS DMS。データベースソースには、オンプレミスのデータベースまたは Amazon EC2 インスタンスで実行されているセルフ管理エンジンを使用できます。DB インスタンスは、全ロードタスクと継続的レプリケーションの変更データキャプチャ (CDC) タスクの両方に使用できます。

セルフマネージド PostgreSQL データベースを AWS DMS ソースとして使用する前提条件

自己管理 PostgreSQL ソースデータベースからデータを移行する前に、次の操作を行います：

- バージョン 9.4.x 以降の PostgreSQL データベースを使用していることを確認する。
- 全ロードと CDC タスクまたは CDC のみのタスクの場合は、PostgreSQL ソースデータベースに指定されたユーザーアカウントにスーパーユーザー許可を付与します。ユーザー アカウントはソース内のレプリケーション固有機能にアクセスするには、スーパーユーザー許可が必要です。全ロードのみのタスクの場合、それらを移行するには、ユーザー アカウントはテーブルで SELECT 許可が必要です。
- AWS DMS レプリケーションサーバーの IP アドレスを設定pg_hba.confファイルに追加し、レプリケーションとソケット接続を有効にします。以下に例を示します。

```
# Replication Instance
host all all 12.3.4.56/00 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
host replication dms 12.3.4.56/00 md5
```

  PostgreSQL の pg_hba.conf の設定ファイルはクライアント認証を制御します。(HBA はホストベースの認証を表します) ファイルは従来、データベースクラスターのデータディレクトリに保存されています。

- を使用して論理レプリケーションのソースとしてデータベースを設定する場合は、 AWS DMS 「」を参照してください。 セルフマネージド PostgreSQL データベースをソースとして使用して CDC を有効にする AWS DMS

> **Note**
>
> 一部の AWS DMS トランザクションは、DMS エンジンが再度使用するまでしばらくアイドル状態になっています。PostgreSQL バージョン 9.6 以降で idle_in_transaction_session_timeout パラメータを使用すると、アイドル状態の

トランザクションでタイムアウトやエラーが生じる場合があります。 AWS DMSを使用する際、アイドル状態のトランザクションを終了しないでください。

セルフマネージド PostgreSQL データベースをソースとして使用して CDC を有効にする AWS DMS

AWS DMS は、論理レプリケーションを使用した変更データキャプチャ (CDC) をサポートします。セルフ管理 PostgreSQL ソースデータベースの論理レプリケーションを有効にするには、`postgresql.conf` の設定ファイルで以下のパラメータと値を設定します:

- `wal_level = logical` を設定します。
- `max_replication_slots` を 1 より大きい値に設定します。

  `max_replication_slots` 値は、実行するタスクの数に従って設定してください。たとえば、5 つのタスクを実行するには、少なくとも 5 つのスロットを設定する必要があります。スロットは、タスクが開始するとすぐに自動的に開き、タスクが実行されなくなった場合でも開いたままです。開いているスロットは手動で削除してください。DMS でレプリケーションスロットを作成した場合、タスクを削除すると、DMS は自動的にレプリケーションスロットを削除することに注意してください。
- `max_wal_senders` を 1 より大きい値に設定します。

  `max_wal_senders` パラメータは、実行可能な同時タスクの数を設定します。
- `wal_sender_timeout` パラメータは、指定されたミリ秒数が過ぎても非アクティブなレプリケーション接続を終了します。オンプレミスの PostgreSQL データベースのデフォルトは 60,000 ミリ秒 (60 秒)。値を 0 (ゼロ) に設定するとタイムアウトメカニズムが無効になる。この設定は DMS では有効。

  `wal_sender_timeout` をゼロ以外の値に設定すると、CDC での DMS タスクには最低 10,000 ミリ秒 (10 秒) が必要となり、値が 10,000 ミリ秒未満になると失敗する。DMS レプリケーションインスタンスのマルチ AZ フェイルオーバー中に遅延が発生しないように、この値は 5 分未満にする。

一部のパラメータは静的であり、サーバースタート時にのみ設定できます。設定ファイル (セルフマネージドデータベースの場合) または DB パラメータグループ (RDS for PostgreSQL データベース) への変更は、サーバーが再起動されるまで無視されます。詳細については、PostgreSQL ドキュメントをご参照ください。

CDC を有効にすることに関する詳細については、「論理レプリケーションを使用した変更データキャプチャ (CDC) の有効化」をご参照ください。

DMS ソースとしての AWSマネージド PostgreSQL データベースの使用

AWSマネージド PostgreSQL DB インスタンスを のソースとして使用できます AWS DMS。 AWSが管理する PostgreSQL ソースを使用して、全ロードタスクと変更データキャプチャ (CDC) タスクの両方を実行できます。

AWSが管理する PostgreSQL データベースを DMS ソースとして使用するための前提条件

AWS管理の PostgreSQL ソースデータベースからデータを移行する前に、次の操作を行います。

- の PostgreSQL ソースエンドポイントの AWS ユーザーアカウントとして、PostgreSQL DB インスタンスに必要な最小限のアクセス許可を持つユーザーアカウントを使用することをお勧めします AWS DMS。管理アカウントの使用はお勧めしません。このアカウントには `rds_superuser` ロールと `rds_replication` ロールが必要です。`rds_replication` ロールは、論理スロットを管理し、論理スロットを使用してデータをストリーミングするアクセス権許可付与します。

  使用するアカウントの管理ユーザーアカウントで複数のオブジェクトを作成します。その作成についての詳細は、「マスター ユーザーアカウントを使用せず Amazon RDS for PostgreSQL データベースの移行」をご参照ください。
- ソースデータベースが仮想プライベートクラウド (VPC) にある場合は、データベースが常駐する DB インスタンスへのアクセス権を提供する VPC セキュリティグループを選択します。これは、DMS レプリケーション インスタンスがソース DB インスタンスに正常に接続するために必要です。データベースと DMS レプリケーション インスタンスが同じ VPC 内にある場合は、適切なセキュリティグループを独自のインバウンド ルールに追加します。

> **Note**
>
> 一部の AWS DMS トランザクションは、DMS エンジンが再度使用するまでしばらくアイドル状態になっています。PostgreSQL バージョン 9.6 以降で `idle_in_transaction_session_timeout` パラメータを使用すると、アイドル状態のトランザクションでタイムアウトやエラーが生じる場合があります。 AWS DMSを使用する際、アイドル状態のトランザクションを終了しないでください。

で AWSマネージド PostgreSQL DB インスタンスで CDC を有効にする AWS DMS

AWS DMS DB インスタンスが論理レプリケーションを使用するように設定されている場合、 は Amazon RDS PostgreSQL データベースで CDC をサポートします。次の表は、 が AWS管理する各 PostgreSQL バージョンの論理レプリケーションの互換性をまとめたものです。

CDC (継続的レプリケーション) には RDS for PostgreSQL リードレプリカを使用できません。

PostgreSQL バージョン	AWS DMS フルロードサポート	AWS DMS CDC サポート
Aurora PostgreSQL バージョン 2.1 (PostgreSQL 10.5 以前と互換)	はい	いいえ
PostgreSQL 10.6 と互換性のある Aurora PostgreSQL バージョン 2.2 (またはそれ以降)	はい	はい
PostgreSQL 10.21 と互換性のある RDS for PostgreSQL (またはそれ以降)	はい	はい

RDS PostgreSQL DB インスタンスに対して論理レプリケーションを有効にするには

1. PostgreSQL DB インスタンスの AWS マスターユーザーアカウントを PostgreSQL ソースエンドポイントのユーザーアカウントとして使用します。マスターユーザーアカウントには、CDC を設定するために必要なロールがあります。

   マスター ユーザーアカウント以外のアカウントを使用する場合は、使用するアカウントのマスター ユーザーアカウントから複数のオブジェクトを作成する必要があります。詳細については、「マスター ユーザーアカウントを使用せず Amazon RDS for PostgreSQL データベースの移行」を参照してください。
2. DB CLUSTER パラメータグループの `rds.logical_replication` パラメータを 1 に設定します。この静的パラメータを有効にするには、DB インスタンスを再起動する必要があります。このパラメータを適用する一環として、 AWS DMS は `wal_level`、`max_wal_senders`、`max_replication_slots`、`max_connections` の各パラメータを設定します。これらのパラメータの変更によって先書き口グ (WAL) の

生成が増える可能性があるため、論理レプリケーションスロットを使用する場合にのみ rds.logical_replication を設定してください。

3. wal_sender_timeout パラメータは、指定されたミリ秒数が過ぎても非アクティブなレプリケーション接続を終了します。 AWSマネージド PostgreSQL データベースのデフォルトは 30000 ミリ秒 (30 秒) です。値を 0 (ゼロ) に設定するとタイムアウトメカニズムが無効になる。この設定は DMS では有効です。

   wal_sender_timeout をゼロ以外の値に設定すると、CDC での DMS タスクには最低 10,000 ミリ秒 (10 秒) が必要となり、値が 0 ~ 10,000 ミリ秒の場合は失敗します。DMS レプリケーションインスタンスのマルチ AZ フェイルオーバー中に遅延が発生しないように、この値は 5 分未満にします。

4. DB クラスター パラメータグループで max_worker_processes パラメータの値が、max_logical_replication_workers、autovacuum_max_workers、max_parallel_work 合計値以上となるようにしてください。多数のバックグラウンド ワーカー プロセスが、スモールインスタンスではアプリケーション ワークロードに影響を与える可能性があります。したがって、max_worker_processes をデフォルト値より大きく設定した場合は、データベースのパフォーマンスをモニタリングしてください。

5. Aurora PostgreSQL を CDC のソースとして使用する場合は、 synchronous_commitを に設定しますON。

マスター ユーザーアカウントを使用せず Amazon RDS for PostgreSQL データベースの移行

場合によっては、ソースとして使用している Amazon RDS PostgreSQL DB インスタンス用マスターユーザーアカウントを使用しない場合があります。このような場合は、データ定義言語 (DDL) イベントをキャプチャするために複数のオブジェクトを作成します。マスターアカウント以外のアカウントでこれらのオブジェクトを作成し、マスターユーザーアカウントでトリガーを作成します。

> **Note**
>
> ソースエンドポイントで captureDDLs エンドポイント設定を false に設定すると、ソースデータベース上で次のテーブルおよびトリガーを作成する必要はありません。

これらのオブジェクトを作成するには、以下の手順を実行します。

オブジェクトを作成するには

1. オブジェクトが作成されるスキーマを選択します。デフォルトのスキーマは public です。スキーマが存在し、*OtherThanMaster* アカウントからアクセス可能であることを確認します。
2. マスターアカウント以外のユーザーアカウントを使用して PostgreSQL DB インスタンス、ここでは *OtherThanMaster* アカウントにログインします。
3. 以下のコマンドを実行し、以下のコード内の *objects_schema* を使用するスキーマの名前に置き換えて awsdms_ddl_audit テーブルを作成します。

```
CREATE TABLE objects_schema.awsdms_ddl_audit
(
  c_key    bigserial primary key,
  c_time   timestamp,    -- Informational
  c_user   varchar(64),  -- Informational: current_user
  c_txn    varchar(16),  -- Informational: current transaction
  c_tag    varchar(24),  -- Either 'CREATE TABLE' or 'ALTER TABLE' or 'DROP TABLE'
  c_oid    integer,      -- For future use - TG_OBJECTID
  c_name   varchar(64),  -- For future use - TG_OBJECTNAME
  c_schema varchar(64),  -- For future use - TG_SCHEMANAME. For now - holds
 current_schema
  c_ddlqry  text         -- The DDL query associated with the current DDL event
);
```

4. 以下のコマンドを実行して awsdms_intercept_ddl 関数を作成します。コード内の *objects_schema* は、使用するスキーマの名前に置き換えてください。

```
CREATE OR REPLACE FUNCTION objects_schema.awsdms_intercept_ddl()
  RETURNS event_trigger
LANGUAGE plpgsql
SECURITY DEFINER
  AS $$
  declare _qry text;
BEGIN
  if (tg_tag='CREATE TABLE' or tg_tag='ALTER TABLE' or tg_tag='DROP TABLE' or
 tg_tag = 'CREATE TABLE AS') then
         SELECT current_query() into _qry;
         insert into objects_schema.awsdms_ddl_audit
```

```
        values
        (
        default,current_timestamp,current_user,cast(TXID_CURRENT()as
 varchar(16)),tg_tag,0,'',current_schema,_qry
        );
        delete from objects_schema.awsdms_ddl_audit;
end if;
END;
$$;
```

5. *OtherThanMaster* アカウントからログアウトし、rds_superuser ロールが割り当てられたアカウントを使用してログインします。
6. 以下のコマンドを実行してイベントトリガー awsdms_intercept_ddl を作成します。

```
CREATE EVENT TRIGGER awsdms_intercept_ddl ON ddl_command_end
EXECUTE PROCEDURE objects_schema.awsdms_intercept_ddl();
```

7. これらのイベントにアクセスするすべてのユーザーおよびロールに必要な DDL 許可があることを確認します。例:

```
grant all on public.awsdms_ddl_audit to public;
grant all on public.awsdms_ddl_audit_c_key_seq to public;
```

前の手順を完了したら、*OtherThanMaster* アカウントを使用して AWS DMS ソースエンドポイントを作成できます。

> **Note**
>
> これらのイベントは、CREATE TABLE、ALTER TABLE、および DROP TABLE ステートメントによってトリガーされます。

論理レプリケーションを使用した変更データ キャプチャ (CDC) の有効化

PostgreSQL のネイティブ論理レプリケーション機能を使用して、PostgreSQL DB ソース用データベース移行中に変更データ キャプチャ (CDC) を有効にすることができます。この機能は、セルフ管

理 PostgreSQL および Amazon RDS for PostgreSQL SQL DB インスタンスで使用できます。この方法では、ダウンタイムが短縮され、ターゲットデータベースがソース PostgreSQL データベースと同期しやすくなります。

AWS DMS は、プライマリキーを持つ PostgreSQL テーブルの CDC をサポートします。テーブルにプライマリ キーがない場合、先書きログ (WAL) にはデータベース行の前イメージが含まれません。この場合、DMS はテーブルを更新できません。ここでは、追加の構成設定を使用し、回避策としてテーブルレプリカ アイデンティティを使用できます。ただし、この方法では追加の口グが生成される可能性があります。注意深いテストの後にのみ、回避策としてテーブルレプリカ アイデンティティ を使用することをお勧めします。詳細については、「DMS ソースとして PostgreSQL データベースを使用する場合の追加設定」を参照してください。

> **Note**
>
> REPLICA IDENTITY FULL は論理デコードプラグインではサポートされていますが、pglogical プラグインではサポートされていません。詳細については、「pglogical ドキュメント」を参照してください。

全ロード、CDC、CDC のみのタスクの場合、 は論理レプリケーションスロット AWS DMS を使用して、ログがデコードされるまで、レプリケーション用の WAL ログを保持します。全ロードおよび CDC タスクまたは CDC タスクの再起動(再開ではない)によって、レプリケーション スロットが再作成されます。

> **Note**
>
> 論理デコードの場合、DMS は test_decoding または pglogical プラグインのいずれかを使用します。pglogical プラグインがソース PostgreSQL データベースで使用できる場合、DMS は pglogical を使用してレプリケーション スロットを作成し、それ以外の場合は test_decoding プラグインが使用されます。test_decoding プラグインの詳細については、「PostgreSQL のドキュメント」を参照してください。
>
> データベースパラメータ `max_slot_wal_keep_size` がデフォルト以外の値に設定されており、レプリケーションスロットの `restart_lsn` の値が現在の LSN よりも大きい場合、必要な WAL ファイルが削除されるため、DMS タスクは失敗します。

pgglogical プラグインの設定

PostgreSQL 拡張機能として実装された pglogical プラグインは、選択的データ レプリケーション用論理レプリケーションシステムとモデルです。次の表に、pglogical プラグインをサポートするソース PostgreSQL データベースバージョンを挙げます。

PostgreSQL ソース	plogical のサポート
セルフ管理 PostgreSQL 9.4 以降	はい
Amazon RDS PostgreSQL 9.5 以降	いいえ
Amazon RDS PostgreSQL 9.6 以降	はい
Aurora PostgreSQL 1.x から 2.5.x まで	いいえ
Aurora PostgreSQL 2.6 以上	はい
Aurora PostgreSQL 3.3 以上	はい

で使用するように pglogical を設定する前に AWS DMS、まず PostgreSQL ソースデータベースで変更データキャプチャ (CDC) の論理レプリケーションを有効にします。

- セルフ管理 PostgreSQL ソースデータベースにおいて CDC での論理レプリケーションの有効化について詳しくは、「セルフマネージド PostgreSQL データベースをソースとして使用して CDC を有効にする AWS DMS」をご参照ください。
- AWSが管理するPostgreSQL ソースデータベースでの CDC 用論理レプリケーションの有効化については、「で AWSマネージド PostgreSQL DB インスタンスで CDC を有効にする AWS DMS」をご参照ください。

PostgreSQL ソースデータベースで論理レプリケーションを有効にした後、次のステップに従って、DMS で使用する pglogical を設定します。

で PostgreSQL ソースデータベースの論理レプリケーションに pglogical プラグインを使用するには AWS DMS

1. PostgreSQL ソースデータベースに pglogical 拡張機能を作成します：

    a. 正しいパラメータを設定します：

- セルフ管理 PostgreSQL データベースの場合は、データベースパラメータ `shared_preload_libraries= 'pglogical'` を設定します。
- Amazon RDS と Amazon Aurora PostgreSQL 互換エディションの PostgreSQL で、パラメーター `shared_preload_libraries` を同じ RDS パラメーターグループ内の `pglogical` に設定します。

b. PostgreSQL ソースデータベースを再起動します。

c. PostgreSQL データベースで、コマンド `create extension pglogical;` を実行します。

2. pglogical が正常にインストールされたことを確認するには、次のコマンドを実行します：

```
select * FROM pg_catalog.pg_extension
```

PostgreSQL ソースデータベースエンドポイントの変更データキャプチャを実行する AWS DMS タスクを作成できるようになりました。

> **Note**
>
> PostgreSQL ソースデータベースで pglogical を有効にしないとき、 AWS DMS はデフォルトで `test_decoding` プラグインを使用します。pglogical が論理デコードに対して有効になっている場合、 はデフォルトで pglogical AWS DMS を使用します。ただし、代わりに `test_decoding` のプラグインを使用する追加の接続属性 `PluginName` を設定することもできます。

ネイティブ CDC スタートポイントを使用して PostgreSQL ソース エンドポイントの CDC ロードを設定するには

エンドポイントを作成するときに、`slotName` の追加の接続属性を既存の論理レプリケーション スロットの名前に設定することで、ソースとして PostgreSQL によりネイティブ CDC スタートポイントを有効にします。この論理レプリケーションスロットは、エンドポイントの作成時点からの継続的な変更を保持するため、以前の時点からのレプリケーションをサポートします。

PostgreSQL は、 AWS DMS が論理レプリケーションスロットから変更を正常に読み取った後にのみ破棄される WAL ファイルにデータベースの変更を書き込みます。論理レプリケーションスロットを使用すると、ログに記録された変更がレプリケーションエンジンによって消費される前に削除されないように保護できます。

ただし、変更率と消費率によっては、論理レプリケーションスロットに保持されている変更により、ディスク使用率が高くなることがあります。論理レプリケーション スロットを使用する場合は、ソース PostgreSQL インスタンスにスペース使用アラームを設定することをお勧めします。追加の slotName 接続属性の設定についての詳細は、「PostgreSQL を DMS ソースとして使用する場合のエンドポイント設定と追加の接続属性 (ECAs)」をご参照ください。

次の手順では、このアプローチについて詳しく説明します。

ネイティブ CDC 開始ポイントを使用して PostgreSQL ソースエンドポイントの CDC ロードを設定するには

1. 開始点として使用する以前のレプリケーションタスク (親タスク) によって使用されていた論理レプリケーションスロットを特定します。次に、ソースデータベースの pg_replication_slots ビューにクエリを実行して、このスロットにアクティブな接続がないことを確認します。その場合は、先に進む前に解決して終了します。

   次のステップでは、論理レプリケーションスロットが abc1d2efghijk_34567890_z0yx98w7_6v54_32ut_1srq_1a2b34c5d67ef であると仮定します。
2. 次の追加接続属性設定を含む新しいソースエンドポイントを作成します。

   ```
   slotName=abc1d2efghijk_34567890_z0yx98w7_6v54_32ut_1srq_1a2b34c5d67ef;
   ```
3. コンソール AWS CLI または AWS DMS API を使用して、新しい CDC 専用タスクを作成します。たとえば、CLI を使用して、次の create-replication-task コマンドを実行できます。

   ```
   aws dms create-replication-task --replication-task-identifier postgresql-slot-name-
   test
   --source-endpoint-arn arn:aws:dms:us-
   west-2:012345678901:endpoint:ABCD1EFGHIJK2LMNOPQRST3UV4
   --target-endpoint-arn arn:aws:dms:us-
   west-2:012345678901:endpoint:ZYX9WVUTSRQONM8LKJIHGF7ED6
   --replication-instance-arn arn:aws:dms:us-
   west-2:012345678901:rep:AAAAAAAAAAA5BB4CCC3DDDD2EE
   --migration-type cdc --table-mappings "file://mappings.json" --cdc-start-position
    "4AF/B00000D0"
   --replication-task-settings "file://task-pg.json"
   ```

   前述のコマンドでは、次のオプションが設定されています。

- source-endpoint-arn は、ステップ 2 で作成した新しい値に設定されます。
- replication-instance-arn オプションは、ステップ 1 の親タスクと同じ値に設定されます。
- table-mappings および replication-task-settings オプションは、ステップ 1 の親タスクと同じ値に設定されます。
- cdc-start-position はオプションはスタート位置の値に設定されます。この開始位置を調べるには、ソースデータベースの pg_replication_slots ビューにクエリを実行するか、ステップ 1 で親タスクのコンソール詳細を表示します。詳細については、「CDC ネイティブ開始点の決定」を参照してください。

AWS DMS コンソールを使用して新しい CDC 専用タスクを作成するときにカスタム CDC 開始モードを有効にするには、次の手順を実行します。

- [タスク設定] セクションの [ソーストランザクションの CDC 開始モード] では、[カスタム CDC 開始モードを有効にする] を選択する。
- [ソーストランザクションのカスタム CDC 開始点] では、[ログシーケンス番号を指定する] を選択する。システム変更番号を指定するか、[復旧チェックポイントを指定する] を選択して、復旧チェックポイントを指定する。

この CDC タスクが実行されると、指定された論理レプリケーションスロットが存在しない場合、 はエラーを AWS DMS 解決します。また、cdc-start-position の有効な設定を使用してタスクが作成されていない場合、エラーを返します。

pglogical プラグインでネイティブ CDC スタートポイントを使用し、新しいレプリケーション スロットを使用する場合は、CDC タスクを作成する前に、次のステップを実行してください。

別の DMS タスクの一部として以前に作成されていない新しいレプリケーション スロットを使用するには

1. 次に示すように、レプリケーション スロットを作成します：

```
SELECT * FROM pg_create_logical_replication_slot('replication_slot_name',
 'pglogical');
```

2. データベースがレプリケーションスロットを作成した後、次のとおりスロットの restart_lsn と confirmed_flush_lsn の値を取得してメモしておきます。

```
select * from pg_replication_slots where slot_name like 'replication_slot_name';
```

レプリケーションスロットの後に作成された CDC タスクのネイティブな CDC 開始点は、confirmed_flush_lsn 値以前にできないことに注意します。

restart_lsn と confirmed_flush_lsn の値の詳細については、「pg_replication_slots」を参照してください。

3. pglogical ノードを作成します。

```
SELECT pglogical.create_node(node_name := 'node_name', dsn := 'your_dsn_name');
```

4. pglogical.create_replication_set 関数を使用して 2 つのレプリケーションセットを作成します。最初のレプリケーションセットは、プライマリキーを持つテーブルの更新と削除を追跡します。2 番目のレプリケーションセットは挿入のみを追跡し、最初のレプリケーションセットと同じ名前にプレフィックス「i」が追加されています。

```
SELECT pglogical.create_replication_set('replication_slot_name', false, true, true,
 false);
SELECT pglogical.create_replication_set('ireplication_slot_name', true, false,
 false, true);
```

5. レプリケーション集合にテーブルを追加します。

```
SELECT pglogical.replication_set_add_table('replication_slot_name',
 'schemaname.tablename', true);
SELECT pglogical.replication_set_add_table('ireplication_slot_name',
 'schemaname.tablename', true);
```

6. ソース エンドポイントを作成するときに、次の追加接続属性 (ECA) を設定します。

```
PluginName=PGLOGICAL;slotName=slot_name;
```

新しいレプリケーション スロットを使用して PostgreSQL ネイティブのスタートポイントを使用して CDC のみのタスクを作成できるようになりました。pglogical プラグインの詳細については、「pglogical 3.7 ドキュメント」を参照してください。

を使用した PostgreSQL から PostgreSQL への移行 AWS DMS

PostgreSQL 以外のデータベースエンジンから PostgreSQL データベースに移行する場合、 AWS DMS はほとんどの場合、最適な移行ツールです。一方、PostgreSQL データベースから PostgreSQL データベースに移行する場合、PostgreSQL ツールがより効果的な場合があります。

PostgreSQL ネイティブ ツールを使用してデータを移行する

以下の条件では、pg_dump などの PostgreSQL データベース移行ツールを使用することをお勧めします。

- ソース PostgreSQL データベースからターゲット PostgreSQL データベースに移行する同種移行である。
- データベース全体を移行する。
- ネイティブツールで最小のダウンタイムでデータを移行できる。

pg_dump ユーティリティでは、COPY コマンドを使用して、PostgreSQL データベースのスキーマとデータダンプを作成します。pg_dump によって生成されるダンプ スクリプトは、同じ名前のデータベースにデータをロードし、テーブル、インデックス、外部キーを再作成します。`pg_restore` コマンドと `-d` パラメータを使用して、データを別の名前でデータベースに復元できます。

EC2 で実行されている PostgreSQL ソースデータベースから Amazon RDS for PostgreSQL ターゲットにデータを移行する場合は、pglogical プラグインを使用できます。

PostgreSQL データベースを Amazon RDS for PostgreSQL や Amazon Aurora (PostgreSQL 互換エディション) にインポートする詳しい方法については、「https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/PostgreSQL.Procedural.Importing.html」をご参照ください。

DMS を使用した PostgreSQL から PostgreSQL へのデータの移行

AWS DMS は、例えば、オンプレミスのソース PostgreSQL データベースからターゲット Amazon RDS for PostgreSQL または Aurora PostgreSQL インスタンスにデータを移行できます。通常、PostgreSQL のコアまたは基本のデータ型は正常に移行されます。

> **Note**
>
> パーティションテーブルを PostgreSQL ソースから PostgreSQL ターゲットにレプリケーションする場合、DMS タスクの選択条件の一部として親テーブルを言及する必要はありません。親テーブルに言及すると、ターゲット上の子テーブルでデータが複製され、PK 違反が発生する可能性があります。テーブルマッピングの選択基準で子テーブルのみを選択すると、親テーブルに自動代入されます。

ソースデータベースでサポートされているが、ターゲットでサポートされていないデータ型は、正常に移行されない場合があります。データ型が不明な場合、 は一部のデータ型を文字列として AWS DMS ストリーミングします。XML や JSON などの一部のデータ型は、小さなファイルの場合は正常に移行されますが、大きなドキュメントの場合は失敗することがあります。

データ型の移行を実行するときは、以下について注意してください：

- 場合によっては、PostgreSQL NUMERIC (p, s) データ型で精度とスケールが指定されない場合があります。DMS バージョン 3.4.2 以前では、DMS はデフォルトでは 28 の精度と 6 のスケール (NUMERIC (28,6)) を使用します。たとえば、ソースからの値 0.611111104488373 は、PostgreSQL ターゲットの 0.611111 に変換されます。
- ARRAY データ型を持つテーブルにはプライマリ キーが必要です。ARRAY データ型にプライマリキーがないテーブルは、全ロード中に中断されます。

次の表は、ソース PostgreSQL のデータ型と、これらのデータ型が正常に移行されるかどうかを示しています。

データ型	正常に移行	部分的に移行	移行しない	コメント
INTEGER	X			
SMALLINT	X			
BIGINT	X			
NUMERIC/DECIMAL(p,s)		X		この場合、0<p<39 と 0<s

データ型	正常に移行	部分的に移行	移行しない	コメント
NUMERIC/DECIMAL		X		この場合、p>38 または p=s=0
REAL	X			
DOUBLE	X			
SMALLSERIAL	X			
SERIAL	X			
BIGSERIAL	X			
MONEY	X			
CHAR		X		精度の指定なし
CHAR(n)	X			
VARCHAR		X		精度の指定なし
VARCHAR(n)	X			
TEXT	X			
BYTEA	X			
TIMESTAMP	X			正と負の無限大値はそれぞれ '9999-12-31 23:59:59 'と '4713-01-01 00:00:00 BC' に切り捨てられます。

データ型	正常に移行	部分的に移行	移行しない	コメント
TIMESTAMP WITH TIME ZONE		X		
DATE	X			
TIME	X			
TIME WITH TIME ZONE		X		
INTERVAL		X		
BOOLEAN	X			
ENUM			X	
CIDR	X			
INET			X	
MACADDR			X	
TSVECTOR			X	
TSQUERY			X	
XML		X		
POINT	X			PostGIS 空間データ型
LINE			X	
LSEG			X	
BOX			X	
PATH			X	
POLYGON	X			PostGIS 空間データ型

データ型	正常に移行	部分的に移行	移行しない	コメント
CIRCLE			X	
JSON		X		
配列	X			プライマリ キーが必要です
COMPOSITE			X	
RANGE			X	
LINESTRING	X			PostGIS 空間データ型
MULTIPOINT	X			PostGIS 空間データ型
MULTILINESTRING	X			PostGIS 空間データ型
MULTIPOLYGON	X			PostGIS 空間データ型
GEOMETRYCOLLECTION	X			PostGIS 空間データ型

PostGIS 空間データ型の移行

空間データは、空間内のオブジェクトまたは位置のジオメトリ情報を識別します。PostgreSQL オブジェクトリレーショナルデータベースは、PostGIS 空間データ型をサポートしています。

PostgreSQL 空間データオブジェクトを移行する前に、PostGIS プラグインがグローバルレベルで有効になっていることを確認してください。これにより、 は PostgreSQL ターゲット DB インスタンスの正確なソース空間データ列 AWS DMS を作成します。

PostgreSQL から PostgreSQL への同種移行の場合、 は PostGIS のジオメトリおよび地理的 (ジオデティック座標) データオブジェクトタイプとサブタイプの移行 AWS DMS をサポートします。

- POINT
- LINESTRING
- POLYGON
- MULTIPOINT
- MULTILINESTRING
- MULTIPOLYGON
- GEOMETRYCOLLECTION

を使用した Babelfish for Amazon Aurora PostgreSQL からの移行 AWS DMS

Babelfish for Aurora PostgreSQL ソーステーブルは、 を使用してサポートされている任意のターゲットエンドポイントに移行できます AWS DMS。

DMS コンソール、API、または CLI コマンドを使用して AWS DMS ソースエンドポイントを作成する場合、ソースを Amazon Aurora PostgreSQL に設定し、データベース名を に設定しますbabelfish_db。エンドポイント設定セクションで、 DatabaseModeが Babelfish に設定BabelfishDatabaseNameされ、ソース Babelfish T-SQL データベースの名前に設定されていることを確認します。Babelfish TCP ポート を使用する代わりに1433、Aurora PostgreSQL TCP ポート を使用します5432。

DMS が正しいデータ型とテーブルメタデータを使用していることを確認するには、データを移行する前にテーブルを作成する必要があります。移行を実行する前にターゲットにテーブルを作成しない場合、DMS は誤ったデータ型とアクセス許可を持つテーブルを作成することがあります。

移行タスクへの変換ルールの追加

Babelfish ソースの移行タスクを作成するときは、DMS が事前に作成されたターゲットテーブルを使用するようにするための変換ルールを含める必要があります。

Babelfish for PostgreSQL クラスターを定義したときにマルチデータベース移行モードを設定する場合は、スキーマ名を T-SQL スキーマに変更する変換ルールを追加します。例えば、T-SQL スキーマ名が でdbo、Babelfish for PostgreSQL スキーマ名が の場合mydb_dbo、変換ルールdboを使用してスキーマの名前を に変更します。PostgreSQL スキーマ名を確認するには、「Amazon Aurora ユーザーガイド」の「Babelfish アーキテクチャ」を参照してください。

シングルデータベースモードを使用する場合、変換ルールを使用してデータベーススキーマの名前を変更する必要はありません。PostgreSQL スキーマ名には、T-SQL データベース内のスキーマ名への one-to-one マッピングがあります。

次の変換ルールの例は、スキーマ名を mydb_dbo から に戻す方法を示していますdbo。

```
{
    "rules": [
        {
            "rule-type": "transformation",
            "rule-id": "566251737",
            "rule-name": "566251737",
            "rule-target": "schema",
            "object-locator": {
                "schema-name": "mydb_dbo"
            },
            "rule-action": "rename",
            "value": "dbo",
            "old-value": null
        },
        {
            "rule-type": "selection",
            "rule-id": "566111704",
            "rule-name": "566111704",
            "object-locator": {
                "schema-name": "mydb_dbo",
                "table-name": "%"
            },
            "rule-action": "include",
            "filters": []
        }
    ]
}
```

Babelfish テーブルで PostgreSQL ソースエンドポイントを使用するための制限事項

Babelfish テーブルで PostgreSQL ソースエンドポイントを使用する場合、次の制限が適用されます。

- DMS は、Babelfish バージョン 16.2/15.6 以降、および DMS バージョン 3.5.3 以降の移行のみをサポートします。
- DMS は、Babelfish テーブル定義の変更をターゲットエンドポイントにレプリケートしません。この制限の回避策は、まずターゲットにテーブル定義の変更を適用し、次に Babelfish ソースのテーブル定義を変更することです。

- BYTEA データ型を使用して Babelfish テーブルを作成すると、DMS は SQL Server をターゲットとして移行するときに、テーブルを`varbinary(max)`データ型に変換します。
- DMS は、バイナリデータ型のフル LOB モードをサポートしていません。バイナリデータ型には制限付き LOB モードを使用してください。
- DMS は、ソースとしての Babelfish のデータ検証をサポートしていません。
- ターゲットテーブル準備モードのタスク設定では、何もしないモードまたは切り捨てモードのみを使用します。[ターゲット上のテーブルを削除] モードは使用しないでください。ターゲット でドロップテーブルを使用する場合、DMS は誤ったデータ型でテーブルを作成することがあります。
- 継続的なレプリケーション (CDC または全ロードと CDC) を使用する場合は、`PluginName`追加の接続属性 (ECA) を に設定します`TEST_DECODING`。

PostgreSQL ソースデータベースからの AWS DMS アーティファクトの削除

DDL イベントをキャプチャするために、 は移行タスクの開始時に PostgreSQL データベースにさまざまなアーティファクト AWS DMS を作成します。タスクが完了したら、これらのアーティファクトを削除できます。

アーティファクトを削除するには、以下のステートメントを発行します (示されている順序で)。ここに、`{AmazonRDSMigration}` は、アーティファクトが作成されたスキーマです。スキーマを削除する場合でも、細心の注意を払ってください。運用中のスキーマ (特に公開スキーマ) は絶対に削除しないでください。

```
drop event trigger awsdms_intercept_ddl;
```

イベントトリガーは特定のスキーマに属していません。

```
drop function {AmazonRDSMigration}.awsdms_intercept_ddl()
drop table {AmazonRDSMigration}.awsdms_ddl_audit
drop schema {AmazonRDSMigration}
```

DMS ソースとして PostgreSQL データベースを使用する場合の追加設定

PostgreSQL データベースからデータを移行するときは、2 つの方法で詳細設定を追加できます。

- 追加接続属性に値を追加して、DDL イベントをキャプチャし、運用中の DDL データベースアーティファクトが作成されたスキーマを指定します。詳細については、「PostgreSQL を DMS ソースとして使用する場合のエンドポイント設定と追加の接続属性 (ECAs)」を参照してください。

- 接続文字列パラメータを上書きできます。これを行うには、次のいずれかのオプションを選択します：
  - 内部 AWS DMS パラメータを指定します。このようなパラメータが必要になることはめったにないため、ユーザー インターフェイスには表示されません。
  - 特定のデータベースクライアントのパススルー (パススルー) 値を指定します。データベースクライアントに渡される接続スティングにパススルーパラメータ AWS DMS を含めます。
- PostgreSQL バージョン 9.4 以降でテーブルレベルのパラメータ `REPLICA IDENTITY` を使用すると、ログ先行書き込み (WAL) に書き込まれる情報を制御できる。特に、更新または削除される行を識別する WAL に対してそうします。`REPLICA IDENTITY FULL` は行のすべての列の古い値を記録します。不必要に余分な量の WAL が FULL により生成されるため、`REPLICA IDENTITY FULL` はテーブルごとに慎重に使用してください。詳細については、「ALTER TABLE-REPLICA IDENTITY」を参照。

MapBooleanAsBoolean PostgreSQL エンドポイント設定の使用

PostgreSQL エンドポイント設定を使用して、ブール値を PostgreSQL ソースから Amazon Redshift ターゲットにブール値としてマッピングできます。ブール型はデフォルトで varchar (5) として移行されます。次の例のとおり、`MapBooleanAsBoolean` を指定すると、PostgreSQL がブール型をブール型として移行できるようになります。

```
--postgre-sql-settings '{"MapBooleanAsBoolean": true}'
```

この設定を有効にするには、ソースエンドポイントとターゲットエンドポイントの両方でこの設定を設定する必要があることに注意します。

MySQL には BOOLEAN 型がないため、BOOLEAN データを MySQL に移行する場合は、この設定ではなく変換ルールを使用します。

PostgreSQL を DMS ソースとして使用する場合のエンドポイント設定と追加の接続属性 (ECAs)

エンドポイント設定と追加の接続属性 (ECAs) を使用して、PostgreSQL ソースデータベースを設定できます。ソースエンドポイントの作成時に AWS DMS 、コンソールを使用するか、`--postgre-sql-settings '{"EndpointSetting": "value", ...}'` JSON 構文で の `create-endpoint` コマンドを使用してAWS CLIエンドポイント設定を指定します。

次の表は、PostgreSQL をソースとして使用できるエンドポイント設定と ECAs を示しています。

属性名	説明
CaptureDDLs	DDL イベントをキャプチャするために、 はタスクの開始時に PostgreSQL データベースにさまざまなアーティファクト AWS DMS を作成します。PostgreSQL ソースデータベースからの AWS DMS アーティファクトの削除に記載されているように、これらのアーティファクトは後で削除できます。 この値が false に設定されている場合、ソースデータベースにテーブルやトリガーを作成する必要はない。 ストリーミングされた DDL イベントがキャプチャされます。 デフォルト値: true 有効な値: true/false 例: --postgre-sql-settings '{"CaptureDDLs": true}'
ConsumeMonotonicEvents	重複するログシーケンス番号 (LSN) を持つモノリシック・トランザクションのレプリケーション方法を制御するために使用します。このパラメータが false の場合、重複した LSN を持つイベントが消費され、ターゲット上でレプリケーションされます。このパラメータが true の場合、最初のイベントのみレプリケーションされ、重複する LSN を持つイベントはターゲット上で消費もレプリケーションもされません。 デフォルト値: false 有効な値: false/true 例: --postgre-sql-settings '{"ConsumeMonotonicEvents": true}'

属性名	説明
DdlArtifactsSchema	運用中の DDL データベース アーティファクトが作成されるスキーマを設定します。 デフォルト値: public 有効な値: 文字列 例: `--postgre-sql-settings '{"DdlArtifactsSchema": "  xyzddlschema "}'`
ExecuteTimeout	PostgreSQL インスタンスのクライアントステートメントタイムアウト (秒単位) を設定します。デフォルト値は 60 秒です。 例: `--postgre-sql-settings '{"ExecuteTimeout": 100}'`
FailTasksOnLobTruncation	true に設定されている場合、LOB 列の実際のサイズが、指定された LobMaxSize を上回ると、この値によりタスクは失敗します。 タスクが制限付き LOB モードに設定され、このオプションが true に設定されている場合、LOB データを切り捨てるのではなくタスクが失敗します。 デフォルト値: false 有効な値: ブール値 例: `--postgre-sql-settings '{"FailTasksOnLobTruncation": true}'`

属性名	説明
fetchCacheSize	この追加の接続属性 (ECA) は、フルロード実行中にカーソルがフェッチする行数を設定する。レプリケーションインスタンスで利用可能なリソースに応じて、この値を増減して調整できる。 デフォルト値: 10000 有効な値: 数値 ECA の例:fetchCacheSize=10000;
HeartbeatFrequency	WAL ハートビートの頻度を設定します (分)。 デフォルト値: 5 有効な値: 数値 例: --postgre-sql-settings '{"HeartbeatFrequency": 1}'
HeartbeatSchema	ハートビート アーティファクトが作成されるスキーマを設定します。 デフォルト値: public 有効な値: 文字列 例: --postgre-sql-settings '{"HeartbeatSchema": "xyzheartbeatschema"}'
MapJsonbAsClob	デフォルトでは、 は JSONB を NCLOB に AWS DMS マッピングします。MapJsonbAsClob を指定すると、PostgreSQL で JSONB 型を CLOB として移行できるようになる。 例: --postgre-sql-settings='{"MapJsonbAsClob": "true"}'

属性名	説明
MapLongVarcharAs	デフォルトでは、 は VARCHAR を WSTRING に AWS DMS マッピングします。MapLongVarcharAs を指定すると、PostgreSQL が VARCHAR (N) 型 (N は 16,387 より大きい) を次のタイプに移行するように指定できる。 • WSTRING • CLOB • NCLOB 例: --postgre-sql-settings='{"MapLongVarcharAs": "CLOB"}'

属性名	説明
MapUnboundedNumericAsString	このパラメータは、数値の精度を失うことなく正常に移行するために、境界なしの NUMERIC データ型の列を文字列として扱います。このパラメーターは、PostgreSQL ソースから PostgreSQL ターゲットへのレプリケーション、または PostgreSQL 互換のデータベースにのみ使用します。 デフォルト値: false 有効な値: false/true 例: --postgre-sql-settings '{"MapUnboundedNumericAsString": true}' このパラメータを使用すると、数値から文字列への変換、および数値への変換により、レプリケーションのパフォーマンスが低下する可能性があります。このパラメータは、DMS バージョン 3.4.4 以降で使用するためにサポートされています **Note** PostgreSQL のソースとターゲット エンドポイントを一緒にのみ MapUnboundedNumericAsString を使用します。 ソース PostgreSQL エンドポイントで MapUnboundedNumericAsString を使用すると、CDC 中に精度が 28 に制限されます。ターゲット エンドポイントで MapUnboundedNumericAsString を使用すると、精度 28 スケール 6 でデータを移行します。 PostgreSQL 以外のターゲットとは MapUnboundedNumericAsString を使用しません。

属性名	説明
PluginName	レプリケーション スロットの作成に使用するプラグインを指定します。 有効な値: pglogical 、test_decoding 例: --postgre-sql-settings '{"PluginName": "test_decoding"}'
SlotName	PostgreSQL ソースインスタンスの CDC ロード用に以前に作成された論理レプリケーションスロットの名前を設定します。 AWS DMS API CdcStartPosition リクエストパラメータとともに使用すると、この属性はネイティブ CDC 開始ポイントの使用も有効にします。DMS は、CDC ロードタスクを開始する前に、指定された論理レプリケーションスロットが存在することを確認します。また、タスクが有効な CdcStartPosition 設定を使用して作成されたことも確認します。指定されたスロットが存在しないか、タスクに有効な CdcStartPosition 設定がない場合、DMS によりエラーが発生します。 CdcStartPosition リクエストパラメータの設定の詳細については、「CDC ネイティブ開始点の決定」をご参照ください。CdcStartPosition の使用方法の詳細については、AWS Database Migration Service API リファレンスのCreateReplicationTask 、StartReplicationTask 、ModifyReplicationTask でのAPI オペレーションについてのドキュメントをご参照ください。 有効な値: 文字列 例: --postgre-sql-settings '{"SlotName": "abc1d2efghijk_34567890_z0yx98w7_6v54_32ut_1srq_1a2b34c5d67ef"}'

属性名	説明
unboundedVarcharMaxSize	この追加接続属性 (ECA) は、最大長指定子 VarChar のないタイプとして定義されたデータ列の最大サイズを定義します。デフォルトは 8000 バイトです。最大値は 10485760 バイトです。

DMS のソースとして PostgreSQL データベースを使用する場合の制限

PostgreSQL を AWS DMSのソースとして使用する場合は、以下の制限が適用されます。

- AWS DMS は、Amazon RDS for PostgreSQL 10.4 または Amazon Aurora PostgreSQL 10.4 をソースまたはターゲットとして使用することはできません。
- キャプチャされたテーブルにはプライマリキーが必要です。テーブルにプライマリキーがない場合、 はそのテーブルの DELETE および UPDATE レコードオペレーション AWS DMS を無視します。回避策については、「Enabling change data capture (CDC) using logical replication」を参照してください。

  注: プライマリキーまたは一意のインデックスなしでの移行はお勧めしません。この場合、「いいえ」Batch 適用機能、フル LOB 機能、データ検証、Redshift ターゲットへの効率的なレプリケートができないなどの追加の制限が適用されます。
- AWS DMS は、プライマリキーセグメントを更新しようとする試みを無視します。この場合、ターゲットは、その更新によってどの行も更新されないと識別します。ただし、PostgreSQL でのプライマリキーの更新結果は予測できないため、どのレコードも例外テーブルには書き込まれません。
- AWS DMS は、タイムスタンプからプロセス変更を開始する実行オプションをサポートしていません。
- AWS DMS は、パーティションまたはサブパーティションオペレーション (ADD、、または) に起因する変更をレプリケートしませんDROPTRUNCATE。
- 大文字と小文字の組み合わせが異なる同じ名前 (table1、TABLE1、Table1) を持つ複数のテーブルをレプリケートすると、予測できない動作が生じることがあります。この問題のため、 AWS DMS はこの種のレプリケーションをサポートしていません。
- ほとんどの場合、 は tables. AWS DMS does の CREATE、ALTER、DROP DDL ステートメントの変更処理 AWS DMS をサポートします。テーブルが内部関数またはプロシージャ本文ブロッ

ク、または他のネストされたコンストラクトに保持されている場合、この変更処理はサポートされません。

たとえば、以下の変更はキャプチャされません。

```
CREATE OR REPLACE FUNCTION attu.create_distributors1() RETURNS void
LANGUAGE plpgsql
AS $$
BEGIN
create table attu.distributors1(did serial PRIMARY KEY,name
varchar(40) NOT NULL);
END;
$$;
```

- 現在のところ、PostgreSQL ソースの boolean データ型は、一貫性のない値を持つ bit データ型として SQLServer ターゲットに移行されます。回避策として、列VARCHAR(1)のデータ型を使用してテーブルを事前に作成します (または AWS DMS でテーブルを作成します)。次に、ダウンストリーム処理で「F」を False、「T」を True として扱います。
- AWS DMS は TRUNCATE オペレーションの変更処理をサポートしていません。
- OID LOB データ型は、ターゲットに移行されません。
- AWS DMS は、同種移行のみの PostGIS データ型をサポートします。
- 使用するソースがオンプレミスあるいは Amazon EC2 インスタンス上の PostgreSQL データベースの場合、test_decoding 出力プラグインがソースのエンドポイントにインストールされていることを確認してください。このプラグインは、PostgreSQL contrib パッケージで提供されています。test-decoding プラグインの詳細については、「PostgreSQL のドキュメント」をご参照ください。
- AWS DMS では、列のデフォルト値を設定および設定解除する変更処理はサポートされていません (ALTER TABLE ステートメントの ALTER COLUMN SET DEFAULT 句を使用)。
- AWS DMS は、列の nullability を設定する変更処理をサポートしていません (ALTER TABLE ステートメントで ALTER COLUMN [SET|DROP] NOT NULL 句を使用)。
- 論理レプリケーションが有効な場合、トランザクションあたりのメモリに保持される変更の最大数は 4 MB です。その後、変更がディスクにこぼれます。その結果、ReplicationSlotDiskUsageが増加し、トランザクションが完了または停止し、ロールバックが完了するまで restart_lsn は進みません。長いトランザクションであるため、ロールバックに時間がかかることがあります。そのため、論理レプリケーションが有効になっている場合は、長時間実行されるトランザクションや多数のサブトランザクションは避けてください。代わりに、トランザクションをいくつかの小さなトランザクションに分割します。

Aurora PostgreSQL バージョン 13 以降では、DMS が変更データをディスクにスピルするタイミングを制御するために `logical_decoding_work_mem`パラメータを調整できます。詳細については、「Aurora PostgreSQL のスピルファイル」を参照してください。

- ARRAY データ型を持つテーブルにはプライマリ キーが必要です。ARRAY データ型にプライマリキーがないテーブルは、全ロード中に中断されます。
- AWS DMS は、パーティションテーブルのレプリケーションをサポートしていません。パーティション分割されたテーブルが検出された場合、以下の状況が発生します。
  - エンドポイントは、親テーブルと子テーブルのリストを報告します。
  - AWS DMS は、選択したテーブルと同じプロパティを持つ通常のテーブルとしてターゲット上にテーブルを作成します。
  - ソースデータベースの親テーブルに子テーブルと同じプライマリキー値がある場合、「重複するキー」エラーが生成されます。
- パーティション分割されたテーブルを PostgreSQL ソースから PostgreSQL ターゲットにレプリケーションする場合、まずターゲットで親テーブルと子テーブルを手動で作成する必要があります。次に、それらのテーブルにレプリケーションする別個のタスクを定義します。その場合、タスク設定を [Truncate before loading] (読み取り前切り捨て) に設定します。
- PostgreSQL `NUMERIC` データ型はサイズが固定されていません。`NUMERIC` データ型ですが、精度とスケールがないデータを転送する場合、DMS はデフォルトで `NUMERIC(28,6)` (精度が 28、スケールが 6) を使用します。たとえば、ソースからの値 0.611111104488373 は、PostgreSQL ターゲットの 0.611111 に変換されます。
- AWS DMS は、Aurora PostgreSQL Serverless V1 をフルロードタスクのソースとしてのみサポートします。 は、Aurora PostgreSQL Serverless V2 をフルロード、フルロード、CDC、CDC のみのタスクのソースとして AWS DMS サポートします。
- AWS DMS は、coalesce 関数で作成された一意のインデックスを持つテーブルのレプリケーションをサポートしていません。
- LOB モードを使用する場合、ソーステーブルと対応するターゲットテーブルの両方に同一のプライマリキーが必要です。いずれかのテーブルにプライマリキーがない場合、DELETE および UPDATE レコード操作は予測できない結果となります。
- 並列ロード機能を使用する場合、パーティションまたはサブパーティションに基づくテーブルのセグメント化はサポートされません。並列ロードの詳細については、「選択したテーブルおよびビューさらにコレクションで並列ロードを使用する」を参照してください。
- AWS DMS は遅延制約をサポートしていません。

- AWS DMS バージョン 3.4.7 ではPostgreSQL 14.x をソースとしてサポートしていますが、次の制限があります。
  - AWS DMS は、2 フェーズコミットの変更処理をサポートしていません。
  - AWS DMS は、進行中の長いトランザクションをストリーミングするための論理レプリケーションをサポートしていません。
- AWS DMS は、ソースとして Amazon RDS Proxy for PostgreSQL の CDC をサポートしていません。
- プライマリキー列を含まない ソースフィルター を使用する場合、DELETE 操作はキャプチャされません。
- ソースデータベースが別のサードパーティーのレプリケーションシステムのターゲットでもある場合、CDC 中に DDL の変更が移行されない可能性があります。これは、このような状況では awsdms_intercept_ddl イベントトリガーが起動しなくなる可能性があるためです。この状況を回避するには、ソースデータベースでこのトリガーを次のとおり変更します。

```
alter event trigger awsdms_intercept_ddl enable always;
```

- AWS DMS RDS for PostgreSQL マルチ AZ データベースクラスターは論理レプリケーションをサポートしていないため、 はソースとして PostgreSQL 用の Amazon RDS マルチ AZ データベースクラスターの CDC をサポートしていません。

PostgreSQL のソースデータ型

次の表は、 の使用時にサポートされる PostgreSQL ソースデータ型 AWS DMS と、 AWS DMS データ型へのデフォルトのマッピングを示しています。

ターゲットにマッピングされるデータ型を表示する方法については、使用しているターゲットエンドポイントのセクションをご参照ください。

AWS DMS データ型の詳細については、「」を参照してくださいAWS Database Migration Service のデータ型。

PostgreSQL のデータ型	DMS データ型
INTEGER	INT4
SMALLINT	INT2

PostgreSQL のデータ型	DMS データ型
BIGINT	INT8
NUMERIC (p,s)	精度が 0 ~ 38 の場合、NUMERIC を使用します。 精度が 39 以上の場合、STRING を使用します。
DECIMAL(P,S)	精度が 0 ~ 38 の場合、NUMERIC を使用します。 精度が 39 以上の場合、STRING を使用します。
REAL	REAL4
DOUBLE	REAL8
SMALLSERIAL	INT2
SERIAL	INT4
BIGSERIAL	INT8
MONEY	NUMERIC(38,4) MONEY データ型は、SQL Server の FLOAT にマッピングされます。
CHAR	WSTRING (1)
CHAR(N)	WSTRING (n)
VARCHAR(N)	WSTRING (n)
TEXT	NCLOB
CITEXT	NCLOB

PostgreSQL のデータ型	DMS データ型
BYTEA	BLOB
TIMESTAMP	DATETIME
タイムスタンプ時間帯あり	DATETIME
DATE	DATE
TIME	TIME
TIME WITH TIME ZONE	TIME
INTERVAL	STRING (128)—1 YEAR、2 MONTHS、3 DAYS、4 HOURS、5 MINUTES、6 SECONDS
BOOLEAN	CHAR (5) false または true
ENUM	STRING (64)
CIDR	STRING (50)
INET	STRING (50)
MACADDR	STRING (18)
BIT (n)	STRING (n)
BIT VARYING (n)	STRING (n)
UUID	STRING
TSVECTOR	CLOB
TSQUERY	CLOB
XML	CLOB
POINT	STRING (255) "(x,y)"

PostgreSQL のデータ型	DMS データ型
LINE	STRING (255) "(x,y,z)"
LSEG	STRING (255) "((x1,y1),(x2,y2))"
BOX	STRING (255) "((x1,y1),(x2,y2))"
PATH	CLOB "((x1,y1),(xn,yn))"
POLYGON	CLOB "((x1,y1),(xn,yn))"
CIRCLE	STRING (255) "(x,y),r"
JSON	NCLOB
JSONB	NCLOB
配列	NCLOB
COMPOSITE	NCLOB
HSTORE	NCLOB
INT4RANGE	STRING (255)
INT8RANGE	STRING (255)
NUMRANGE	STRING (255)
STRRANGE	STRING (255)

PostgreSQL 用の LOB ソースデータ型での作業

PostgreSQL の列サイズは、PostgreSQL LOB データ型の AWS DMS データ型への変換に影響します。これを使用するには、次の AWS DMS データ型で以下の手順を実行します。

- BLOB - タスク作成で [Limit LOB size to] (LOB サイズ制限) 値を [Maximum LOB Size (KB)] (最大 LOB サイズ (KB)) に設定します。
- CLOB - レプリケーションは各文字を UTF8 文字として処理します。次に、列で最長の文字テキストの長さ (ここでは、`max_num_chars_text`) を見つけます。この長さを使用して、[Limit LOB

size to] (LOB サイズを以下に制限する) の値を指定します。データに 4 バイト文字が含まれている場合には、2 倍にして [Limit LOB size to (LOB サイズ制限)] 値をバイト単位で指定します。この場合、[Limit LOB size to] (LOB サイズ制限) は `max_num_chars_text` の 2 乗と等しくなります。

- NCLOB - レプリケーションは各文字を 2 バイト文字として処理します。次に、列で最長の文字テキストの長さ (`max_num_chars_text`) を見つけ、2 倍します。これは、[Limit LOB size to] (LOB サイズを以下に制限する) の値の値を指定するために行います。この場合、[Limit LOB size to] (LOB サイズ制限) は `max_num_chars_text` の 2 乗と等しくなります。データに 4 バイト文字が含まれている場合は、再度 2 倍にします。この場合、[Limit LOB size to (LOB サイズ制限)] は `max_num_chars_text` の 4 乗と等しくなります。

MySQL 互換データベースの AWS DMSのソースとしての使用

Database Migration Service を使用して、MySQL と互換性のある任意のデータベース (MySQL、MariaDB、または Amazon Aurora MySQL) からデータを移行できます。 AWS

AWS DMS がソースとしてサポートする MySQL のバージョンについては、「のソース AWS DMS」を参照してください。

SSL を使用して、MySQL 互換のエンドポイントとレプリケーションインスタンスとの接続を暗号化できます。MySQL 互換のエンドポイントで SSL を使用する方法の詳細については、「での SSL の使用 AWS Database Migration Service」をご参照ください。

以下のセクションでは、「セルフ管理」という用語は、オンプレミスまたは Amazon EC2 にインストールされているデータベースが対象です。「AWSが管理する」という用語は、Amazon RDS、Amazon Aurora、Amazon S3 のデータベースが対象です。

MySQL 互換データベースおよびの操作の詳細については AWS DMS、以下のセクションを参照してください。

トピック

- AWS DMSを使用して MySQL から MySQL へ移行します。
- MySQL 互換のデータベースをソースとして使用する AWS DMS
- 自己管理型のMySQL互換データベースをソースとして使用する AWS DMS
- AWSマネージド型の MySQL 互換データベースをソースとして使用する AWS DMS
- MySQL データベースをソースとして使用する場合の制限事項 AWS DMS
- XA トランザクションのサポート
- MySQL をソースとして使用する場合のエンドポイント設定 AWS DMS

- MySQL のソースデータ型

AWS DMSを使用して MySQL から MySQL へ移行します。

MySQL 以外のデータベースエンジンから MySQL データベースに移行する異種移行では、ほとんどの場合、 AWS DMS 最適な移行ツールです。ただし、MySQL データベースから MySQL データベースに移行する同種移行の場合は、同種データ移行プロジェクトを使用することをお勧めします。同種データ移行では、ネイティブのデータベースツールを使用して、 AWS DMSと比較してデータ移行のパフォーマンスと精度が向上します。

MySQL 互換のデータベースをソースとして使用する AWS DMS

MySQL データベースをソースとして使用する前に AWS DMS、次の前提条件を満たしていることを確認してください。これらの前提条件は、自己管理型ソースと管理型ソースのどちらにも適用されます。 AWS

Replication Admin ロールを持つアカウントが必要です。 AWS DMS ロールには、次の権限が必要です。

- [REPLICATION CLIENT] (レプリケーション クライアント) – この権限は、CDC タスクにのみ必要です。つまり、 full-load-only タスクにはこの権限は必要ありません。
- [REPLICATION SLAVE] (レプリケーション スレーブ) – この権限は、CDC タスクにのみ必要です。つまり、 full-load-only タスクにはこの権限は必要ありません。
- SUPER – この権限は、バージョン 5.6.6 より前の MySQL でのみ必要です。

AWS DMS ユーザーには、レプリケーションの対象となるソーステーブルに対する SELECT 権限も必要です。

自己管理型のMySQL互換データベースをソースとして使用する AWS DMS

次のセルフマネージド型 MySQL 互換データベースを AWS DMSのソースとして使用できます。

- MySQL Community Edition
- MySQL Standard Edition
- MySQL Enterprise Edition
- MySQL Cluster Carrier Grade Edition
- MariaDB Community Edition

- MariaDB Enterprise Edition
- MariaDB Column Store

CDC を使用するには、バイナリログを有効にしてください。バイナリロギングを有効にするには、MySQL の `my.ini` (Windows) または `my.cnf` (UNIX) ファイルで以下のパラメータを設定する必要があります。

パラメータ	値
`server_id`	このパラメータは、1 以上の値に設定します。
`log-bin`	パスをバイナリログファイル (`log-bin=E:\MySql_Logs\BinLog`) に設定します。ファイル拡張子を含めないでください。
`binlog_format`	このパラメータは `ROW` に設定します。`binlog_format` が `STATEMENT` に設定されているときは場合によっては、ターゲットにデータレプリケーション時に矛盾が生じる可能性があるため、レプリケーション中にこの設定をお勧めします。データベースエンジンは、`binlog_format` が `MIXED` に設定されているとき、類似の矛盾したデータもターゲットに書き込みます。これはデータベースエンジンが自動的に`STATEMENT` ベースのログに切り替わり、ターゲットデータベースに一貫性のないデータが書き込まれることがあるためです。
`expire_logs_days`	このパラメータは、1 以上の値に設定します。ディスク容量の使いすぎを防ぐため、デフォルト値の 0 は使用しないことをお勧めします。
`binlog_checksum`	DMS バージョン 3.4.7 `NONE` 以前の場合は、このパラメータをに設定します。
`binlog_row_image`	このパラメータは `FULL` に設定します。
`log_slave_updates`	MySQL または MariaDB リードレプリカをソースとして使用している場合は、このパラメータを `TRUE` に設定します。

ソースで NDB (クラスター化) データベースエンジンを使用している場合、そのストレージエンジンを使用するテーブルで CDC を有効にするには以下のパラメータを設定する必要があります。これらの変更を MySQL の `my.ini` (Windows) または `my.cnf` (UNIX) ファイルに追加します。

パラメータ	値
`ndb_log_bin`	このパラメータは `ON` に設定します。この値により、クラスター化されたテーブルでの変更が確実にバイナリログに記録されます。
`ndb_log_update_as_write`	このパラメータは `OFF` に設定します。この値に設定すると、UPDATE ステートメントが INSERT ステートメントとしてバイナリログに書き込まれなくなります。
`ndb_log_updated_only`	このパラメータは `OFF` に設定します。この値に設定すると、バイナリログに変更された列だけでなく行全体が含められます。

AWSマネージド型の MySQL 互換データベースをソースとして使用する AWS DMS

AWS以下のマネージド MySQL 互換データベースをソースとして使用できます。 AWS DMS

- MySQL Community Edition
- MariaDB Community Edition
- Amazon Aurora MySQL 互換エディション

AWSマネージド MySQL 互換データベースをのソースとして使用する場合は AWS DMS、CDC の以下の前提条件を満たしていることを確認してください。

- RDS for MySQL と RDS for MariaDB のバイナリログを有効にするには、インスタンスレベルで自動バックアップを有効にします。Aurora MySQL クラスターのバイナリログを有効にするには、パラメータグループで変数 `binlog_format` を変更します。

  自動バックアップの設定の詳細については、Amazon RDS ユーザーガイドの「自動バックアップの使用」をご参照ください。

  Amazon RDS for MySQL データベースのバイナリログ設定の詳細については、Amazon RDS ユーザーガイドの「バイナリログ形式の設定」をご参照ください。

Aurora MySQL クラスターのバイナリログ設定の詳細については、「Amazon Aurora MySQL クラスターのバイナリログを有効にする方法」をご参照ください。

- CDC を使用する予定の場合は、バイナリログを有効にします。Amazon RDS for MySQL データベースのバイナリログの設定の詳細については、Amazon RDS ユーザーガイドの「バイナリログ形式の設定」をご参照ください。
- バイナリログがで使用できることを確認してください。 AWS DMS AWS管理された MySQL 互換データベースはバイナリログをできるだけ早く消去するため、ログが使用可能な期間を長くする必要があります。たとえば、ログ保持を 24 時間に伸ばすには、次のコマンドを実行します。

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

- `binlog_format` パラメータを `"ROW"` に設定します。

> **Note**
>
> MySQL または MariaDB では、`binlog_format` は動的パラメータであるため、新しい値を有効にするために再起動する必要はありません。ただし、新しい値は新しいセッションにのみ適用されます。レプリケーションの目的で `binlog_format` を `ROW` に切り替えても、値を変更する前にセッションが開始されていれば、データベースは引き続き `MIXED` 形式を使用して続くバイナリログを作成できます。これにより、 AWS DMS ソースデータベースのすべての変更を正しくキャプチャできなくなる可能性があります。MariaDB または MySQL データベースの `binlog_format` 設定を変更する場合は、必ずデータベースを再起動して既存のすべてのセッションを閉じるか、DML (データ操作言語) 操作を実行するアプリケーションを再起動します。`binlog_format`ROWパラメータをに変更した後でデータベースのすべてのセッションを強制的に再起動させると、それ以降のすべてのソースデータベースの変更が正しい形式で書き込まれるため、 AWS DMS それらの変更を正しくキャプチャできます。

- `binlog_row_image` パラメータを `"Full"` に設定します。
- DMS バージョン `"NONE"` 3.4.7 以前の場合は、`binlog_checksum`パラメータをに設定します。Amazon RDS MySQL でのパラメータの設定の詳細については、Amazon RDS ユーザーガイドの「自動バックアップの使用」をご参照ください。

- Amazon RDS MySQL または Amazon RDS MariaDB リードレプリカをソースとして使用する場合は、リードレプリカでバックアップを有効にして、`log_slave_updates` パラメータが `TRUE` と設定されていることを確認します。

MySQL データベースをソースとして使用する場合の制限事項 AWS DMS

MySQL データベースをソースとして使用する場合は、次の点を考慮してください。

- 変更データキャプチャ (CDC) は、Amazon RDS MySQL 5.5 以下ではサポートされていません。Amazon RDS for MySQL の場合、CDC を有効にするには、バージョン 5.6、5.7、または 8.0 を使用する必要があります。CDC は、セルフ管理 MySQL 5.5 ソースでサポートされています。
- CDC の場合、列データ型を変更する `CREATE TABLE`、`ADD COLUMN`、`DROP COLUMN`、`renaming a column` がサポートされています。ただし、`DROP TABLE`、`RENAME TABLE`、およびカラムのデフォルト値、カラムのNULL可能性、文字セットなどの他の属性に対する更新はサポートされていません。
- ソース上のパーティションテーブルの場合、ターゲットテーブル準備モードを Drop tables on target に設定すると、MySQL AWS DMS ターゲットにパーティションのない単純なテーブルが作成されます。パーティション分割されたテーブルをターゲットのパーティション分割されたテーブルに移行するには、ターゲット MySQL データベースにパーティション分割されたテーブルを事前に作成します。
- `ALTER TABLE table_name ADD COLUMN column_name` ステートメントを使用して、テーブルの先頭 (FIRST) または中間 (AFTER) に列を追加することはできません。列は常にテーブルの末尾に追加されます。
- テーブル名に大文字と小文字が含まれていて、大文字と小文字が区別されるオペレーティングシステムにソースエンジンがホストされている場合、CDC はサポートされません。たとえば、Microsoft Windows や HFS+ を使用する OS X などです。
- Aurora MySQL 互換エディションサーバーレス v1 は全ロードに使用できますが、CDC には使用できません。これは MySQL の前提条件を有効にできないためです。詳細については、「パラメータグループと Aurora サーバーレス v1」をご参照ください。

  Aurora MySQL 互換エディションサーバーレス v2 は CDC をサポートしています。
- 列の AUTO_INCREMENT 属性は、ターゲットデータベース列に移行されません。
- バイナリログが標準のブロックストレージに保存されている場合の変更のキャプチャはサポートされていません。たとえば、バイナリログが Amazon S3 に保存されていると、CDC は機能しません。

- AWS DMS デフォルトで InnoDB ストレージエンジンを使用してターゲットテーブルを作成します。InnoDB 以外のストレージエンジンを使用する必要がある場合、テーブルを手動で作成し、何もしないモードを使用してそのテーブルに移行する必要があります。
- DMS 移行タスクモードが [既存データを移行-全ロードのみ] AWS DMS になっていない限り、Aurora MySQL レプリカをソースとして使用することはできません。
- 全ロード時に MySQL 互換ソースが停止している場合、 AWS DMS タスクはエラーで停止しません。タスクは正常に終了しますが、ターゲットとソースが同期しない可能性があります。この場合、タスクを再開するか、影響を受けたテーブルを再ロードしてください。
- 列の値の一部で作成されたインデックスは移行されません。たとえば、インデックス CREATE INDEX first_ten_chars ON customer (name(10)) はターゲットに作成されません。
- タスクが LOB を複製しないように設定されている場合もあります (タスク設定で SupportLobs「」が false になっているか、タスクコンソールで「LOB 列を含めない」が選択されている)。このような場合、は MEDIUMBLOB、LONGBLOB、MEDIUMTEXT、LONGTEXT AWS DMS の各列をターゲットに移行しません。

  BLOB、TINYBLOB、TEXT、および TINYTEXT 列は影響を受けず、ターゲットに移行されます。
- 一時データテーブルまたはシステムでバージョン管理されたテーブルは、MariaDB ソースおよびターゲットデータベースではサポートされていません。
- 2 つの Amazon RDS Aurora MySQL クラスター間で移行する場合、RDS Aurora MySQL ソース エンドポイントは、レプリカ インスタンスではなく読み取り/書き込みインスタンスである必要があります。
- AWS DMS 現在、MariaDBのビュー移行はサポートされていません。
- AWS DMS は MySQL のパーティションテーブルの DDL 変更をサポートしていません。CDC 中にパーティション DDL が変更されてもテーブルが中断されないようにするには、`skipTableSuspensionForPartitionDdl` を `true` に設定します。
- AWS DMS バージョン 3.5.0 以降の XA トランザクションのみをサポートします。以前のバージョンは XA トランザクションをサポートしていません。 AWS DMS は MariaDB バージョン 10.6 の XA トランザクションをサポートしていません。詳細については、「the section called “XA トランザクションのサポート”」を参照してください。
- AWS DMS ソースデータに GTID が含まれていても、レプリケーションには GTID を使用しません。
- AWS DMS バイナリログのトランザクション圧縮はサポートされていません。
- AWS DMS InnoDB ストレージエンジンを使用する MySQL データベースの ON DELETE CASCADE イベントと ON UPDATE CASCADE イベントは伝播されません。このようなイベント

については、MySQL は子テーブルでのカスケード操作を反映する binlog イベントを生成しません。そのため、 AWS DMS 対応する変更を子テーブルに複製することはできません。詳細については、「インデックス、外部キー、カスケード更新、または削除が移行されない」を参照してください。

- AWS DMS 計算列 (VIRTUALおよびGENERATED ALWAYS) への変更はキャプチャされません。この制限を回避するには、次の手順を実行します。
  - ターゲットデータベースにターゲットテーブルを事前に作成して、DO_NOTHINGまたは TRUNCATE_BEFORE_LOAD のフルロード設定の AWS DMS タスク設定を使用してタスクを作成する。
  - 計算列をタスクスコープから削除する変換ルールを追加する。変換ルールの詳細については、「変換ルールおよび変換アクション」を参照。

XA トランザクションのサポート

Extended Architecture (XA) トランザクションは、複数のトランザクションリソースによる操作セットを単一の信頼性の高いグローバルトランザクションにグループ化するために使用できるトランザクションです。XA トランザクションは 2 フェーズコミットプロトコルを使用します。通常、XA トランザクションがオープンである間に変更をキャプチャすると、データが損失する可能性があります。データベースが XA トランザクションを使用していない場合は、IgnoreOpenXaTransactionsCheck のデフォルト値 TRUE を使用してこのアクセス権限と設定を無視できます。XA トランザクションのあるソースからレプリケーションを開始するには、次を実行します。

- AWS DMS エンドポイントユーザーに以下の権限があることを確認してください。

```
grant XA_RECOVER_ADMIN on *.* to 'userName'@'%';
```

- エンドポイント設定 IgnoreOpenXaTransactionsCheck を false に設定する。

> **Note**
>
> AWS DMS は MariaDB ソース DB バージョン 10.6 の XA トランザクションをサポートしていません。

MySQL をソースとして使用する場合のエンドポイント設定 AWS DMS

追加の接続属性を使用する場合と同様、エンドポイント設定を使用してソースの MySQL データベースを設定できます。ソースエンドポイントを作成するときに、 AWS DMS コンソールを使用するか、`--my-sql-settings '{"EndpointSetting": "value", ...}'` JSON `create-endpoint` AWS CLI構文のコマンドを使用して設定を指定します。

次の表は、MySQL をソースとして使用できるエンドポイント設定を説明しています。

名前	説明
`EventsPollInterval`	データベースがアイドル状態のとき、バイナリログで新しい変更/イベントをチェックする頻度を指定します。 デフォルト値: 5 有効な値: 1 ~ 60 例: `--my-sql-settings '{"EventsPollInterval": 5}'` この例では、5 AWS DMS 秒ごとにバイナリログの変更をチェックします。
`ExecuteTimeout`	AWS DMS バージョン 3.4.7 以降では、MySQL ソースエンドポイントのクライアントステートメントのタイムアウトを秒単位で設定します。 デフォルト値: 60 例: `--my-sql-settings '{"ExecuteTimeout": 1500}'`
`ServerTimezone`	ソース MySQL データベースのタイムゾーンを指定します。 例: `--my-sql-settings '{"ServerTimezone": "US/Pacific "}'`
`AfterConnectScript`	AWS DMS エンドポイントに接続した直後に実行するスクリプトを指定します。移行タスクは、SQL ステートメ

名前	説明
	ントが成功するか失敗するかにかかわらず、引き続き実行されます。 有効な値: セミコロンで区切られた 1 つ以上の有効な SQL ステートメント。 例: `--my-sql-settings '{"AfterConnectScript": "ALTER SESSION SET CURRENT_SCHEMA=system"}'`
`CleanSrcMetadataOnMismatch`	不一致が発生すると、レプリケーションインスタンスのテーブルメタデータ情報をクリーンアップして再作成します。たとえば、テーブルの DDL を変更すると、レプリケーションインスタンスにキャッシュされているテーブルに関する情報が変更される場合があります。Boolean。 デフォルト値: `false` 例: `--my-sql-settings '{"CleanSrcMetadataOnMismatch": false}'`
`skipTableSuspensionForPartitionDdl`	AWS DMS は MySQL のパーティションテーブルの DDL 変更をサポートしていません。 AWS DMS バージョン 3.4.6 以降では、これを設定すると CDC 中のパーティション DDL `true` 変更によるテーブルの一時停止がスキップされます。 AWS DMS partitioned-table-related DDL を無視し、バイナリログの変更をさらに処理し続けます。 デフォルト値: `false` 例: `--my-sql-settings '{"skipTableSuspensionForPartitionDdl": true}'`

名前	説明
IgnoreOpenXaTransactionsCheck	AWS DMS バージョン 3.5.0 以降では、タスクが開始時に開いている XA トランザクションを無視するかどうかを指定します。ソースに XA トランザクションがある場合はこれを false に設定する。 デフォルト値: true 例: --my-sql-settings '{"IgnoreOpenXaTransactionsCheck": false}'

MySQL のソースデータ型

次の表は、使用時にサポートされる MySQL AWS DMS データベースのソースデータ型と、 AWS DMS データ型からのデフォルトマッピングを示しています。

ターゲットにマッピングされるデータ型を表示する方法については、使用しているターゲットエンドポイントのセクションをご参照ください。

AWS DMS データ型に関する追加情報については、を参照してくださいAWS Database Migration Service のデータ型。

MySQL のデータ型	AWS DMS データタイプ
INT	INT4
BIGINT	INT8
MEDIUMINT	INT4
TINYINT	INT1
SMALLINT	INT2
UNSIGNED TINYINT	UINT1
UNSIGNED SMALLINT	UINT2
UNSIGNED MEDIUMINT	UINT4

MySQL のデータ型	AWS DMS データタイプ
UNSIGNED INT	UINT4
UNSIGNED BIGINT	UINT8
DECIMAL(10)	NUMERIC (10,0)
BINARY	BYTES(1)
BIT	BOOLEAN
BIT(64)	BYTES(8)
BLOB	BYTES(65535)
LONGBLOB	BLOB
MEDIUMBLOB	BLOB
TINYBLOB	BYTES(255)
DATE	DATE
DATETIME	DATETIME 括弧で囲まれた値が指定されていない DATETIME は、ミリ秒なしでレプリケートされる。括弧内の値が 1~5 (`DATETIME(5)` など) の DATETIME は、ミリ秒単位でレプリケートされる。 DATETIME 列をレプリケートしても、ターゲット上の時刻は変わらない。UTC には変換されない。
TIME	STRING

MySQL のデータ型	AWS DMS データタイプ
TIMESTAMP	DATETIME TIMESTAMP 列をレプリケートすると、時間はターゲットで UTC に変換される。
YEAR	INT2
DOUBLE	REAL8
FLOAT	REAL(DOUBLE) FLOAT 値が次の範囲にない場合は、変換を使用して FLOAT を STRING にマップする。変換の詳細については、「変換ルールおよび変換アクション」をご参照ください。 サポートされる FLOAT の範囲は、-1.79E+308 ~ -2.23E-308、0、2.23E-308 ~ 1.79E+308。
VARCHAR(45)	WSTRING (45)
VARCHAR(2000)	WSTRING (2000)
VARCHAR(4000)	WSTRING (4000)
VARBINARY (4000)	BYTES (4000)
VARBINARY (2000)	BYTES (2000)
CHAR	WSTRING
TEXT	WSTRING
LONGTEXT	NCLOB
MEDIUMTEXT	NCLOB
TINYTEXT	WSTRING(255)

MySQL のデータ型	AWS DMS データタイプ
GEOMETRY	BLOB
POINT	BLOB
LINESTRING	BLOB
POLYGON	BLOB
MULTIPOINT	BLOB
MULTILINESTRING	BLOB
MULTIPOLYGON	BLOB
GEOMETRYCOLLECTION	BLOB
ENUM	WSTRING (*[length]* (長さ)) ここで、*[length]* (長さ) は列挙型の最長値の長さです。
SET	WSTRING (*[length]* (長さ)) ここで、*[length]* (長さ) は、カンマを含む SET 内のすべての値の合計の長さです。
JSON	CLOB

> **Note**
>
> 場合によっては、値が「ゼロ」(つまり、0000-00-00) の DATETIME データ型と TIMESTAMP データ型を指定できます。存在する場合は、レプリケーション タスクのターゲットデータベースが DATETIME データ型と TIMESTAMP データ型で「ゼロ」値に対応していることを確認してください。サポートされていない場合、これらの値はターゲットで NULL として記録されます。

AWS DMS のソースとしての SAP ASE データベースの使用

AWS DMS を使用して、SAP Adaptive Server Enterprise (ASE) データベース (旧名 Sybase) からデータを移行できます。SAP ASE データベースをソースとして使用する場合、AWS DMS がサポートする任意のターゲットにデータを移行できます。

AWS DMS がソースとしてサポートする SAP ASE のバージョンについては、「のソース AWS DMS」を参照してください。

SAP ASE ソースデータベースと AWS DMS の使用の詳細については、次のセクションを参照してください。

トピック

- SAP ASE データベースを AWS DMS のソースとして使用するための前提条件
- SAP ASE を AWS DMS のソースとして使用する場合の制限
- SAP ASE を AWS DMS のソースとして使用する場合に必要なアクセス権限
- 切り捨て点の削除
- SAP ASE を AWS DMS のソースとして使用する場合のエンドポイントの設定
- ソースの SAP ASE のデータ型

SAP ASE データベースを AWS DMS のソースとして使用するための前提条件

SAP ASE データベースを AWS DMS のソースとする場合、次を実行します。

- `sp_setreptable` コマンドを使用して、テーブルの SAP ASE レプリケーションを有効にします。詳細については、「Sybase Infocenter Archive」を参照してください。
- SAP ASE データベースの `RepAgent` を無効にします。詳細については、「Stop and disable the RepAgent thread in the primary database」を参照してください。
- 非ラテン文字 (中国語など) 向けに設定された、Windows EC2 インスタンス上の SAP ASE バージョン 15.7 にレプリケートするには、ターゲットコンピュータに SAP ASE 15.7 SP121 をインストールします。

> **Note**
>
> 継続的な変更データキャプチャ (CDC) レプリケーションの場合、DMS は `dbcc logtransfer` と `dbcc log` を実行して、トランザクションログからデータを読み取ります。

SAP ASE を AWS DMS のソースとして使用する場合の制限

SAP ASE データベースを AWS DMS のソースとして使用する場合、以下の制限が適用されます。

- SAP ASE データベースごとに、継続的なレプリケーションまたは CDC を使用して実行できる AWS DMS のタスクは 1 つのみです。フルロードのみのタスクは複数で並行して実行できます。
- テーブル名の変更はできません。例えば、以下のコマンドは失敗します。

```
sp_rename 'Sales.SalesRegion', 'SalesReg;
```

- 列名の変更はできません。例えば、以下のコマンドは失敗します。

```
sp_rename 'Sales.Sales.Region', 'RegID', 'COLUMN';
```

- バイナリデータ型文字列の末尾にあるゼロ値は、ターゲットデータベースにレプリケートされる際に切り捨てられます。例えば、ソーステーブル内の `0x00000000000000000000000001000001000000000` は、ターゲットテーブルでは `0x00000000000000000000000001000001` となります。
- データベースがデフォルトで NULL 値が許容しない場合、AWS DMS は NULL 値を許容しない列を持つターゲットテーブルを作成します。そのため、フルロードまたは CDC レプリケーションタスクに空の値が含まれる場合、AWS DMS はエラーを発生させます。このような エラーは、次のコマンドを使用してソースデータベースで NULL 値を許可すると回避できます。

```
sp_dboption database_name, 'allow nulls by default', 'true'
go
use database_name
CHECKPOINT
go
```

- `reorg rebuild` インデックスコマンドはサポートされていません。
- AWS DMS は、クラスターやソースとしての MSA (マルチサイト可用性) やウォームスタンバイの使用をサポートしていません。

- AR_H_TIMESTAMP 変換ヘッダー式がマッピング ルールで使用されている場合、追加された列のミリ秒はキャプチャされません。
- CDC 中に Merge 操作を実行すると、回復不可能なエラーが発生します。ターゲットを同期状態に戻すには、フルロードを実行します。
- ロールバックトリガーイベントは、データ行ロックスキームを使用するテーブルではサポートされません。
- AWS DMS は、ソース SAP データベースからタスクの範囲内のテーブルを削除した後はレプリケーションタスクを再開できません DMS レプリケーションタスクが停止し、DML 操作 (INSERT、UPDATE、DELETE) が実行された後にテーブルを削除した場合、レプリケーションタスクをもう一度開始する必要があります。

SAP ASE を AWS DMS のソースとして使用する場合に必要なアクセス権限

SAP ASE データベースを AWS DMS タスクのソースとして使用するには、アクセス権限を付与する必要があります。AWS DMS データベース定義に指定されたユーザーアカウントに、SAP ASE データベースで次のアクセス権限を付与します。

- sa_role
- replication_role
- sybase_ts_role
- デフォルトでは、sp_setreptable ストアドプロシージャを実行するための許可が必要な場合、AWS DMS は SAP ASE レプリケーションオプションを有効にします。AWS DMS 自体を介さず、データベースエンドポイントから直接テーブルに対してsp_setreptable を実行する場合は、enableReplication 追加接続属性を使用できます。詳細については、「SAP ASE を AWS DMS のソースとして使用する場合のエンドポイントの設定」を参照してください。

切り捨て点の削除

タスクが開始されると、AWS DMS は、syslogshold システムビューに $replication_truncation_point エントリを確立して、レプリケーションプロセスが進行中であることを示します。この間、既にターゲットにコピーされたデータの量に応じて、AWS DMS は定期的にレプリケーション切り捨て点を進めます。

$replication_truncation_point エントリが確立された後、データベースログの増大を避けるため、AWS DMS タスクは実行したままにしておく必要があります。AWS DMS タスクを永続的に停止する場合、次のコマンドを発行してレプリケーション切り捨て点を削除します。

```
dbcc settrunc('ltm','ignore')
```

切り捨て点が削除されたら、AWS DMS タスクを再開することはできません。ログは、引き続きチェックポイントで自動的に切り捨てられます (自動切り捨てが設定されている場合)。

SAP ASE を AWS DMS のソースとして使用する場合のエンドポイントの設定

追加の接続属性の使用と同様、エンドポイントの設定を使用して、ソースの SAP ASE データベースを設定できます。ソースエンドポイントを作成する際に、AWS DMS コンソールを使用するか、AWS CLI で `--sybase-settings '{"EndpointSetting": "value", ...}'` の JSON 構文を指定して `create-endpoint` コマンドを使用して設定を指定します。

次の表は、ソースとして SAP ASE を使用できるエンドポイント設定を説明しています。

名前	説明
`Charset`	この属性を国際文字セットに対応する SAP ASE 名に設定する。 デフォルト値: `iso_1` 例: `--sybase-settings '{"Charset": "utf8"}'` 有効値: • `acsii_8` • `big5hk` • `cp437` • `cp850` • `cp852` • `cp852` • `cp855` • `cp857` • `cp858` • `cp860` • `cp864`

名前	説明
	• cp866 • cp869 • cp874 • cp932 • cp936 • cp950 • cp1250 • cp1251 • cp1252 • cp1253 • cp1254 • cp1255 • cp1256 • cp1257 • cp1258 • deckanji • euccns • eucgb • eucjis • eucksc • gb18030 • greek8 • iso_1 • iso88592 • iso88595 • iso88596 • iso88597 • iso88598 • iso88599

名前	説明
	• iso15 • kz1048 • koi8 • roman8 • iso88599 • sjis • tis620 • turkish8 • utf8 SAP ASE データベースでサポートされる文字セットに関するその他の不明な点については、「Adaptive Server Enterprise: Supported character sets」を参照。
EnableReplication	AWS DMS 経由でなくデータベース側からテーブルで sp_setreptable を有効にする場合、この属性を設定する。 デフォルト値: true 有効値: true または false 例: --sybase-settings '{"EnableReplication": false}'

名前	説明
EncryptPassword	ソースデータベースで "net password encryption reqd" を有効にした場合は、この属性を設定する。 デフォルト値: 0 有効値: 0、1、または 2 例: --sybase-settings '{"EncryptPassword": 1}' これらのパラメータ値の詳細については、「Adaptive Server Enterprise: Using the EncryptPassword Connection string property」を参照。
Provider	ASE 15.7 以降のバージョンで Transport Layer Security (TLS) 1.1 または 1.2 を使用する場合は、この属性を設定する。AWS では TLS バージョン 1.2 以降が必要であり、推奨はバージョン 1.3 であることに注意する。 デフォルト値: Adaptive Server Enterprise 有効値: Adaptive Server Enterprise 16.03.06 例: --sybase-settings '{"Provider": "Adaptive Server Enterprise 16.03.06"}'

ソースの SAP ASE のデータ型

AWS DMS の使用時にサポートされるソースの SAP ASE のデータ型のリストと、AWS DMS データ型へのデフォルトのマッピングについては、次の表を参照してください。AWS DMS は、ユーザー定義型 (UDT) データ型の列を含む SAP ASE ソーステーブルをサポートしていません。このデータ型のレプリケート列は NULL として作成されます。

ターゲットにマップされるデータ型を確認する方法については、使用しているターゲットエンドポイントの「「データ移行のターゲット」」セクションを参照してください。

AWS DMS のデータ型の詳細については、「AWS Database Migration Service のデータ型」を参照してください。

SAP ASE のデータ型	AWS DMS のデータ型
BIGINT	INT8
UNSIGNED BIGINT	UINT8
INT	INT4
UNSIGNED INT	UINT4
SMALLINT	INT2
UNSIGNED SMALLINT	UINT2
TINYINT	UINT1
DECIMAL	NUMERIC
NUMERIC	NUMERIC
FLOAT	REAL8
DOUBLE	REAL8
REAL	REAL4
MONEY	NUMERIC
SMALLMONEY	NUMERIC
DATETIME	DATETIME
BIGDATETIME	DATETIME(6)
SMALLDATETIME	DATETIME
DATE	DATE
TIME	TIME
BIGTIME	TIME

SAP ASE のデータ型	AWS DMS のデータ型
CHAR	STRING
UNICHAR	WSTRING
NCHAR	WSTRING
VARCHAR	STRING
UNIVARCHAR	WSTRING
NVARCHAR	WSTRING
BINARY	BYTES
VARBINARY	BYTES
BIT	BOOLEAN
TEXT	CLOB
UNITEXT	NCLOB
IMAGE	BLOB

AWS DMS のソースとしての MongoDB の使用

AWS DMS がソースとしてサポートする MongoDB のバージョンについては、「のソース AWS DMS」を参照してください。

MongoDB バージョンのサポートについては、次の点に注意します。

- AWS DMS 3.4.5 以降のバージョンは、MongoDB バージョン 4.2 および 4.4 をサポートします。
- AWS DMS 3.4.5 以降のバージョンと MongoDB 4.2 以降のバージョンは、分散トランザクションをサポートします。MongoDB の分散トランザクションの詳細については、「MongoDB ドキュメント」の「Transactions」を参照してください。
- AWS DMS 3.5.0 以降のバージョンは、3.6 より前のバージョンの MongoDB をサポートしていません。

- AWS DMS 3.5.1 以降のバージョンは、MongoDB バージョン 5.0 をサポートします。
- AWS DMS 3.5.2 以降のバージョンは、MongoDB バージョン 6.0 をサポートしています。

MongoDB を初めてお使いの方のために、以下の MongoDB データベースの概念について説明します。

- MongoDB のレコードは、フィールドと値のペアで構成されるデータ構造であるドキュメントです。フィールドの値には、他のドキュメント、配列、およびドキュメントの配列を含めることができます。ドキュメントは、リレーショナルデータベースのテーブルの行とほぼ同等です。
- MongoDB のコレクションはドキュメントのグループであり、リレーショナルデータベーステーブルとほぼ同等です。
- MongoDB のデータベースはコレクションの集合であり、リレーショナル データベーステーブルのスキーマとほぼ同等です。
- 内部的には、MongoDB ドキュメントは、ドキュメント内の各フィールドのタイプを含む、圧縮形式でバイナリ JSON (BSON) ファイルとして格納されます。各ドキュメントには一意の ID があります。

AWS DMS は、MongoDB をソースとして使用する場合、ドキュメントモード または テーブルモードの 2 つの移行モードをサポートしています。MongoDB エンドポイントを作成するときか、AWS DMS コンソールからのメタデータモード パラメータを設定して、使用する移行モードを指定します。オプションとして、エンドポイント設定パネルで_id を個別の列として指定するのチェックマークボタンを選択して、プライマリ キーとして機能する _id という名前の 2 番目の列を作成できます。

移行モードの選択は、以下に説明するように、ターゲットデータの結果形式に影響します。

ドキュメントモード

ドキュメントモードでは、MongoDB ドキュメントは「現状のまま」移行されます。これは、その中のドキュメントデータが _doc という名前のターゲットテーブルの 1 つの列と見なされることを意味します。MongoDB をソースエンドポイントとして使用する場合のデフォルト設定はドキュメントモードです。

たとえば、myCollection という MongoDB コレクションの次のドキュメントを考えてみましょう。

```
> db.myCollection.find()
```

```
{ "_id" : ObjectId("5a94815f40bd44d1b02bdfe0"), "a" : 1, "b" : 2, "c" : 3 }
{ "_id" : ObjectId("5a94815f40bd44d1b02bdfe1"), "a" : 4, "b" : 5, "c" : 6 }
```

ドキュメントモードを使用してデータをリレーショナルデータベーステーブルに移行した後、データは以下のように構成されます。MongoDB ドキュメントのデータフィールドは _doc 列に統合されています。

oid_id	_doc
5a94815f40bd44d1b02bdfe0	{ "a" : 1, "b" : 2, "c" : 3 }
5a94815f40bd44d1b02bdfe1	{ "a" : 4, "b" : 5, "c" : 6 }

オプションで、追加の接続属性 extractDocID を true に設定して、プライマリキーとして動作する、"_id" という名前の 2 つ目の列を作成できます。CDC を使用する場合は、このパラメータを true に設定します。

ドキュメントモードでは、AWS DMS は、コレクションの作成と名前の変更を次のように管理します。

- 新しいコレクションをソースデータベースに追加すると、AWS DMS はコレクションの新しいターゲットテーブルを作成し、すべてのドキュメントをレプリケートします。
- ソースデータベースで既存コレクションの名前を変更すると、AWS DMS はターゲット テーブルの名前を変更しません。

ターゲット エンドポイントが Amazon DocumentDB の場合は、ドキュメントモードで移行を実行します。

テーブルモード

テーブルモードでは、AWS DMS は MongoDB ドキュメントのそれぞれの最上位フィールドをターゲットテーブルの列に変換します。フィールドがネストされている場合、AWS DMS はネストされた値を 1 つの列にフラット化します。次に AWS DMS は、キーフィールドとデータ型をターゲットテーブルの列セットに追加します。

各 MongoDB ドキュメントについては、AWS DMS はターゲットテーブルの列セットに各キーとタイプを追加します。たとえば、テーブルモードを使用すると、AWS DMS は前の例を次のテーブルに移行します。

oid_id	a	b	c
5a94815f4 0bd44d1b02bdfe0	1	2	3
5a94815f4 0bd44d1b02bdfe1	4	5	6

入れ子の値は、ドット区切りのキー名を含む列にフラット化されます。列は、ピリオドで区切られたフラット化されたフィールド名の連結と呼ばれます。たとえば、AWS DMS は、`{"a" : {"b" : {"c": 1}}}` のようなネストされた値のフィールドを持つ JSON ドキュメントを `a.b.c.` という名前の列に移行します。

ターゲット列を作成するため、AWS DMS は指定された数の MongoDB ドキュメントをスキャンして、すべてのフィールドおよびタイプのセットを作成します。次に AWS DMS は、このセットを使用してターゲットテーブルの列を作成します。コンソールを使用して MongoDB ソースエンドポイントを作成または変更する場合は、スキャンするドキュメントの数を指定できます。デフォルト値は 1000 ドキュメントです。AWS CLI を使用する場合は、追加の接続属性 `docsToInvestigate`を使用できます。

テーブルモードでは、AWS DMS は、ドキュメントとコレクションを次のように管理します。

- 既存のコレクションにドキュメント (行) を追加する場合は、ドキュメントがレプリケートされます。ターゲットに存在しないフィールドがある場合、それらのフィールドはレプリケーションされません。
- ドキュメントを更新すると、更新されたドキュメントはレプリケートされます。ターゲットに存在しないフィールドがある場合、それらのフィールドはレプリケーションされません。
- ドキュメントの削除は完全にサポートされています。
- CDC タスクの実行中に新しいコレクションを追加しても、ターゲット上に新しいテーブルは作成されません。
- 変更データキャプチャ (CDC) フェーズでは、AWS DMS はコレクション名の変更をサポートしていません。

トピック

- AWS DMS のソースとして MongoDB を使用する場合に必要なアクセス許可
- CDC 用 MongoDB レプリカセットの設定

- AWS DMS のソースとして MongoDB を使用する場合のセキュリティ要件
- MongoDB コレクションをセグメント化した並行移行
- AWS DMS のソースとして MongoDB を使用する際複数のデータベースを移行する
- MongoDB を AWS DMS のソースとして使用する場合の制限
- AWS DMS のソースとして MongoDB を使用する場合のエンドポイント設定
- MongoDB のソースデータ型

AWS DMS のソースとして MongoDB を使用する場合に必要なアクセス許可

MongoDB ソースを使用した AWS DMS の移行では、ルート権限を持つユーザーアカウントを作成するか、移行するデータベースに対してのみアクセス許可を持つユーザーを作成することができます。

次のコードは、ルートアカウントとなるユーザーを作成します。

```
use admin
db.createUser(
  {
    user: "root",
    pwd: "password",
    roles: [ { role: "root", db: "admin" } ]
  }
)
```

MongoDB 3.x ソースに関して次のコードは、移行するデータベースに対して最小限の権限を持つユーザーを作成します。

```
use database_to_migrate
db.createUser(
{
    user: "dms-user",
    pwd: "password",
    roles: [ { role: "read", db: "local" }, "read"]
})
```

MongoDB 4.x ソースの場合、以下のコードは最小限の権限を持つユーザーを作成します。

```
{ resource: { db: "", collection: "" }, actions: [ "find", "changeStream" ] }
```

たとえば、「admin」データベースに次のロールを作成します。

```
use admin
db.createRole(
{
role: "changestreamrole",
privileges: [
{ resource: { db: "", collection: "" }, actions: [ "find","changeStream" ] }
],
roles: []
}
)
```

ロールが作成されたら、移行するデータベースにユーザーを作成します。

```
> use test
> db.createUser(
{
user: "dms-user12345",
pwd: "password",
roles: [ { role: "changestreamrole", db: "admin" }, "read"]
})
```

CDC 用 MongoDB レプリカセットの設定

MongoDB で継続的レプリケーションまたは CDC を使用するには、AWS DMS が MongoDB オペレーションログ (oplog) にアクセスする必要があります。レプリカセットが存在しない場合に oplog を作成するには、レプリカセットをデプロイする必要があります。詳細については、MongoDB のドキュメントをご参照ください。

ソースエンドポイントとして設定された MongoDB レプリカのプライマリまたはセカンダリノードを持つ CDC を使用することができます。

スタンドアロンインスタンスをレプリカセットに変換するには

1. コマンドラインを使用して、mongo に接続します。

   ```
   mongo localhost
   ```

2. mongod サービスを停止します。

```
service mongod stop
```

3. 次のコマンドを使用して、mongod を再起動します。

```
mongod --replSet "rs0" --auth -port port_number
```

4. 次のコマンドを使用して、レプリカセットへの接続をテストします。

```
mongo -u root -p password --host rs0/localhost:port_number
  --authenticationDatabase "admin"
```

ドキュメントモードの移行を実行する予定がある場合は、MongoDB エンドポイントを作成するときに _id as a separate column オプションを選択します。このオプションを選択すると、プライマリキーとして機能する _id という名前の 2 番目の列が作成されます。この 2 番目の列は、AWS DMS がデータ操作言語 (DML) オペレーションをサポートするために必要です。

> **Note**
>
> AWS DMS 操作ログ (oplog) を使用して、継続的なレプリケーション中の変更をキャプチャします。AWS DMS が読み取る前に MongoDB が oplog からレコードをフラッシュすると、タスクは失敗します。少なくとも 24 時間は変更が保持されるように oplog のサイジングを行うことをお勧めします。

AWS DMS のソースとして MongoDB を使用する場合のセキュリティ要件

AWS DMS は、MongoDB で 2 つの認証方法をサポートしています。この 2 つの認証方法はパスワードを暗号化する際に使用するため、authType パラメータが PASSWORD に設定されているときにのみ使用されます。

MongoDB の認証方法は次のとおりです。

- MONGODB-CR — 下位互換性のために
- SCRAM-SHA-1 – バージョン 3.x と 4.0 認証を使用する場合のデフォルト

認証方法が指定されていない場合、AWS DMS は、MongoDB ソースバージョンのデフォルト方法を使用します。

MongoDB コレクションをセグメント化した並行移行

移行タスクのパフォーマンスを向上させるために、MongoDB ソース エンドポイントは、テーブルマッピングでの並列全ロードの 2 つのオプションをサポートしています。

つまり、自動セグメンテーションまたはテーブルマッピングによる範囲セグメンテーションを使用して、JSON 設定で並列全ロードを行うことで、コレクションを並行移行できます。自動セグメンテーションでは、AWS DMS の条件を指定して各スレッドで移行用のソースを自動的にセグメント化できます。範囲セグメンテーションを使用すると、DMS が各スレッドで移行する各セグメントの特定の範囲を AWS DMS に指定できます。これらの設定の詳細については、「テーブルとコレクション設定のルールとオペレーション」をご参照ください。

自動セグメンテーション範囲を使用して MongoDB データベースを並列移行する

スレッドごとにデータを自動的にパーティション (セグメント化) する AWS DMS の基準を指定することで、ドキュメントを並列移行できます。特に、スレッドごとに移行するドキュメントの数を指定します。このアプローチを使用して、AWS DMS は、スレッドあたりのパフォーマンスを最大化するために、セグメント境界を最適化しようとします。

テーブルマッピングで以下のテーブル設定オプションを使用して、セグメンテーション基準を指定できます。

テーブル設定オプション	説明
`"type"`	(必須) ソースとして `"partitions-auto"` for MongoDB を設定します。
`"number-of-partitions"`	(オプション) 移行に使用されるパーティション (セグメント) の総数。デフォルトは 16 です。
`"collection-count-from-metadata"`	(オプション) このオプションが `true` に設定されている場合,AWS DMS は、パーティションの数を決定するために、推定コレクション数を使用します。このオプションが `false` に設定されている場合、AWS DMS は実際のコレクション数を使用します。デフォルトは `true` です。
`"max-records-skip-per-page"`	(オプション) 各パーティションの境界を決定するときに一度にスキップするレコード数。AW

テーブル設定オプション	説明
	S DMS は、ページ割りスキップアプローチを使用して、パーティションの最小境界を決定します。デフォルトは 10,000 です。 比較的高い値を設定すると、カーソルがタイムアウトし、タスクが失敗する可能性がある。相対的に低い値を設定すると、ページあたりのオペレーションが増え、全ロードが遅くなります。
"batch-size"	(オプション) 1 つのバッチで返されるドキュメントの数を制限します。各バッチには、サーバーへの往復が必要です。バッチサイズがゼロ (0) の場合、カーソルはサーバー定義の最大バッチサイズを使用します。デフォルトは 0 です。

次の例は、自動セグメンテーションのテーブルマッピングを示しています。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "admin",
                "table-name": "departments"
            },
            "rule-action": "include",
            "filters": []
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "admin",
                "table-name": "departments"
```

```
            },
            "parallel-load": {
                "type": "partitions-auto",
                "number-of-partitions": 5,
                "collection-count-from-metadata": "true",
                "max-records-skip-per-page": 1000000,
                "batch-size": 50000
            }
        }
    ]
}
```

自動セグメンテーションには次のような制限があります。各セグメントの移行は、コレクション数とコレクション用の最小値 _id を別個にフェッチします。次に、ページ割りされたスキップを使用して、そのセグメントの最小境界を計算します。

従って、コレクション内のすべてのセグメント境界が計算されるまで、各コレクションで _id の最小値が一定のままであるようにします。セグメント境界の計算中にコレクションの _id 最小値を変更すると、データの損失や重複行のエラーが発生する可能性があります。

範囲セグメンテーションを使用して MongoDB データベースを並列移行する

スレッド内の各セグメント範囲を指定することで、ドキュメントを並列移行できます。このアプローチを使用して、AWS DMS に対してスレッドごとに選択したドキュメント範囲に従って、各スレッドで移行する特定のドキュメントを指定します。

次のイメージは、7 つの項目を持つ MongoDB コレクションとプライマリキーとしての _id を示しています。

Key	Value	Type
(1) ObjectId("5f805c74873173399a278d78")	{ 3 fields }	Object
_id	ObjectId("5f805c74873173399a278d78")	ObjectId
num	1	Int32
name	a	String
(2) ObjectId("5f805c97873173399a278d79")	{ 3 fields }	Object
_id	ObjectId("5f805c97873173399a278d79")	ObjectId
num	2	Int32
name	b	String
(3) ObjectId("5f805cb0873173399a278d7a")	{ 3 fields }	Object
_id	ObjectId("5f805cb0873173399a278d7a")	ObjectId
num	3	Int32
name	c	String
(4) ObjectId("5f805cbb873173399a278d7b")	{ 3 fields }	Object
_id	ObjectId("5f805cbb873173399a278d7b")	ObjectId
num	4	Int32
name	d	String
(5) ObjectId("5f805cc5873173399a278d7c")	{ 3 fields }	Object
_id	ObjectId("5f805cc5873173399a278d7c")	ObjectId
num	5	Int32
name	e	String
(6) ObjectId("5f805cd0873173399a278d7d")	{ 3 fields }	Object
_id	ObjectId("5f805cd0873173399a278d7d")	ObjectId
num	6	Int32
name	f	String
(7) ObjectId("5f805cdd873173399a278d7e")	{ 3 fields }	Object
_id	ObjectId("5f805cdd873173399a278d7e")	ObjectId
num	7	Int32
name	g	String

AWS DMS を並列移行させるためコレクションを 3 つの特定のセグメントに分割するには、移行タスクにテーブルマッピングルールを追加できます。これは次の JSON 例で示されます。

```
{ // Task table mappings:
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "testdatabase",
        "table-name": "testtable"
      },
      "rule-action": "include"
```

```
    }, // "selection" :"rule-type"
    {
      "rule-type": "table-settings",
      "rule-id": "2",
      "rule-name": "2",
      "object-locator": {
        "schema-name": "testdatabase",
        "table-name": "testtable"
      },
      "parallel-load": {
        "type": "ranges",
        "columns": [
           "_id",
           "num"
        ],
        "boundaries": [
          // First segment selects documents with _id less-than-or-equal-to
 5f805c97873173399a278d79
          // and num less-than-or-equal-to 2.
          [
             "5f805c97873173399a278d79",
             "2"
          ],
          // Second segment selects documents with _id > 5f805c97873173399a278d79 and
          // _id less-than-or-equal-to 5f805cc5873173399a278d7c and
          // num > 2 and num less-than-or-equal-to 5.
          [
             "5f805cc5873173399a278d7c",
             "5"
          ]
          // Third segment is implied and selects documents with _id >
 5f805cc5873173399a278d7c.
       ] // :"boundaries"
      } // :"parallel-load"
    } // "table-settings" :"rule-type"
  ] // :"rules"
} // :Task table mappings
```

このテーブルマッピング定義は、ソースコレクションを 3 つのセグメントに分割し、並列移行します。以下は、セグメンテーション境界です。

```
Data with _id less-than-or-equal-to "5f805c97873173399a278d79" and num less-than-or-
equal-to 2 (2 records)
Data with _id > "5f805c97873173399a278d79" and num > 2 and _id  less-than-or-equal-to
 "5f805cc5873173399a278d7c" and num less-than-or-equal-to 5 (3 records)
Data with _id > "5f805cc5873173399a278d7c" and num > 5 (2 records)
```

移行タスクが完了したら、次の例に示すように、テーブルが並列にロードされたことをタスクログから確認できます。ソーステーブルから各セグメントをアンロードするために使用する MongoDB `find` 句 を確認することもできます。

```
[TASK_MANAGER    ] I:  Start loading segment #1 of 3 of table
 'testdatabase'.'testtable' (Id = 1) by subtask 1. Start load timestamp
 0005B191D638FE86  (replicationtask_util.c:752)

[SOURCE_UNLOAD   ] I:  Range Segmentation filter for Segment #0 is initialized.
 (mongodb_unload.c:157)

[SOURCE_UNLOAD   ] I:  Range Segmentation filter for Segment #0 is: { "_id" :
 { "$lte" : { "$oid" : "5f805c97873173399a278d79" } }, "num" : { "$lte" :
 { "$numberInt" : "2" } } }  (mongodb_unload.c:328)

[SOURCE_UNLOAD   ] I: Unload finished for segment #1 of segmented table
 'testdatabase'.'testtable' (Id = 1). 2 rows sent.

[TASK_MANAGER    ] I: Start loading segment #1 of 3 of table
 'testdatabase'.'testtable' (Id = 1) by subtask 1. Start load timestamp
 0005B191D638FE86 (replicationtask_util.c:752)

[SOURCE_UNLOAD   ] I: Range Segmentation filter for Segment #0 is initialized.
 (mongodb_unload.c:157)

[SOURCE_UNLOAD   ] I: Range Segmentation filter for Segment #0 is: { "_id" : { "$lte" :
 { "$oid" : "5f805c97873173399a278d79" } }, "num" : { "$lte" : { "$numberInt" :
 "2" } } } (mongodb_unload.c:328)

[SOURCE_UNLOAD   ] I: Unload finished for segment #1 of segmented table
 'testdatabase'.'testtable' (Id = 1). 2 rows sent.

[TARGET_LOAD     ] I: Load finished for segment #1 of segmented table
 'testdatabase'.'testtable' (Id = 1). 1 rows received. 0 rows skipped. Volume
 transfered 480.
```

```
[TASK_MANAGER    ] I: Load finished for segment #1 of table
 'testdatabase'.'testtable' (Id = 1) by subtask 1. 2 records transferred.
```

現在、AWS DMS は、セグメントキー列として次の MongoDB データ型をサポートしています。

- ダブル
- 文字列
- ObjectId
- 32 ビット整数
- 64 ビット整数

AWS DMS のソースとして MongoDB を使用する際複数のデータベースを移行する

AWS DMS バージョン 3.4.5 以降では、サポートされているすべての MongoDB バージョンについて、単一のタスクで複数のデータベースを移行できます。複数のデータベースを移行する場合は、次のステップを実行します：

1. MongoDB ソース エンドポイントを作成するときは、次のいずれかの操作を行います：
   - DMS コンソールのエンドポイントの作成ページで、[Endpoint configuration] (エンドポイント設定) の下にある[Database name] (データベース名) が空であることを確認してください。
   - AWS CLI `CreateEndpoint` コマンドを使用して、空の文字列値を `MongoDBSettings` の `DatabaseName` パラメータに割り当てます。
2. MongoDB ソースから移行するデータベースごとに、タスクのテーブルマッピングにおけるスキーマ名としてのデータベース名を指定します。これを行うには、コンソール内のガイド付き入力を使用するか、JSON で直接入力できます。ガイド付き入力の詳細については、「コンソールからテーブル選択および変換を指定する」をご参照ください。JSON の詳細については、「選択ルールと選択アクション」をご参照ください。

たとえば、次の JSON を指定して、3 つの MongoDB データベースを移行するとします。

Example スキーマ内のすべてのテーブルの移行

次の JSON は、すべてのテーブルをソースエンドポイントの `Customers`、`Orders`、`Suppliers` データベースからターゲット エンドポイントに移行します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Customers",
                "table-name": "%"
            },
            "rule-action": "include",
            "filters": []
        },
        {
            "rule-type": "selection",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "Orders",
                "table-name": "%"
            },
            "rule-action": "include",
            "filters": []
        },
        {
            "rule-type": "selection",
            "rule-id": "3",
            "rule-name": "3",
            "object-locator": {
                "schema-name": "Inventory",
                "table-name": "%"
            },
            "rule-action": "include",
            "filters": []
        }
    ]
}
```

MongoDB を AWS DMS のソースとして使用する場合の制限

MongoDB を AWS DMS のソースとして使用する場合の制限は次のとおりです。

- テーブルモードの場合、コレクション内のドキュメントは、同じフィールドの値で使用されるデータ型と一致している必要があります。例えば、コレクション内のドキュメントに '{ a: { b:*value* ... }' が含まれている場合、a.b フィールドの *value* を参照するコレクション内のすべてのドキュメントは、値が登場するコレクションの場所を問わず、*value* で同じデータ型を使用する必要があります。
- _id オプションが別の列として設定されている場合、ID 文字列は 200 文字以下でなければなりません。
- オブジェクト ID および配列タイプキーは、テーブルモードで oid および array というプレフィックスが付けられた列に変換されます。

  内部的には、これらの列はプレフィックスが付けられた名前で参照されます。これらの列を参照する変換ルールを AWS DMS で使用する場合は、プレフィックス列を指定します。たとえば、${_id} ではなく ${oid__id} を指定するか、${_addresses} ではなく ${array__addresses} を指定します。
- コレクション名とキー名にドル記号 ($) を含めることはできません。
- AWS DMS は、RDBMS ターゲットを使用したテーブルモードで、大文字と小文字 (upperm lower) が異なる同じフィールドがあるコレクションをサポートしていません。例えば、AWS DMS は、Field1 と field1 という名前の 2 つのコレクションがあることをサポートしていません。
- 前述のように、テーブルモードとドキュメントモードには制限があります。
- 自動セグメンテーションを使用して並列移行するには、前述の制限があります。
- ソースフィルターは、MongoDB ではサポートされていません。
- AWS DMS は、ネストレベルが 97 を超えるドキュメントをサポートしていません。
- AWS DMS は、MongoDB バージョン 5.0 の次の機能をサポートしていません。
  - ライブリシャーディング
  - クライアント側のフィールドレベルの暗号化 (CSFLE)
  - 時系列コレクションの移行

> **Note**
>
> DocumentDB は、時系列コレクションをサポートしていないため、フルロードフェーズで移行された時系列コレクションは Amazon DocumentDB の通常のコレクションに変換されます。

AWS DMS のソースとして MongoDB を使用する場合のエンドポイント設定

MongoDB ソース エンドポイントを設定する場合、AWS DMSコンソールを使用して、複数のエンドポイント設定を指定できます。

次の表は、AWS DMS のソースとして MongoDB データベースを使用する際利用可能な設定についてです。

設定 (属性)	有効値	デフォルト値と説明
[Authentication mode] (認証モード)	`"none"` `"password"`	値 `"password"` では、ユーザー名とパスワードの入力が求められます。`"none"` を指定した場合、ユーザー名とパスワードのパラメータは使用されません。
[Authentication source] (認証ソース)	有効な MongoDB データベース名を指定してください。	認証情報を検証するために使用する MongoDB データベースの名前。デフォルト値は、`"admin"`です。

[Authentication mechanism] (認証メカニズム)	`"default"`
	`"mongodb_cr"`
	`"scram_sha_1"`
		認証メカニズム。 `"default"` 値は `"scram_sha_1"` です。`authType` が `"no"` に設定されている場合、この設定は使用されません。

[Metadata mode] (メタデータモード)	ドキュメントとテーブル	ドキュメントモードまたはテーブルモードを選択します。
スキャンするドキュメントの数 (`docsToInvestigate`)	`0` より大きい正の整数。	このオプションは、テーブルモードでのみターゲットテーブル定義を定義するために使用します。
[_id as a separate column] (_id を個別の列	チェックボックス	このチェックを入れると、プライマリ キーとして機能する `_id` という名前の 2 番目の列が作成されます。

設定 (属性)	有効値	デフォルト値と説明
として指定する)		
socketTimeoutMS	NUMBER 追加の接続属性 (ECA) のみ	この設定はミリ秒単位で、MongoDB クライアントの接続タイムアウトを設定する。値が 0 以下の場合、MongoDB クライアントのデフォルトが使用される。
UseUpdateLookUp	boolean true false	true の場合、CDC 更新イベント中に、AWS DMS は更新されたドキュメント全体をターゲットにコピーする。false に設定すると、AWS DMS は MongoDB の update コマンドを使用して、ターゲットのドキュメント内の変更されたフィールドのみを更新する。
ReplicateShardCollections	boolean true false	true の場合、AWS DMS はデータをシャードコレクションにレプリケートする。AWS DMS は、ターゲットエンドポイントが DocumentDB の伸縮自在なクラスターである場合にのみこの設定を使用する。 この設定が true の場合、次の点に注意する。 • TargetTablePrepMode を nothing に設定する必要がある。 • AWS DMS は自動的に useUpdateLookup を false に設定する。

ドキュメントをメタデータモードとして選択した場合、さまざまなオプションを利用できます。

ターゲット エンドポイントが DocumentDB の場合は、ドキュメントモードで移行を実行し、また、ソース エンドポイントを変更し、[_id as separate column] (_id を個別の列として) オプションを選択します。ソースの MongoDB ワークロードにトランザクションが含まれる場合、これは必須の前提条件です。

MongoDB のソースデータ型

AWS DMS のソースとして MongoDB を使用するデータ移行では、ほとんどの MongoDB データ型がサポートされます。次のテーブルに、AWS DMS を使用する場合にサポートされる MongoDB の

ソースデータ型と、AWS DMS のデータ型からのデフォルトマッピングを示します。MongoDB データ型の詳細については、『MongoDB のドキュメント』の「BSON 型」をご参照ください。

ターゲットにマッピングされるデータ型を表示する方法については、使用しているターゲットエンドポイントのセクションをご参照ください。

AWS DMS のデータ型の詳細については、「AWS Database Migration Service のデータ型」をご参照ください。

MongoDB データ型	AWS DMS データ型
ブール値	Bool
バイナリ	BLOB
日付	日付
タイムスタンプ	日付
Int	INT4
Long	INT8
ダブル	REAL8
String (UTF-8)	CLOB
配列	CLOB
OID	文字列
REGEX	CLOB
コード	CLOB

のソースとしての Amazon DocumentDB (MongoDB 互換) の使用 AWS DMS

AWS DMS がソースとしてサポートする Amazon DocumentDB (MongoDB 互換) のバージョンについては、「のソース AWS DMS」を参照してください。

ソースとして Amazon DocumentDB を使用すると、1 つの Amazon DocumentDB クラスターから別のAmazon DocumentDB クラスターにデータを移行できます。Amazon DocumentDB クラスターから、 でサポートされている他のターゲットエンドポイントのいずれかにデータを移行することもできます AWS DMS。

Amazon DocumentDB を初めて使用する場合は、Amazon DocumentDB データベースの以下の重要な概念に注意してください：

- Amazon DocumentDB のレコードは、フィールドと値のペアで構成されるデータ構造であるドキュメントです。フィールドの値には、他のドキュメント、配列、およびドキュメントの配列を含めることができます。ドキュメントは、リレーショナルデータベースのテーブルの行とほぼ同等です。
- Amazon DocumentDB でのコレクションとは、ドキュメントのグループであり、リレーショナルデータベーステーブルとほぼ同等です。
- Amazon DocumentDB の データベースはコレクションのセットであり、リレーショナルデータベース内のスキーマとほぼ同等です。

AWS DMS は、Amazon DocumentDB をソースとして使用する場合、ドキュメントモードとテーブルモードの 2 つの移行モードをサポートします。 AWS DMS コンソールで Amazon DocumentDB ソースエンドポイントを作成するときに、メタデータモードオプション または追加の接続属性 を使用して移行モードを指定しますnestingLevel。移行モードの選択は、以下に説明するように、ターゲットデータの結果の形式に影響します。

ドキュメントモード

ドキュメントモードでは、JSON ドキュメントがそのまま移行されます。つまり、ドキュメントデータは 2 つの項目のいずれかに統合されます。ターゲットとしてリレーショナルデータベースを使用する場合、データはターゲットテーブルで _doc という名前の単一の列になります。非リレーショナルデータベースをターゲットとして使用する場合、データは単一の JSON ドキュメントになります。ドキュメントモードは、Amazon DocumentDB ターゲットに移行する際にお勧めするデフォルトモードです。

myCollection という Amazon DocumentDB コレクション内の次のドキュメントを例にとって説明します。

```
> db.myCollection.find()
{ "_id" : ObjectId("5a94815f40bd44d1b02bdfe0"), "a" : 1, "b" : 2, "c" : 3 }
{ "_id" : ObjectId("5a94815f40bd44d1b02bdfe1"), "a" : 4, "b" : 5, "c" : 6 }
```

ドキュメントモードを使用してデータをリレーショナルデータベーステーブルに移行した後、データは以下のように構成されます。ドキュメントのデータフィールドは _doc 列に統合されています。

oid_id	_doc
5a94815f40bd44d1b02bdfe0	{ "a" : 1, "b" : 2, "c" : 3 }
5a94815f40bd44d1b02bdfe1	{ "a" : 4, "b" : 5, "c" : 6 }

オプションで、追加の接続属性 `extractDocID` を `true` に設定して、プライマリ キーとして機能する "_id" という名前の 2 つ目の列を作成できます。変更データ キャプチャ (CDC) を使用する場合は、ターゲットとして Amazon DocumentDB を使用する場合を除き、このパラメータを `true` に設定します。

> **Note**
>
> ソースデータベースに新しいコレクションを追加すると、 はコレクションの新しいターゲットテーブル AWS DMS を作成し、ドキュメントをレプリケートします。

テーブルモード

テーブルモードでは、AWS DMS は Amazon DocumentDB ドキュメントの各最上位フィールドをターゲットテーブルの列に変換します。フィールドがネストされている場合、 はネストされた値を 1 つの列に AWS DMS フラット化します。 AWS DMS は、キーフィールドとデータ型をターゲットテーブルの列セットに追加します。

Amazon DocumentDB ドキュメントごとに、 は各キーとタイプをターゲットテーブルの列セット AWS DMS に追加します。例えば、テーブルモードを使用すると、 は前の例を次のテーブル AWS DMS に移行します。

oid_id	a	b	c
5a94815f4 0bd44d1b02bdfe0	1	2	3

5a94815f4 0bd44d1b02bdfe1	4	5	6

入れ子の値は、ドット区切りのキー名を含む列にフラット化されます。この列は、ピリオドで区切られた平滑化されたフィールド名の連結を使用して名前が付けられます。例えば、 は などのネストされた値のフィールドを持つ JSON ドキュメント`{"a" : {"b" : {"c": 1}}}`を という名前の列 AWS DMS に移行します。 `a.b.c`.

ターゲット列を作成するには、 は指定された数の Amazon DocumentDB ドキュメント AWS DMS をスキャンし、すべてのフィールドとそのタイプのセットを作成します。 AWS DMS 次に、このセットを使用してターゲットテーブルの列を作成します。コンソールを使用して Amazon DocumentDB ソース エンドポイントを作成または変更する場合は、スキャンするドキュメントの数を指定できます。デフォルト値は 1000 ドキュメントです。を使用する場合は AWS CLI、追加の接続属性 を使用できます`docsToInvestigate`。

テーブルモードでは、 は次のようなドキュメントとコレクション AWS DMS を管理します。

- 既存のコレクションにドキュメント (行) を追加する場合は、ドキュメントがレプリケートされます。ターゲットに存在しないフィールドがある場合、それらのフィールドはレプリケーションされません。
- ドキュメントを更新すると、更新されたドキュメントはレプリケートされます。ターゲットに存在しないフィールドがある場合、それらのフィールドはレプリケーションされません。
- ドキュメントの削除は完全にサポートされています。
- CDC タスクの実行中に新しいコレクションを追加しても、ターゲット上に新しいテーブルは作成されません。
- 変更データキャプチャ (CDC) AWS DMS フェーズでは、コレクションの名前変更はサポートされていません。

トピック

- ソースとして Amazon DocumentDB を使用するためのアクセス許可の設定
- Amazon DocumentDB クラスター用 CDC の設定
- TLS を使用して Amazon DocumentDB に接続する
- Amazon DocumentDB ソース エンドポイントの作成
- Amazon DocumentDB コレクションをセグメント化し、並列移行する
- のソースとして Amazon DocumentDB を使用する場合の複数のデータベースの移行 AWS DMS

- のソースとして Amazon DocumentDB を使用する場合の制限 AWS DMS
- ソースとしての Amazon DocumentDB のエンドポイントの設定
- Amazon DocumentDB のソースデータ型

ソースとして Amazon DocumentDB を使用するためのアクセス許可の設定

AWS DMS 移行に Amazon DocumentDB ソースを使用する場合、ルート権限を持つユーザーアカウントを作成できます。または、移行するデータベースに対してのみ許可されたユーザーを作成することもできます。

次のコードは、ルートアカウントとしてのユーザーを作成します。

```
use admin
db.createUser(
  {
    user: "root",
    pwd: "password",
    roles: [ { role: "root", db: "admin" } ]
  })
```

Amazon DocumentDB 3.6 の場合、次のコードは、移行するデータベースに対して最小限の権限を持つユーザーを作成します。

```
use database_to_migrate
db.createUser(
{
    user: "dms-user",
    pwd: "password",
    roles: [ { role: "read", db: "db_name" }, "read"]
})
```

Amazon DocumentDB 4.0 以降では、デプロイ全体の変更ストリーム AWS DMS を使用します。この例では、次のコードを使用して最小限の権限を持つユーザーを作成します。

```
db.createUser(
{
    user: "dms-user",
    pwd: "password",
```

```
    roles: [ { role: "readAnyDatabase", db: "admin" }]
})
```

Amazon DocumentDB クラスター用 CDC の設定

Amazon DocumentDB で継続的なレプリケーションまたは CDC を使用するには、Amazon DocumentDB クラスターの変更ストリームへのアクセス AWS DMS が必要です。クラスターのコレクションおよびデータベース内の更新イベントの時系列については、Amazon DocumentDB デベロッパー+ガイドの「変更ストリームの使用」をご参照ください。

MongoDB シェルを使用して Amazon DocumentDB クラスターに関して認証します。次に、以下のコマンドを実行して、変更ストリームを有効にします。

```
db.adminCommand({modifyChangeStreams: 1,
    database: "DB_NAME",
    collection: "",
    enable: true});
```

この方法により、データベース内ですべてのコレクションの変更ストリームが有効になります。変更ストリームを有効にすると、既存のデータを移行し、同時に進行中の変更をレプリケートする移行タスクを作成できます。 AWS DMS は、バルクデータがロードされた後でも変更をキャプチャして適用します。最終的にソースデータベースとターゲットデータベースは同期され、移行でのダウンタイムは最小限に抑えられます。

> **Note**
>
> AWS DMS は、オペレーションログ (oplog) を使用して、継続的なレプリケーション中の変更をキャプチャします。が AWS DMS レコードを読み取る前に Amazon DocumentDB が oplog からレコードをフラッシュすると、タスクは失敗します。少なくとも 24 時間は変更が保持されるように oplog のサイジングを行うことをお勧めします。

TLS を使用して Amazon DocumentDB に接続する

デフォルトでは、新しく作成された Amazon DocumentDB クラスターは、Transport Layer Security (TLS) を使用したセキュアな接続のみを受け入れます。TLS が有効になっている場合、Amazon DocumentDB へのすべての接続で公開キーが必要になります。

Amazon DocumentDB のパブリックキーを取得するには、 がホストする Amazon AWS S3 バケットrds-combined-ca-bundle.pemから ファイルをダウンロードします。 Amazon S3 このファイルのダウンロードの詳細については、Amazon DocumentDB デベロッパーガイドの「TLS を使用した接続の暗号化」をご参照ください。

rds-combined-ca-bundle.pem ファイルをダウンロードしたら、ファイルに含まれるパブリックキーを にインポートできます AWS DMS。以下のステップでは、その方法について説明します。

AWS DMS コンソールを使用してパブリックキーをインポートするには

1. にサインイン AWS Management Console し、 を選択します AWS DMS。
2. ナビゲーションペインで [Certificates] を選択します。
3. [証明書のインポート] を選択します。[Import certificate] (新しい CA 証明書のインポート) ページが表示されます。
4. [Certificate configuration] (証明書設定) セクションで、次のいずれかの操作を行います：
   - [Certificate identifier] (証明書の識別子) に、docdb-cert などの、証明書の一意の名前を入力します。
   - [Choose file] (ファイルの選択) を選択し、rds-combined-ca-bundle.pem ファイルを保存した場所に移動し、選択します。
5. [Add new CA certificate (新しい CA 証明書の追加)] を選択します。

AWS CLI 次の例では、 AWS DMS import-certificate コマンドを使用してパブリックキーrds-combined-ca-bundle.pemファイルをインポートします。

```
aws dms import-certificate \
    --certificate-identifier docdb-cert \
    --certificate-pem file://./rds-combined-ca-bundle.pem
```

Amazon DocumentDB ソース エンドポイントの作成

Amazon DocumentDB ソース エンドポイントを作成するには、コンソールまたは AWS CLIを使用します。以下の手順では、コンソールを使用します。

AWS DMS コンソールを使用して Amazon DocumentDB ソースエンドポイントを設定するには

1. にサインイン AWS Management Console し、 を選択します AWS DMS。

2. ナビゲーションペインで [Endpoints] (エンドポイント) を選択し、[Create Endpoint] (エンドポイントの作成) を選択します。
3. [Endpoint identifier] (エンドポイント識別子) には、`docdb-source` などの簡単に識別できるような名前を入力します。
4. [Source engine] (ソースエンジン) には、Amazon DocumentDB (MongoDB 互換性) を選択します。
5. を使用する場合[Server name] (サーバー名) には、Amazon DocumentDB データベース エンドポイントが常駐するサーバー名を入力します。たとえば、Amazon EC2 インスタンスの公開 DNS 名に `democluster.cluster-cjf6q8nxfefi.us-east-2.docdb.amazonaws.com` などを入力できます。
6. [Port] (ポート) に 27017 と入力します。
7. [SSL mode (SSL モード)] で [verify-full] を選択します。Amazon DocumentDB クラスターで SSL を無効化している場合、このステップは省略できます。
8. [CA certificate] (CA 証明書) には、Amazon DocumentDB 証明書として `rds-combined-ca-bundle.pem` を選択します。この証明書の追加手順については、「TLS を使用して Amazon DocumentDB に接続する」をご参照ください。
9. [Database name] (データベース名) に、移行するデータベースの名前を入力します。

CLI に次の手順を値で使用します。

を使用して Amazon DocumentDB ソースエンドポイントを設定するには AWS CLI

- 次の AWS DMS `create-endpoint`コマンドを実行して Amazon DocumentDB ソースエンドポイントを設定し、プレースホルダーを独自の値に置き換えます。

```
aws dms create-endpoint \
           --endpoint-identifier a_memorable_name \
           --endpoint-type source \
           --engine-name docdb \
           --username value \
           --password value \
           --server-name servername_where_database_endpoint_resides \
           --port 27017 \
           --database-name name_of_endpoint_database
```

Amazon DocumentDB コレクションをセグメント化し、並列移行する

移行タスクのパフォーマンスを向上させるために、Amazon DocumentDB ソース エンドポイントは、テーブルマッピングの並列全ロード機能の 2 つのオプションをサポートしています。つまり、JSON 設定で並列全ロードに対してテーブルマッピングの自動セグメンテーションまたは範囲セグメンテーションオプションを使用して、コレクションを並列移行できます。自動セグメンテーションオプションを使用すると、 の基準を指定 AWS DMS して、各スレッドで移行するソースを自動的にセグメント化できます。範囲セグメンテーションオプションを使用すると、DMS が各スレッドで移行する各セグメントの AWS DMS 特定の範囲を指定できます。これらの設定の詳細については、「テーブルとコレクション設定のルールとオペレーション」をご参照ください。

自動セグメンテーション範囲を使用して Amazon DocumentDB データベースを並列移行する

各スレッド、特にスレッドごとに移行するドキュメントの数に対してデータを自動的にパーティション化 (セグメント化) AWS DMS する の基準を指定することで、ドキュメントを並行して移行できます。このアプローチを使用して、 AWS DMS は、スレッドあたりのパフォーマンスを最大化するために、セグメント境界を最適化しようとします。

テーブルマッピングで次のテーブル設定オプションを使用して、セグメンテーション基準を指定できます。

テーブル設定オプション	説明
`"type"`	(必須) ソースとして Amazon DocumentDB 用の `"partitions-auto"` に設定します。
`"number-of-partitions"`	(オプション) 移行に使用されるパーティション (セグメント) の総数。デフォルトは 16 です。
`"collection-count-from-metadata"`	(オプション) に設定すると`true`、パーティションの数を決定するために推定コレクション数 AWS DMS が使用されます。に設定すると`false`、実際のコレクション数 AWS DMS が使用されます。デフォルトは `true` です。
`"max-records-skip-per-page"`	(オプション) 各パーティションの境界を決定するときに一度にスキップするレコードの数。は、ページ分割されたスキップアプローチ AWS DMS を使用してパーティションの最小境界を決

テーブル設定オプション	説明
	定します。デフォルトは 10,000 です。比較的大きな値を設定すると、カーソルのタイムアウトやタスク障害の可能性があります。相対的に低い値を設定すると、ページあたりのオペレーションが増え、全ロードが遅くなります。
"batch-size"	(オプション) 1 つのバッチで返されるドキュメントの数を制限します。各バッチには、サーバーへの往復が必要です。バッチサイズがゼロ (0) の場合、カーソルはサーバー定義の最大バッチサイズを使用します。デフォルトは 0 です。

次の例は、自動セグメンテーションのテーブルマッピングを示しています。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "admin",
                "table-name": "departments"
            },
            "rule-action": "include",
            "filters": []
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "admin",
                "table-name": "departments"
            },
            "parallel-load": {
                "type": "partitions-auto",
                "number-of-partitions": 5,
                "collection-count-from-metadata": "true",
```

```
                "max-records-skip-per-page": 1000000,
                "batch-size": 50000
            }
        }
    ]
}
```

自動セグメンテーションには次の制限があります。各セグメントの移行は、コレクション数とコレクション用の最小値 _id を別個にフェッチします。次に、ページ割りされたスキップを使用して、そのセグメントの最小境界を計算します。従って、コレクション内のすべてのセグメント境界が計算されるまで、各コレクションで _id の最小値が一定のままであるようにします。セグメント境界の計算中にコレクションの最小 _id 値を変更すると、データ損失や重複行のエラーが発生する可能性があります。

特定のセグメント範囲を使用して Amazon DocumentDB データベースを並列移行する

次の例は、7 つの項目を持つ Amazon DocumentDB コレクションおよび _id をプライマリ キーとして示しています。

Key	Value	Type
(1) ObjectId("5f805c74873173399a278d78")	{ 3 fields }	Object
_id	ObjectId("5f805c74873173399a278d78")	ObjectId
num	1	Int32
name	a	String
(2) ObjectId("5f805c97873173399a278d79")	{ 3 fields }	Object
_id	ObjectId("5f805c97873173399a278d79")	ObjectId
num	2	Int32
name	b	String
(3) ObjectId("5f805cb0873173399a278d7a")	{ 3 fields }	Object
_id	ObjectId("5f805cb0873173399a278d7a")	ObjectId
num	3	Int32
name	c	String
(4) ObjectId("5f805cbb873173399a278d7b")	{ 3 fields }	Object
_id	ObjectId("5f805cbb873173399a278d7b")	ObjectId
num	4	Int32
name	d	String
(5) ObjectId("5f805cc5873173399a278d7c")	{ 3 fields }	Object
_id	ObjectId("5f805cc5873173399a278d7c")	ObjectId
num	5	Int32
name	e	String
(6) ObjectId("5f805cd0873173399a278d7d")	{ 3 fields }	Object
_id	ObjectId("5f805cd0873173399a278d7d")	ObjectId
num	6	Int32
name	f	String
(7) ObjectId("5f805cdd873173399a278d7e")	{ 3 fields }	Object
_id	ObjectId("5f805cdd873173399a278d7e")	ObjectId
num	7	Int32
name	g	String

コレクションを 3 つのセグメントに分割して並列に移行するには、次の JSON の例に示すように、移行タスクにテーブルマッピングルールを追加します。

```
{ // Task table mappings:
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "testdatabase",
        "table-name": "testtable"
      },
      "rule-action": "include"
```

```
    }, // "selection" :"rule-type"
    {
      "rule-type": "table-settings",
      "rule-id": "2",
      "rule-name": "2",
      "object-locator": {
        "schema-name": "testdatabase",
        "table-name": "testtable"
      },
      "parallel-load": {
        "type": "ranges",
        "columns": [
           "_id",
           "num"
        ],
        "boundaries": [
          // First segment selects documents with _id less-than-or-equal-to
 5f805c97873173399a278d79
          // and num less-than-or-equal-to 2.
          [
             "5f805c97873173399a278d79",
             "2"
          ],
          // Second segment selects documents with _id > 5f805c97873173399a278d79 and
          // _id less-than-or-equal-to 5f805cc5873173399a278d7c and
          // num > 2 and num less-than-or-equal-to 5.
          [
             "5f805cc5873173399a278d7c",
             "5"
          ]
          // Third segment is implied and selects documents with _id >
 5f805cc5873173399a278d7c.
       ] // :"boundaries"
      } // :"parallel-load"
    } // "table-settings" :"rule-type"
  ] // :"rules"
} // :Task table mappings
```

このテーブルマッピング定義は、ソースコレクションを 3 つのセグメントに分割し、並列移行します。以下は、セグメンテーション境界です。

```
Data with _id less-than-or-equal-to "5f805c97873173399a278d79" and num less-than-or-
equal-to 2 (2 records)
Data with _id less-than-or-equal-to "5f805cc5873173399a278d7c" and num less-than-or-
equal-to 5 and not in (_id less-than-or-equal-to  "5f805c97873173399a278d79" and num
 less-than-or-equal-to 2) (3 records)
Data not in (_id less-than-or-equal-to "5f805cc5873173399a278d7c" and num less-than-or-
equal-to 5) (2 records)
```

移行タスクが完了したら、次の例に示すように、テーブルが並列にロードされたことをタスクログから確認できます。ソーステーブルから各セグメントをアンロードするために使用される Amazon DocumentDB `find` 句を確認することもできます。

```
[TASK_MANAGER    ] I:  Start loading segment #1 of 3 of table
 'testdatabase'.'testtable' (Id = 1) by subtask 1. Start load timestamp
 0005B191D638FE86  (replicationtask_util.c:752)

[SOURCE_UNLOAD   ] I:  Range Segmentation filter for Segment #0 is initialized.
 (mongodb_unload.c:157)

[SOURCE_UNLOAD   ] I:  Range Segmentation filter for Segment #0 is: { "_id" :
 { "$lte" : { "$oid" : "5f805c97873173399a278d79" } }, "num" : { "$lte" :
 { "$numberInt" : "2" } } }  (mongodb_unload.c:328)

[SOURCE_UNLOAD   ] I: Unload finished for segment #1 of segmented table
 'testdatabase'.'testtable' (Id = 1). 2 rows sent.

[TASK_MANAGER    ] I: Start loading segment #1 of 3 of table
 'testdatabase'.'testtable' (Id = 1) by subtask 1. Start load timestamp
 0005B191D638FE86 (replicationtask_util.c:752)

[SOURCE_UNLOAD   ] I: Range Segmentation filter for Segment #0 is initialized.
 (mongodb_unload.c:157)

[SOURCE_UNLOAD   ] I: Range Segmentation filter for Segment #0 is: { "_id" : { "$lte" :
 { "$oid" : "5f805c97873173399a278d79" } }, "num" : { "$lte" : { "$numberInt" :
 "2" } } } (mongodb_unload.c:328)

[SOURCE_UNLOAD   ] I: Unload finished for segment #1 of segmented table
 'testdatabase'.'testtable' (Id = 1). 2 rows sent.
```

```
[TARGET_LOAD     ] I: Load finished for segment #1 of segmented table
 'testdatabase'.'testtable' (Id = 1). 1 rows received. 0 rows skipped. Volume
 transfered 480.

[TASK_MANAGER    ] I: Load finished for segment #1 of table
 'testdatabase'.'testtable' (Id = 1) by subtask 1. 2 records transferred.
```

現在、 はセグメントキー列として次の Amazon DocumentDB データ型 AWS DMS をサポートしています。

- ダブル
- 文字列
- ObjectId
- 32 ビット整数
- 64 ビット整数

のソースとして Amazon DocumentDB を使用する場合の複数のデータベースの移行 AWS DMS

AWS DMS バージョン 3.4.5 以降では、Amazon DocumentDB バージョン 4.0 以降でのみ、1 つのタスクで複数のデータベースを移行できます。複数のデータベースを移行する場合は、以下を実行します：

1. Amazon DocumentDB ソース エンドポイントを作成すると、次のようになります：
   - AWS Management Console の で AWS DMS、エンドポイントの作成ページのエンドポイント設定でデータベース名を空のままにします。
   - AWS Command Line Interface （ AWS CLI) で、CreateEndpointアクションに指定した DocumentDBSettings の DatabaseNameパラメータに空の文字列値を割り当てます。
2. この Amazon DocumentDB ソース エンドポイントから移行するデータベースごとに、コンソールでのガイド付き入力を使用するか、JSON で直接使用するタスクのテーブルマッピングで、各データベースの名前をスキーマの名前として指定します。ガイド付き入力の詳細については、「コンソールからテーブル選択および変換を指定する」をご参照ください。。JSON の詳細については、「選択ルールと選択アクション」をご参照ください。

たとえば、次の JSON を指定して 3 つの Amazon DocumentDB データベースを移行できます。

Example スキーマ内のすべてのテーブルの移行

次の JSON は、ソース エンポイント内の Customers、Orders、Suppliersデータベースからすべてのテーブルをターゲット エンドポイントに移行します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Customers",
                "table-name": "%"
            },
            "object-locator": {
                "schema-name": "Orders",
                "table-name": "%"
            },
            "object-locator": {
                "schema-name": "Inventory",
                "table-name": "%"
            },
            "rule-action": "include"
        }
    ]
}
```

のソースとして Amazon DocumentDB を使用する場合の制限 AWS DMS

のソースとして Amazon DocumentDB を使用する場合の制限事項を次に示します AWS DMS。

- _id オプションが別の列として設定されている場合、ID 文字列は 200 文字以下でなければなりません。
- オブジェクト ID および配列タイプキーは、テーブルモードで oid および array というプレフィックスが付けられた列に変換されます。

  内部的には、これらの列はプレフィックスが付けられた名前で参照されます。これらの列を参照 AWS DMS する変換ルールを で使用する場合は、必ずプレフィックス付き列を指定してください。たとえば、${_id} ではなく ${oid__id} を指定するか、${_addresses} ではなく ${array__addresses} を指定します。

- コレクション名とキー名にドル記号 ($) を含めることはできません。
- 前述のように、テーブルモードとドキュメントモードには制限があります。
- 自動セグメンテーションを使用して並列移行するには、前述の制限があります。
- Amazon DocumentDB (MongoDB 互換) ソースは、変更データ キャプチャ (CDC) のスタート位置として特定のタイムスタンプの使用をサポートしていません。進行中のレプリケーション タスクは、タイムスタンプに関係なく変更のキャプチャをスタートします。
- DocumentDB (MongoDB 互換) をソースとして使用する場合、DMS は 1 秒あたり最大 250 レコードを処理できます。
- AWS DMS は、ネストレベルが 97 より大きいドキュメントをサポートしていません。
- ソースフィルターは、DocumentDB ではサポートされていません。
- AWS DMS は、Elastic クラスターモードのソースとしての DocumentDB の CDC (変更データキャプチャ) レプリケーションをサポートしていません。

ソースとしての Amazon DocumentDB のエンドポイントの設定

追加の接続属性の使用と同様、エンドポイントの設定を使用して、ソースの Amazon DocumentDB データベースを設定できます。ソースエンドポイントを作成するときはAWS CLI、 AWS DMS コンソールを使用するか、`--doc-db-settings '{"EndpointSetting": "value", ...}'`JSON 構文で の `create-endpoint` コマンドを使用して設定を指定します。

次の表は、ソースとして Amazon DocumentDB を使用できるエンドポイント設定を説明しています。

属性名	有効値	デフォルト値と説明

`NestingLevel`	`"none"`
	
	`"one"`
		`"none"` - ドキュメントモードを使用するよう `"none"` を指定します。テーブルモードを使用するよう `"one"` を指定します。

`ExtractDocID`	ブール値
	
	`true`
	
	`false`
		`false` - `NestingLevel` が `"none"` に設定されている場合、この属性を使用します。
		
		ターゲット データベースが Amazon DocumentDB の場合は、`'{"ExtractDocID": true}'` を設定します。

属性名	有効値	デフォルト値と説明
DocsToInvestigate	0 より大きい正の整数。	1000 - NestingLevel が "one" に設定されている場合、この属性を使用します。
ReplicateShardCollections	ブール値 true false	true の場合、 はデータをシャードコレクションに AWS DMS レプリケートします。ターゲットエンドポイントが DocumentDB エラスティッククラスターである AWS DMS 場合にのみ、この設定を使用します。 この設定が true の場合、次の点に注意する。 • TargetTablePrepMode を nothing に設定する必要がある。 • AWS DMS は自動的に useUpdateLookup を に設定しますfalse。

Amazon DocumentDB のソースデータ型

次の表に、 AWS DMSを使用する場合にサポートされる Amazon DocumentDB ソースデータ型を示します。この表では、 AWS DMS データ型からのデフォルトのマッピングも確認できます。データ型の詳細については、MongoDB のドキュメントの「BSON 型」をご参照ください。

ターゲットにマッピングされるデータ型を表示する方法については、使用しているターゲットエンドポイントのセクションをご参照ください。

AWS DMS データ型の詳細については、「」を参照してくださいAWS Database Migration Service のデータ型。

Amazon DocumentDB データ型	AWS DMS データ型
ブール値	Bool
バイナリ	BLOB
日付	日付
タイムスタンプ	日付

Amazon DocumentDB データ型	AWS DMS データ型
Int	INT4
Long	INT8
ダブル	REAL8
String (UTF-8)	CLOB
配列	CLOB
OID	文字列

のソースとしての Amazon S3 の使用 AWS DMS

を使用して Amazon S3 バケットからデータを移行できます AWS DMS。これを行うには、1 つ以上のデータファイルを含む Amazon S3 バケットへのアクセスを提供します。その S3 バケットに、データおよびそれらのファイルのデータのデータベーステーブル間のマッピングを記述する JSON ファイルを含めます。

全ロードのスタート前に、ソース データ ファイルが Amazon S3 バケットに存在している必要があります。バケット名は、`bucketName` パラメータを使用して指定します。

ソースデータファイルは、次の形式にすることができます。

- カンマ区切り値 (.csv)
- Parquet (DMS バージョン 3.5.3 以降) 。Parquet 形式のファイルの使用については、「」を参照してくださいのソースとしての Amazon S3 での Parquet 形式のファイルの使用 AWS DMS。

カンマ区切り値 (.csv) 形式のソースデータファイルの場合は、次の命名規則を使用して名前を付けます。この規則では、*`schemaName`* がソーススキーマで、*`tableName`* がそのスキーマ内のテーブルの名前です。

```
/schemaName/tableName/LOAD001.csv
/schemaName/tableName/LOAD002.csv
/schemaName/tableName/LOAD003.csv
...
```

たとえば、データファイルが次の Amazon S3 パスの mybucket にあるとします。

```
s3://mybucket/hr/employee
```

ロード時に、 はソーススキーマ名が でhr、ソーステーブル名が である AWS DMS と仮定しますemployee。

bucketName (必須) に加えて、オプションでbucketFolderパラメータを指定して、 AWS DMS が Amazon S3 バケット内のデータファイルを検索する場所を指定できます。前の例を続けて、bucketFolderを に設定するとsourcedata、 は次のパスでデータファイルを AWS DMS 読み取ります。

```
s3://mybucket/sourcedata/hr/employee
```

追加の接続属性を使用して、列の区切り文字、行の区切り文字、null 値のインジケータ、およびその他のパラメータを指定できます。詳細については、「のソースとしての Amazon S3 のエンドポイント設定 AWS DMS」を参照してください。

次のとおり、ExpectedBucketOwner Amazon S3 エンドポイント設定を使用してバケット所有者を指定して、スナイプ攻撃を防ぐことができます。その後、接続をテストしたり移行の実行をリクエストしたりすると、S3 は指定されたパラメータに対してバケット所有者のアカウント ID をチェックします。

```
--s3-settings='{"ExpectedBucketOwner": "AWS_Account_ID"}'
```

トピック

- のソースとしての Amazon S3 の外部テーブルの定義 AWS DMS
- Amazon S3 を AWS DMSのソースとして CDC の使用
- のソースとして Amazon S3 を使用する場合の前提条件 AWS DMS
- のソースとして Amazon S3 を使用する場合の制限 AWS DMS
- のソースとしての Amazon S3 のエンドポイント設定 AWS DMS
- Amazon S3 のソースデータ型
- のソースとしての Amazon S3 での Parquet 形式のファイルの使用 AWS DMS

のソースとしての Amazon S3 の外部テーブルの定義 AWS DMS

データファイルに加えて、外部テーブル定義も指定する必要があります。外部テーブル定義は、 が Amazon S3 AWS DMS からのデータを解釈する方法を説明する JSON ドキュメントです。 Amazon S3 このドキュメントの最大サイズは 2 MB です。 AWS DMS マネジメントコンソールを使用してソースエンドポイントを作成する場合は、JSON をテーブルマッピングボックスに直接入力できます。 AWS Command Line Interface （ AWS CLI) または AWS DMS API を使用して移行を実行する場合は、JSON ファイルを作成して外部テーブル定義を指定できます。

以下のものが含まれるデータファイルがあるとします。

```
101,Smith,Bob,2014-06-04,New York
102,Smith,Bob,2015-10-08,Los Angeles
103,Smith,Bob,2017-03-13,Dallas
104,Smith,Bob,2017-03-13,Dallas
```

このデータ用の外部テーブル定義の例を次に示します。

```
{
    "TableCount": "1",
    "Tables": [
        {
            "TableName": "employee",
            "TablePath": "hr/employee/",
            "TableOwner": "hr",
            "TableColumns": [
                {
                    "ColumnName": "Id",
                    "ColumnType": "INT8",
                    "ColumnNullable": "false",
                    "ColumnIsPk": "true"
                },
                {
                    "ColumnName": "LastName",
                    "ColumnType": "STRING",
                    "ColumnLength": "20"
                },
                {
                    "ColumnName": "FirstName",
                    "ColumnType": "STRING",
                    "ColumnLength": "30"
                },
```

```
                {
                    "ColumnName": "HireDate",
                    "ColumnType": "DATETIME"
                },
                {
                    "ColumnName": "OfficeLocation",
                    "ColumnType": "STRING",
                    "ColumnLength": "20"
                }
            ],
            "TableColumnsTotal": "5"
        }
    ]
}
```

この JSON ドキュメントの要素は次のとおりです。

TableCount - ソーステーブルの数。この例では、テーブルは 1 つしかありません。

Tables - ソーステーブルあたり 1 つの JSON マップで構成される配列。この例では、マップは 1 つしかありません。各マップは以下の要素で構成されています。

- TableName - ソーステーブルの名前。
- TablePath - AWS DMS が全データ ロード ファイルを見つけることができる Amazon S3 バケット内のパス。bucketFolder の値を指定した場合、この値がパスの先頭に付加されます。
- TableOwner - このテーブルのスキーマ名。
- TableColumns - 1 つ以上のマップの配列、それぞれがソーステーブルの列を記述します：
  - ColumnName - ソーステーブルの列の名前。
  - ColumnType - 列のデータ型。有効なデータ型については、「Amazon S3 のソースデータ型」をご参照ください。
  - ColumnLength - この列のバイト数。S3 ソースは FULL LOB モードをサポートしていないため、列の最大長は 2147483647 バイト (2,047 MegaBytes) に制限されています。ColumnLengthは、次のデータ型で有効です。
    - BYTE
    - STRING
  - ColumnNullable - この列に NULL 値を含めることができる場合、true であるブール値 (デフォルト = false)。

- `ColumnIsPk` - この列がプライマリ キーの一部である場合、`true` であるブール値 (デフォルト = `false`)。
- `ColumnDateFormat` – DATE 型、TIME 型、DATETIME 型の列の入力日付形式。データ文字列を日付オブジェクトに解析するために使用される。可能な値は以下のとおりです:

```
- YYYY-MM-dd HH:mm:ss
- YYYY-MM-dd HH:mm:ss.F
- YYYY/MM/dd HH:mm:ss
- YYYY/MM/dd HH:mm:ss.F
- MM/dd/YYYY HH:mm:ss
- MM/dd/YYYY HH:mm:ss.F
- YYYYMMdd HH:mm:ss
- YYYYMMdd HH:mm:ss.F
```

- `TableColumnsTotal` - 列の総数。この番号は、`TableColumns` 配列内の要素数と一致している必要があります。

特に指定しない場合、 `ColumnLength`は が 0 AWS DMS であると仮定します。

> **Note**
>
> サポートされているバージョンの では AWS DMS、S3 ソースデータに、列値の前の最初の列としてオプションのオペレーション`TableName`列を含めることもできます。このオペレーション列は、フルロード時にデータを S3 ターゲットエンドポイントに移行するために使用されるオペレーション (`INSERT`) を識別します。
> 存在する場合、この列の値は `INSERT` オペレーションキーワードの最初の文字 (`I`) です。指定されている場合、この列は通常、S3 ソースが以前の移行中に S3 ターゲットとして DMS によって作成されたことを示します。
> 3.4.2 以前の DMS バージョンでは、この列は以前の DMS 全ロードから作成された S3 ソースデータにはありませんでした。この列を S3 ターゲットデータに追加すると、S3 ターゲットに書き込まれるすべての行の形式を、全ロード中に書き込まれたか、CDC ロード中に書き込まれたかにかかわらず一貫させることができます。S3 ターゲットデータをフォーマットするオプションの詳細については、「移行済み S3 データでのソース DB オペレーションの表示」をご参照ください。

NUMERIC 型の列の場合は、精度とスケールを指定します。精度は数値の桁の合計数であり、スケールは小数点以下の桁数です。次に示すように、これには ColumnPrecision および ColumnScale 要素を使用します。

```
...
    {
        "ColumnName": "HourlyRate",
        "ColumnType": "NUMERIC",
        "ColumnPrecision": "5"
        "ColumnScale": "2"
    }
...
```

分数の秒を含むデータを持つ DATETIME 型の列の場合は、スケールを指定します。[Scale] (スケール) は小数秒の桁数で、0 ~ 9 の範囲で指定できます。次に示すように、これには ColumnScale 要素を使用します。

```
...
{
      "ColumnName": "HireDate",
      "ColumnType": "DATETIME",
      "ColumnScale": "3"
}
...
```

特に指定しない場合、 ColumnScaleはゼロ AWS DMS であると仮定し、小数秒を切り捨てます。

Amazon S3 を AWS DMSのソースとして CDC の使用

が全データロード AWS DMS を実行すると、オプションでデータ変更をターゲットエンドポイントにレプリケートできます。これを行うには、変更データキャプチャファイル (CDC ファイル) を Amazon S3 bucket.reads にアップロードし、アップロード時にこれらの CDC ファイルを AWS DMS 読み取り、変更をターゲットエンドポイントに適用します。

CDC ファイルは次のように名前が付けられます。

```
CDC00001.csv
CDC00002.csv
CDC00003.csv
...
```

> **Note**
>
> 変更データフォルダ内の CDC ファイルをレプリケートするには、辞書と同じ (順次) 順序で正常にアップロードします。たとえば、ファイル CDC00002.csv はファイル CDC00003.csv より前にアップロードします。そうしないと、CDC00002.csv はスキップされ、CDC00003.csv の後にロードするとレプリケートされません。一方、ファイル CDC00004.csv は、CDC00003.csv の後にロードされた場合に正常にレプリケートされます。

がファイルを検索 AWS DMS できる場所を指定するには、 cdcPathパラメータを指定します。前の例を続けると、cdcPath を *changedata* に設定した場合、 AWS DMS は次のパスで CDC ファイルを読み取ります。

```
s3://mybucket/changedata
```

cdcPath を *changedata* に、bucketFolder を *myFolder* に設定した場合、 AWS DMS は次のパスで CDC ファイルを読み取ります。

```
s3://mybucket/myFolder/changedata
```

CDC ファイル内のレコードは次のような形式になります。

- オペレーション - 実行する変更オペレーション: INSERT または I、UPDATE または U、DELETE または D。これらのキーワードと文字値では大文字と小文字は区別されません。

  > **Note**
  >
  > サポートされている AWS DMS バージョンでは、 AWS DMS は 2 つの方法で各ロードレコードに対して実行するオペレーションを識別 AWS DMS できます。 は、レコードのキーワード値 (などINSERT) またはキーワードの最初の文字 (など) からこれを実行できますI。以前のバージョンでは、 は完全なキーワード値からのみロードオペレーション AWS DMS を認識しました。
  > の以前のバージョンでは AWS DMS、CDC データを ログに記録するために完全なキーワード値が書き込まれていました。さらに以前のバージョンでは、キーワード [initial] のみを使用して、S3 ターゲットにオペレーション値を書き込みました。
  > 両方の形式を認識する AWS DMS と、オペレーション列の記述方法に関係なく、 はオペレーションを処理して S3 ソースデータを作成できます。このアプローチでは、後の移行

のソースとして S3 ターゲットデータを使用できます このアプローチでは、後の S3 ソースのオペレーション列に表示されるキーワードの初期値の形式を変更する必要はありません。

- テーブル名 - ソーステーブルの名前。
- スキーマ名 - ソーススキーマの名前。
- データ - 変更するデータを表す 1 つまたは複数の列。

employee という名前のテーブルの CDC ファイルの例を次に示します。

```
INSERT,employee,hr,101,Smith,Bob,2014-06-04,New York
UPDATE,employee,hr,101,Smith,Bob,2015-10-08,Los Angeles
UPDATE,employee,hr,101,Smith,Bob,2017-03-13,Dallas
DELETE,employee,hr,101,Smith,Bob,2017-03-13,Dallas
```

のソースとして Amazon S3 を使用する場合の前提条件 AWS DMS

Amazon S3 を のソースとして使用するには AWS DMS、ソース S3 バケットが、データを移行する DMS レプリケーションインスタンスと同じ AWS リージョンにある必要があります。また、移行に使用する AWS アカウントには、ソースバケットへの読み取りアクセスも必要です。

移行タスクの作成に使用されるユーザーアカウントに割り当てられた AWS Identity and Access Management (IAM) ロールには、次のアクセス許可のセットが必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::mybucket*/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::mybucket*"
            ]
        }
    ]
}
```

Amazon S3 バケットでバージョニングが有効になっている場合、移行タスクの作成に使用されるユーザーアカウントに割り当てられた AWS Identity and Access Management (IAM) ロールには、次のアクセス許可のセットが必要です。 Amazon S3

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "S3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::mybucket*/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::mybucket*"
            ]
        }
    ]
}
```

のソースとして Amazon S3 を使用する場合の制限 AWS DMS

Amazon S3 をソースとして使用する場合、次の制限が適用されます。

- S3 のバージョニングは有効にしません。S3 のバージョニングが必要な場合は、ライフサイクルポリシーを使用して古いバージョンを積極的に削除します。これを行わないと、S3 `list-object` コールのタイムアウトが原因でエンドポイント接続テストが失敗する可能性があります。S3 バケットのライフサイクルポリシーを作成するには、「ストレージのライフサイクルの管理」を参照してください。S3 オブジェクトのバージョンを削除するには、「バージョニングが有効なバケットからのオブジェクトバージョンの削除」を参照してください。
- VPC 対応 (ゲートウェイ VPC) S3 バケットはバージョン 3.4.7 以降でサポートされます。
- MySQL はtimeデータ型を に変換しますstring。MySQL でtimeデータ型値を表示するには、ターゲットテーブルの 列を として定義しstring、タスクのターゲットテーブル準備モード設定を 切り捨てに設定します。 MySQL
- AWS DMS は、 BYTEと BYTE データ型の両方のデータに内部的に BYTES データ型を使用します。
- S3 ソースエンドポイントは、DMS テーブルの再ロード機能をサポートしていません。
- AWS DMS はAmazon S3LOB モードをサポートしていません。

Amazon S3 で Parquet 形式のファイルをソースとして使用する場合は、次の制限が適用されます。

- 、、または の日付DDMMYYYYはMMYYYYDD、S3 Parquet ソースの日付パーティショニング機能ではサポートされていません。

のソースとしての Amazon S3 のエンドポイント設定 AWS DMS

追加の接続属性の使用と同様、エンドポイントの設定を使用して、ソースの Amazon S3 データベースを設定できます。ソースエンドポイントを作成するときはAWS CLI、 AWS DMS コンソールを使用するか、`--s3-settings '{"EndpointSetting": "value", ...}'`JSON 構文で の `create-endpoint` コマンドを使用して設定を指定します。

次の表は、ソースとして Amazon S3 を使用できるエンドポイント設定を説明しています。

オプション	説明
`BucketFolder`	(オプション) S3 バケットのフォルダ名。この属性を指定すると、ソースデータ ファイルおよび CDC ファイルがそれぞれパス `s3://myBucket/bucketFolder /schemaName /tableName /` と `s3://myBucket/bucketFolder /` から読み取られます。この属性が

オプション	説明
	指定されない場合、使用されるパスは *schemaName* /*tableName* / となります。 '{"BucketFolder": " *sourceData* "}'
BucketName	S3 バケットの名前。 '{"BucketName": " *myBucket*"}'
CdcPath	CDC ファイルの場所。タスクで変更データをキャプチャする場合、この属性が必須です。それ以外の場合はオプションです。CdcPath が存在する場合、 はこのパスから CDC ファイルを AWS DMS 読み取り、データ変更をターゲットエンドポイントにレプリケートします。詳細については、「Amazon S3 を AWS DMSのソースとして CDC の使用」を参照してください。 '{"CdcPath": " *changeData* "}'
CsvDelimiter	ソースファイル内の列を分離するために使用される区切り文字。デフォルトではカンマを使用します。以下に例を示します。 '{"CsvDelimiter": ","}'
CsvNullValue	ソースから読み取るときに null として AWS DMS 扱われるユーザー定義の文字列。デフォルトは空の文字列です。このパラメータを設定しない場合、 は空の文字列を null 値として AWS DMS 扱います。このパラメータを「\N」などの文字列に設定すると、 はこの文字列を null 値として AWS DMS 扱い、空の文字列を空の文字列値として扱います。
CsvRowDelimiter	ソースファイル内の行を分離するために使用される区切り文字。デフォルトでは、改行 (\n) です。 '{"CsvRowDelimiter": "\n"}'
DataFormat	Parquet 形式でデータを読み取るParquetには、この値を に設定します。 '{"DataFormat": "Parquet"}'

オプション	説明
IgnoreHeaderRows	この値を 1 に設定すると、 は .csv ファイルの最初の行ヘッダー AWS DMS を無視します。値が 1 の場合は機能が有効になり、値が 0 の場合は機能が無効になります。 デフォルトは 0 です。 '{"IgnoreHeaderRows": 1}'
Rfc4180	この値を true または y に設定するときは、先頭の二重引用符にはそれぞれ終了の二重引用符が続く必要があります。このフォーマットは、RFC 4180 に準拠しています。この値を false または n に設定すると、文字列リテラルはターゲットにそのままコピーされます。この場合、区切り文字 (行または列) はフィールドの末尾を表します。したがって、区切り文字は値の末尾を示すため、これを文字列の一部とすることはできません。 デフォルトは true です。 有効値: true、false、y、n '{"Rfc4180": false}'

Amazon S3 のソースデータ型

のソースとして Amazon S3 を使用するデータ移行では、Amazon S3 からデータ型に AWS DMS データをマッピング AWS DMS する必要があります。詳細については、「のソースとしての Amazon S3 の外部テーブルの定義 AWS DMS」を参照してください。

ターゲットにマッピングされるデータ型を表示する方法については、使用しているターゲットエンドポイントのセクションをご参照ください。

AWS DMS データ型の詳細については、「」を参照してくださいAWS Database Migration Service のデータ型。

ソースとして Amazon S3 では、次の AWS DMS データ型が使用されます。

- BYTE - ColumnLength が必要です。詳細については、「のソースとしての Amazon S3 の外部テーブルの定義 AWS DMS」を参照してください。

- DATE
- TIME
- DATETIME – 詳細と例については、「のソースとしての Amazon S3 の外部テーブルの定義 AWS DMS」の DATETIME 型の例を参照。
- INT1
- INT2
- INT4
- INT8
- NUMERIC – `ColumnPrecision`と が必要です`ColumnScale`。 は次の最大値 AWS DMS をサポートします。
  - ColumnPrecision: 38
  - ColumnScale: 31

  詳細と例については、「のソースとしての Amazon S3 の外部テーブルの定義 AWS DMS」の NUMERIC 型の例を参照。
- REAL4
- REAL8
- STRING - `ColumnLength` が必要です。詳細については、「のソースとしての Amazon S3 の外部テーブルの定義 AWS DMS」をご参照ください。
- UINT1
- UINT2
- UINT4
- UINT8
- BLOB
- CLOB
- BOOLEAN

のソースとしての Amazon S3 での Parquet 形式のファイルの使用 AWS DMS

AWS DMS バージョン 3.5.3 以降では、S3 バケット内の Parquet 形式のファイルをフルロードレプリケーションと CDC レプリケーションの両方のソースとして使用できます。

DMS は、データを S3 ターゲットエンドポイントに移行することで DMS が生成するソースとして Parquet 形式のファイルのみをサポートします。ファイル名はサポートされている形式である必要があります。そうしないと、DMS は移行にファイル名を含めません。

Parquet 形式のソースデータファイルは、次のフォルダと命名規則にある必要があります。

```
schema/table1/LOAD00001.parquet
schema/table2/LOAD00002.parquet
schema/table2/LOAD00003.parquet
```

Parquet 形式の CDC データのソースデータファイルの場合は、次のフォルダと命名規則を使用して名前を付けて保存します。

```
schema/table/20230405-094615814.parquet
schema/table/20230405-094615853.parquet
schema/table/20230405-094615922.parquet
```

Parquet 形式のファイルにアクセスするには、次のエンドポイント設定を行います。

- `DataFormat` を `Parquet` に設定します。
- `cdcPath` 設定は設定しないでください。Parquet 形式のファイルは、必ず指定されたスキーマ/テーブルフォルダに作成してください。

S3 エンドポイントの設定の詳細については、 AWS Database Migration Service API リファレンスのS3Settings」を参照してください。

Parquet 形式のファイルでサポートされるデータ型

AWS DMS は、Parquet 形式のファイルからデータを移行するときに、次のソースデータ型とターゲットデータ型をサポートします。移行する前に、ターゲットテーブルに正しいデータ型の列があることを確認してください。

出典データ型	ターゲットデータ型
BYTE	BINARY
DATE	DATE32
TIME	TIME32

出典データ型	ターゲットデータ型
DATETIME	TIMESTAMP
INT1	INT8
INT2	INT16
INT4	INT32
INT8	INT64
NUMERIC	DECIMAL
REAL4	FLOAT
REAL8	DOUBLE
STRING	STRING
UINT1	UINT8
UINT2	UINT16
UINT4	UINT32
UINT8	UINT
WSTRING	STRING
BLOB	BINARY
NCLOB	STRING
CLOB	STRING
BOOLEAN	BOOL

Linux、Unix、Windows 用 IBM Db2、および Amazon RDS データベース (Db2 LUW) をソースとして使用しています AWS DMS

() を使用して、Linux、UNIX、Windows、Amazon RDS (Db2 LUW) 用の IBM Db2 データベースから、サポートされている任意のターゲットデータベースにデータを移行できます。 AWS Database Migration Service AWS DMS

Linux、UNIX、Windows、および RDS 上でソースとしてサポートされている Db2 のバージョンについては、を参照してください。 AWS DMS のソース AWS DMS

Secure Sockets Layer (SSL) を使用して、Db2 LUW エンドポイントとレプリケーションインスタンスとの接続を暗号化できます。Db2 LUW エンドポイントで SSL を使用する方法の詳細については、「での SSL の使用 AWS Database Migration Service」をご参照ください。

Db2 LUW をソースとして使用する場合の前提条件 AWS DMS

Db2 LUW データベースをソースとして使用する前に、次の前提条件が必要です。

変更データキャプチャ (CDC) とも呼ばれる継続的なレプリケーションを有効にするには、次を実行します。

- データベースを回復可能に設定する。そのためには変更を取り込む必要がある。 AWS DMS データベース設定パラメータの LOGARCHMETH1 と LOGARCHMETH2 のどちらかまたは両方が ON に設定されている場合、データベースは復元可能です。

  データベースが回復可能な場合は、必要に応じて AWS DMS Db2 にアクセスできます。ARCHIVE LOG
- DB2 のトランザクションログが使用可能で、処理に十分な保存期間があることを確認してください。 AWS DMS
- DB2 ではトランザクションログレコードを抽出するのに SYSADM 認証または DBADM 認証が必要です。ユーザーアカウントに次のアクセス権限を付与します。
  - SYSADM または DBADM
  - DATAACCESS

> **Note**
>
> フルロードのみのタスクの場合、DMS ユーザーアカウントには DATAACCESS アクセス権限が必要です。

- ソースとして IBM DB2 for LUW バージョン 9.7 を使用する場合は、追加接続属性 (ECA) を設定し、以下のように CurrentLSN に設定します：

  CurrentLSN=*LSN*、ここに *LSN* は、レプリケーションをスタートするログ シーケンス番号 (LSN) を指定します。または、CurrentLSN=*scan*。

Db2 LUW をソースとして使用する場合の制限事項 AWS DMS

AWS DMS クラスター化されたデータベースはサポートしていません。ただし、クラスターの各エンドポイントに個別の Db2 LUW を定義することができます。例えば、クラスター内のいずれかのノードでフルロードの移行タスクを作成して、各ノードから個別のタスクを作成できます。

AWS DMS ソース Db2 LUW BOOLEAN データベースのデータ型をサポートしていません。

継続的なレプリケーション (CDC) を使用する場合は、次の制限が適用されます。

- 複数のパーティションがあるテーブルを切り捨てると、 AWS DMS コンソールに表示される DDL イベントの数は、パーティションの数と同じになります。これは、Db2 LUW がパーティションごとに個別の DDL を記録するためです。
- 次の DDL アクションは、分割されたテーブルではサポート外です。
  - ALTER TABLE ADD PARTITION
  - ALTER TABLE DETACH PARTITION
  - ALTER TABLE ATTACH PARTITION
- AWS DMS DB2 高可用性ディザスタリカバリ (HADR) スタンバイインスタンスからの継続的なレプリケーション移行はサポートされていません。スタンバイはアクセスできません。
- DECFLOAT データ型はサポート外です。したがって、DECFLOAT 列への変更は、継続的なレプリケーション中は無視されます。
- RENAME COLUMN ステートメントはサポート外です。
- 多次元クラスタリング (MDC) テーブルを更新すると、 AWS DMS 更新されるたびにコンソールに INSERT + DELETE と表示されます。
- タスクの設定 [レプリケーションに LOB 列を含める] が有効でない場合、LOB 列を持つテーブルは、継続的なレプリケーション中に中断されます。
- Db2 LUW バージョン 10.5 以降では、データが格納されている可変長の文字列列は無視されます。 out-of-row この制限は、VARCHAR や VARGRAPHIC などのデータ型を持つ列の拡張行サイズを使用して作成されたテーブルにのみ適用されます。この制限を回避するには、テーブルをペー

ジサイズの大きいテーブルスペースに移動します。詳細については、「What can I do if I want to change the pagesize of DB2 tablespaces」を参照してください。

- 継続的レプリケーションの場合、DMS は DB2 LOAD ユーティリティによってページレベルでロードされたデータの移行をサポートしていません。代わりに、SQL 挿入を使用する IMPORT ユーティリティを使用します。詳細については、「インポートユーティリティとロードユーティリティの違い」をご参照ください。
- レプリケーションタスクの実行中、DMS が CREATE TABLE DDL をキャプチャするのは、テーブルが DATA CAPTURE CHANGE 属性で作成された場合のみです。
- DMS が Db2 データベースパーティション機能 (DPF) を使用する場合、以下の制限があります。
  - DMS は DPF 環境の Db2 ノード間のトランザクションを調整できません。これは IBM DB2READLOG API インターフェース内の制約によるものです。DPF では、DB2 によるデータの分割方法によっては、トランザクションが複数の Db2 ノードにまたがる場合があります。そのため、DMS ソリューションは各 Db2 ノードから個別にトランザクションをキャプチャする必要があります。
  - DMS は、複数の DMS ソースエンドポイントでを設定することで、DPF クラスター内の各 Db2 connectNode 1 ノードからローカルトランザクションをキャプチャできます。この構成は、DB2 サーバー構成ファイルに定義されている論理ノード番号に対応しています。db2nodes.cfg
  - 個々の Db2 ノード上のローカルトランザクションは、より大きなグローバルトランザクションの一部である場合があります。DMS は、他の Db2 ノード上のトランザクションと連携することなく、各ローカルトランザクションをターゲット上で個別に適用します。このような独立した処理は、特に行がパーティション間で移動される場合に複雑になる可能性があります。
  - DMS が複数の Db2 ノードから複製される場合、DMS は各 Db2 ノードに独立して操作を適用するため、ターゲットでの操作の正しい順序は保証されません。各 Db2 ノードから独立してローカルトランザクションをキャプチャすることが、特定のユースケースで機能することを確認する必要があります。
  - DPF 環境から移行する場合は、まずキャッシュされたイベントなしで全ロードタスクを実行し、次に CDC のみのタスクを実行することをお勧めします。エンドポイント設定を使用して設定した Full Load 開始タイムスタンプまたは LRI (ログレコード識別子) から開始して、Db2 ノードごとに 1 つのタスクを実行することをお勧めします。StartFromContextレプリケーション開始点の決定方法については、IBM Support ドキュメントの「レプリケーション開始の LSN または LRI 値の検索」を参照してください。
- 継続的なレプリケーション (CDC) で、特定のタイムスタンプからレプリケーションを開始する場合は、StartFromContext 接続属性を必要なタイムスタンプに設定する必要があります。

- データベースソリューションのスケーリングに使用できる DB2 LUW の拡張機能である Db2 PureScale 機能は、DMS では現時点ではサポートしていません。
- AWS DMS Amazon RDS 用 Db2 をソースとして使用する場合、CDC はサポートされません。

Db2 LUW をソースとして使用する場合のエンドポイント設定 AWS DMS

追加の接続属性の使用と同様、エンドポイントの設定を使用して、ソースの Db2 LUW データベースを設定できます。ソースエンドポイントを作成するときに、 AWS DMS コンソールを使用するか、`--ibm-db2-settings '{"EndpointSetting": "value", ...}'` JSON `create-endpoint` AWS CLI構文のコマンドを使用して設定を指定します。

次の表は、ソースとして Db2 LUW を使用できるエンドポイント設定を説明しています。

名前	説明
`CurrentLSN`	継続的なレプリケーション (CDC) では、`CurrentLSN` を使用してレプリケーションを開始するログシーケンス番号 (LSN) を指定します。
`MaxKBytesPerRead`	読み取りあたりの最大バイト数。NUMBER 値です。デフォルトは 64 KB です。
`SetDataCaptureChanges`	継続的なレプリケーション (CDC) を BOOLEAN 値として有効にします。デフォルトは true です。
`StartFromContext`	継続的レプリケーション (CDC) の場合は、`StartFromContext` を使用して、レプリケーションのスタート位置からログの下限を指定します。`StartFromContext` は異なる形式の値を受け入れます。有効な値を次に示します。 • `timestamp` (UTC)。例: `'{"StartFromContext": "timestamp:2021-09-21T13:00:00"}'` • `NOW`

名前	説明

	IBM DB2 LUW バージョン 10.5 以降の場合、NOW と CurrentLSN: scan を組み合わせると、最新の LSO からタスクが開始される。例:
	
	`'{"CurrentLSN": "scan", "StartFromContext": "NOW"}'`
	
	• 特定のLRI。例:
	
	`'{"StartFromContext": "0100000000000022C C000000000004FB13"}'`
	
	ログファイルの LRI/LSN 範囲を特定するには、次の例に示すように db2flsn コマンドを実行します。
	
	`db2flsn -db SAMPLE -lrirange 2`
	
	この例の出力は次のようになります。
	
	`S0000002.LOG: has LRI range 000000000000000100 00000000002254000000000004F9A6 to 00000000000000010000000000022CC000000000004 FB13`
	
	その出力では、ログファイルは S0000002.LOG で、StartFromContextLRI 値は範囲の末尾の 34 バイトです。
	
	`0100000000000022CC000000000004FB13`

IBM Db2 LUW のソースデータ型

Db2 LUW をソースとして使用するデータ移行は、ほとんどの Db2 LUW AWS DMS データ型をサポートします。次の表は、使用時にサポートされる Db2 LUW AWS DMS ソースデータ型と、データ型からのデフォルトマッピングを示しています。 AWS DMS Db2 LUW データ型の詳細については、「Db2 LUW のドキュメント」をご参照ください。

ターゲットにマッピングされるデータ型を表示する方法については、使用しているターゲットエンドポイントのセクションをご参照ください。

AWS DMS データ型に関する追加情報については、を参照してください。AWS Database Migration Service のデータ型

Db2 LUW データ型	AWS DMS データタイプ
INTEGER	INT4
SMALLINT	INT2
BIGINT	INT8
DECIMAL (p,s)	NUMERIC (p,s)
FLOAT	REAL8
DOUBLE	REAL8
REAL	REAL4
DECFLOAT (p)	精度が 16 の場合は REAL8 で、精度が 34 の場合は STRING
GRAPHIC (n)	WSTRING、長さが 0 より大きく 127 以下の 2 バイト文字の固定長グラフィック文字列用
VARGRAPHIC (n)	WSTRING、長さが 0 より大きく 16,352 以下の 2 バイト文字の可変長グラフィック文字列用
LONG VARGRAPHIC (n)	CLOB、長さが 0 より大きく 16,352 以下の 2 バイト文字の可変長グラフィック文字列用

Db2 LUW データ型	AWS DMS データタイプ
CHARACTER (n)	STRING、長さが 0 より大きく 255 以下の 2 バイト文字の固定長文字列用
VARCHAR(n)	STRING、長さが 0 より大きく 32,704 以下の 2 バイト文字の可変長文字列用
LONG VARCHAR (n)	CLOB、長さが 0 より大きく 32,704 以下の 2 バイト文字の可変長文字列用
CHAR (n) FOR BIT DATA	BYTES
VARCHAR (n) FOR BIT DATA	BYTES
LONG VARCHAR FOR BIT DATA	BYTES
DATE	DATE
TIME	TIME
タイムスタンプ	DATETIME
BLOB (n)	BLOB 最大長は 2,147,483,647 バイト
CLOB (n)	CLOB 最大長は 2,147,483,647 バイト
DBCLOB (n)	CLOB 最大サイズは 1,073,741,824 の 2 バイト文字
XML	CLOB

AWS DMS のソースとしての IBM Db2 for z/OS データベースの使用

AWS Database Migration Service (AWS DMS) を使用して、IBM for z/OS データベースから、サポートされているすべてのターゲットデータベースにデータを移行できます。

AWS DMS がソースとしてサポートする Db2 for z/OS のバージョンについては、「のソース AWS DMS」を参照してください。

AWS DMS のソースとして Db2 for z/OS を使用する場合の前提条件

AWS DMS でソースとして Db2 for z/OS データベースを使用するには、ソースエンドポイント接続設定で指定された Db2 for z/OS ユーザーに次の権限を付与します。

```
GRANT SELECT ON SYSIBM.SYSTABLES TO Db2USER;
GRANT SELECT ON SYSIBM.SYSTABLESPACE TO Db2USER;
GRANT SELECT ON SYSIBM.SYSTABLEPART TO Db2USER;
GRANT SELECT ON SYSIBM.SYSCOLUMNS TO Db2USER;
GRANT SELECT ON SYSIBM.SYSDATABASE TO Db2USER;
GRANT SELECT ON SYSIBM.SYSDUMMY1 TO Db2USER
```

user defined ソーステーブルでの SELECT ON 権限も付与します。

AWS DMS での IBM Db2 for z/OS ソースエンドポイントは、IBM Data Server Driver for ODBC を基盤にデータにアクセスします。DMS がこのエンドポイントに接続するには、データベースサーバーに有効な IBM ODBC Connect ライセンスが必要です。

Db2 for z/OS を AWS DMS のソースとして使用する場合の制限

IBM Db2 for z/OS データベースを AWS DMS のソースとして使用する場合、以下の制限が適用されます。

- サポートされるのは、フルロードのレプリケーションタスクのみです。変更データキャプチャ (CDC) はサポートされません。
- 並列ロードはサポートされません。
- ビューのデータ検証はサポートされません。
- 列またはテーブルレベルの変換と行レベルの選択フィルターでは、テーブルマッピングでスキーマ、テーブル、列の名前を大文字で指定する必要があります。

ソースの IBM Db2 for z/OS のデータ型

Db2 for z/OS を AWS DMS のソースとして使用するデータ移行では、ほとんどの Db2 for z/OS データ型がサポートされます。次の表は、AWS DMS を使用する場合にサポートされるソースの Db2 for z/OS のデータ型と、AWS DMS のデータ型のデフォルトマッピングを説明しています。

Db2 for z/OS のデータ型の詳細については、「IBM Db2 for z/OS のドキュメント」を参照してください。

ターゲットにマップされるデータ型を確認する方法については、使用しているターゲットエンドポイントのセクションを参照してください。

AWS DMS のデータ型の詳細については、「AWS Database Migration Service のデータ型」を参照してください。

Db2 for z/OS のデータ型	AWS DMS のデータ型
INTEGER	INT4
SMALLINT	INT2
BIGINT	INT8
DECIMAL (p,s)	NUMERIC (p,s) DB2 の設定で小数点がカンマ (,) に設定されている場合は、DB2 設定をサポートするように Replicate を設定する。
FLOAT	REAL8
DOUBLE	REAL8
REAL	REAL4
DECFLOAT (p)	精度が 16 の場合は REAL8、精度が 34 の場合は、STRING
GRAPHIC (n)	n>=127 の場合は WSTRING、長さが 0 より大きく 127 以下の 2 バイト文字の固定長グラフィック文字列用
VARGRAPHIC (n)	WSTRING、長さが 0 より大きく 16,352 文字以下の 2 バイト文字の可変長グラフィック文字列向け

Db2 for z/OS のデータ型	AWS DMS のデータ型
LONG VARGRAPHIC (n)	CLOB、長さが 0 より大きく 16,352 文字以下の 2 バイト文字の可変長グラフィック文字列向け
CHARACTER (n)	STRING、長さが 0 より大きく 255 以下の 2 バイト文字の固定長文字列向け
VARCHAR (n)	STRING、長さが 0 より大きく 32,704 文字以下の 2 バイト文字の可変長グラフィック文字列向け
LONG VARCHAR (n)	CLOB、長さが 0 より大きく 32,704 文字以下の 2 バイト文字の可変長グラフィック文字列向け
CHAR (n) FOR BIT DATA	BYTES
VARCHAR (n) FOR BIT DATA	BYTES
LONG VARCHAR FOR BIT DATA	BYTES
DATE	DATE
TIME	TIME
TIMESTAMP	DATETIME
BLOB (n)	BLOB 最大長は 2,147,483,647 バイト
CLOB (n)	CLOB 最大長は 2,147,483,647 バイト
DBCLOB (n)	CLOB 最大長は 1,073,741,824 の 2 バイト文字

Db2 for z/OS のデータ型	AWS DMS のデータ型
XML	CLOB
BINARY	BYTES
VARBINARY	BYTES
ROWID	BYTES ROWID の使用方法の詳細については、次を参照。
TIMESTAMP WITH TIME ZONE	サポートされない。

タスクのターゲットテーブル準備モードが DROP_AND_CREATE (デフォルト) に設定されている場合、ROWID 列はデフォルトで移行されます。特定のデータベースやテーブル以外では意味をなさないため、このような列はデータ検証では無視されます。このような列の移行を無効にするには、次のいずれかの準備ステップで実行できます。

- このような列を含まないターゲットテーブルを事前に作成します。次に、タスクのターゲットテーブル準備モードを DO_NOTHING または TRUNCATE_BEFORE_LOAD のいずれかに設定します。AWS Schema Conversion Tool (AWS SCT) を使用して、この列を含まないターゲットテーブルを事前に作成できます。
- このような列を除外して無視するテーブルマッピングルールをタスクに追加します。詳細については、「 変換ルールおよび変換アクション」を参照してください。

AWS Mainframe Modernization サービスの PostgreSQL での EBCDIC 照合順序

AWS Mainframe Modernization プログラムは、メインフレームアプリケーションをクラウドネイティブな AWS 管理のランタイム環境にモダナイズするうえでのサポートを提供します。このサービスでは、移行とモダナイゼーションの計画と実装に役立つツールとリソースを提供しています。メインフレームのモダナイゼーションと移行の詳細については、「Mainframe Modernization with AWS」を参照してください。

IBM Db2 for z/OS のデータセットによっては、Extended Binary Coded Decimal Interchange (EBCDIC) 文字セットでエンコードされています。これは ASCII (米国情報交換標準コード) が一般的に使用されるようになる以前に開発された文字セットです。コードページを使用して、テキストの各文字を文字セット内の文字にマップします。従来のコードページには、コードポイントと文字 ID の間のマッピング情報が含まれています。文字 ID は 8 バイトの文字データ文字列です。コードポイン

トは文字を表す 8 ビットの 2 進数です。コードポイントは通常、2 進値の 16 進数表現として示されます。

Mainframe Modernization サービスの Micro Focus または BluAge コンポーネントを現在使用している場合は、AWS DMS に 特定のコードポイントをシフト (変換) するように指示する必要があります。AWS DMS タスク設定を使用してこのシフトを実行できます。以下の例は、AWS DMS の `CharacterSetSettings` 操作を使用して DMS タスク設定のシフトをマッピングする方法を説明しています。

```
"CharacterSetSettings": {
        "CharacterSetSupport": null,
        "CharacterReplacements": [
{"SourceCharacterCodePoint": "0000","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "00B8","TargetCharacterCodePoint": "0160"}
,{"SourceCharacterCodePoint": "00BC","TargetCharacterCodePoint": "0161"}
,{"SourceCharacterCodePoint": "00BD","TargetCharacterCodePoint": "017D"}
,{"SourceCharacterCodePoint": "00BE","TargetCharacterCodePoint": "017E"}
,{"SourceCharacterCodePoint": "00A8","TargetCharacterCodePoint": "0152"}
,{"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0153"}
,{"SourceCharacterCodePoint": "00A6","TargetCharacterCodePoint": "0178"}
            }
        ]
    }
```

必要なシフトを認識するためのいくつかの EBCDIC 照合順序が PostgreSQL 向けに既に提供されています。複数の異なるコードページがサポートされています。次のセクションでは、サポートされているすべてのコードページで変更する必要がある点の JSON サンプルを提供しています。DMS タスクに必要となる JSON コードをコピーして貼り付けるだけです。

Micro Focus 専用 EBCDIC 照合順序

Micro Focus では、必要に応じて文字のサブセットを次の照合順序にシフトします。

```
da-DK-cp1142m-x-icu
de-DE-cp1141m-x-icu
en-GB-cp1146m-x-icu
en-US-cp1140m-x-icu
es-ES-cp1145m-x-icu
fi-FI-cp1143m-x-icu
fr-FR-cp1147m-x-icu
it-IT-cp1144m-x-icu
```

```
 nl-BE-cp1148m-x-icu
```

Example Micro Focus のデータは次のとおり照合ごとにシフトします。

en_us_cp1140m

コードシフト:

```
0000    0180
00A6    0160
00B8    0161
00BC    017D
00BD    017E
00BE    0152
00A8    0153
00B4    0178
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0000","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "00A6","TargetCharacterCodePoint": "0160"}
,{"SourceCharacterCodePoint": "00B8","TargetCharacterCodePoint": "0161"}
,{"SourceCharacterCodePoint": "00BC","TargetCharacterCodePoint": "017D"}
,{"SourceCharacterCodePoint": "00BD","TargetCharacterCodePoint": "017E"}
,{"SourceCharacterCodePoint": "00BE","TargetCharacterCodePoint": "0152"}
,{"SourceCharacterCodePoint": "00A8","TargetCharacterCodePoint": "0153"}
,{"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0178"}
```

en_us_cp1141m

コードシフト:

```
0000    0180
00B8    0160
00BC    0161
00BD    017D
00BE    017E
00A8    0152
00B4    0153
00A6    0178
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0000","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "00B8","TargetCharacterCodePoint": "0160"}
,{"SourceCharacterCodePoint": "00BC","TargetCharacterCodePoint": "0161"}
,{"SourceCharacterCodePoint": "00BD","TargetCharacterCodePoint": "017D"}
,{"SourceCharacterCodePoint": "00BE","TargetCharacterCodePoint": "017E"}
,{"SourceCharacterCodePoint": "00A8","TargetCharacterCodePoint": "0152"}
,{"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0153"}
,{"SourceCharacterCodePoint": "00A6","TargetCharacterCodePoint": "0178"}
```

en_us_cp1142m

コードシフト:

```
0000    0180
00A6    0160
00B8    0161
00BC    017D
00BD    017E
00BE    0152
00A8    0153
00B4    0178
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0000","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "00A6","TargetCharacterCodePoint": "0160"}
,{"SourceCharacterCodePoint": "00B8","TargetCharacterCodePoint": "0161"}
,{"SourceCharacterCodePoint": "00BC","TargetCharacterCodePoint": "017D"}
,{"SourceCharacterCodePoint": "00BD","TargetCharacterCodePoint": "017E"}
,{"SourceCharacterCodePoint": "00BE","TargetCharacterCodePoint": "0152"}
,{"SourceCharacterCodePoint": "00A8","TargetCharacterCodePoint": "0153"}
,{"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0178"}
```

en_us_cp1143m

コードシフト:

```
0000    0180
00B8    0160
00BC    0161
00BD    017D
00BE    017E
00A8    0152
00B4    0153
00A6    0178
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0000","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "00B8","TargetCharacterCodePoint": "0160"}
,{"SourceCharacterCodePoint": "00BC","TargetCharacterCodePoint": "0161"}
,{"SourceCharacterCodePoint": "00BD","TargetCharacterCodePoint": "017D"}
,{"SourceCharacterCodePoint": "00BE","TargetCharacterCodePoint": "017E"}
,{"SourceCharacterCodePoint": "00A8","TargetCharacterCodePoint": "0152"}
,{"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0153"}
,{"SourceCharacterCodePoint": "00A6","TargetCharacterCodePoint": "0178"}
```

en_us_cp1144m

コードシフト:

```
0000    0180
00B8    0160
00BC    0161
00BD    017D
00BE    017E
00A8    0152
00B4    0153
00A6    0178
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0000","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "00B8","TargetCharacterCodePoint": "0160"}
,{"SourceCharacterCodePoint": "00BC","TargetCharacterCodePoint": "0161"}
,{"SourceCharacterCodePoint": "00BD","TargetCharacterCodePoint": "017D"}
```

```
,{"SourceCharacterCodePoint": "00BE","TargetCharacterCodePoint": "017E"}
,{"SourceCharacterCodePoint": "00A8","TargetCharacterCodePoint": "0152"}
,{"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0153"}
,{"SourceCharacterCodePoint": "00A6","TargetCharacterCodePoint": "0178"}
```

en_us_cp1145m

コードシフト:

```
0000    0180
00A6    0160
00B8    0161
00A8    017D
00BC    017E
00BD    0152
00BE    0153
00B4    0178
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0000","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "00A6","TargetCharacterCodePoint": "0160"}
,{"SourceCharacterCodePoint": "00B8","TargetCharacterCodePoint": "0161"}
,{"SourceCharacterCodePoint": "00A8","TargetCharacterCodePoint": "017D"}
,{"SourceCharacterCodePoint": "00BC","TargetCharacterCodePoint": "017E"}
,{"SourceCharacterCodePoint": "00BD","TargetCharacterCodePoint": "0152"}
,{"SourceCharacterCodePoint": "00BE","TargetCharacterCodePoint": "0153"}
,{"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0178"}
```

en_us_cp1146m

コードシフト:

```
0000    0180
00A6    0160
00B8    0161
00BC    017D
00BD    017E
```

```
00BE    0152
00A8    0153
00B4    0178
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0000","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "00A6","TargetCharacterCodePoint": "0160"}
,{"SourceCharacterCodePoint": "00B8","TargetCharacterCodePoint": "0161"}
,{"SourceCharacterCodePoint": "00BC","TargetCharacterCodePoint": "017D"}
,{"SourceCharacterCodePoint": "00BD","TargetCharacterCodePoint": "017E"}
,{"SourceCharacterCodePoint": "00BE","TargetCharacterCodePoint": "0152"}
,{"SourceCharacterCodePoint": "00A8","TargetCharacterCodePoint": "0153"}
,{"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0178"}
```

en_us_cp1147m

コードシフト:

```
0000    0180
00B8    0160
00A8    0161
00BC    017D
00BD    017E
00BE    0152
00B4    0153
00A6    0178
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0000","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "00B8","TargetCharacterCodePoint": "0160"}
,{"SourceCharacterCodePoint": "00A8","TargetCharacterCodePoint": "0161"}
,{"SourceCharacterCodePoint": "00BC","TargetCharacterCodePoint": "017D"}
,{"SourceCharacterCodePoint": "00BD","TargetCharacterCodePoint": "017E"}
,{"SourceCharacterCodePoint": "00BE","TargetCharacterCodePoint": "0152"}
,{"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0153"}
,{"SourceCharacterCodePoint": "00A6","TargetCharacterCodePoint": "0178"}
```

en_us_cp1148m

コードシフト:

```
0000    0180
00A6    0160
00B8    0161
00BC    017D
00BD    017E
00BE    0152
00A8    0153
00B4    0178
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0000","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "00A6","TargetCharacterCodePoint": "0160"}
,{"SourceCharacterCodePoint": "00B8","TargetCharacterCodePoint": "0161"}
,{"SourceCharacterCodePoint": "00BC","TargetCharacterCodePoint": "017D"}
,{"SourceCharacterCodePoint": "00BD","TargetCharacterCodePoint": "017E"}
,{"SourceCharacterCodePoint": "00BE","TargetCharacterCodePoint": "0152"}
,{"SourceCharacterCodePoint": "00A8","TargetCharacterCodePoint": "0153"}
,{"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0178"}
```

BluAge 専用 EBCDIC 照合順序

BluAge の場合、必要に応じて低い値と高い値をすべてシフトします。この照合順序の使用は、Mainframe Migration の BluAge サービスをサポートする目的に限定されます。

```
da-DK-cp1142b-x-icu
 da-DK-cp277b-x-icu
 de-DE-cp1141b-x-icu
 de-DE-cp273b-x-icu
 en-GB-cp1146b-x-icu
 en-GB-cp285b-x-icu
 en-US-cp037b-x-icu
 en-US-cp1140b-x-icu
 es-ES-cp1145b-x-icu
```

```
es-ES-cp284b-x-icu
fi-FI-cp1143b-x-icu
fi-FI-cp278b-x-icu
fr-FR-cp1147b-x-icu
fr-FR-cp297b-x-icu
it-IT-cp1144b-x-icu
it-IT-cp280b-x-icu
nl-BE-cp1148b-x-icu
nl-BE-cp500b-x-icu
```

Example BluAge データシフト:

da-DK-cp277b と da-DK-cp1142b

コードシフト:

```
0180    0180
0001    0181
0002    0182
0003    0183
009C    0184
0009    0185
0086    0186
007F    0187
0097    0188
008D    0189
008E    018A
000B    018B
000C    018C
000D    018D
000E    018E
000F    018F
0010    0190
0011    0191
0012    0192
0013    0193
009D    0194
0085    0195
0008    0196
0087    0197
0018    0198
0019    0199
0092    019A
```

```
008F    019B
001C    019C
001D    019D
001E    019E
001F    019F
0080    01A0
0081    01A1
0082    01A2
0083    01A3
0084    01A4
000A    01A5
0017    01A6
001B    01A7
0088    01A8
0089    01A9
008A    01AA
008B    01AB
008C    01AC
0005    01AD
0006    01AE
0007    01AF
0090    01B0
0091    01B1
0016    01B2
0093    01B3
0094    01B4
0095    01B5
0096    01B6
0004    01B7
0098    01B8
0099    01B9
009A    01BA
009B    01BB
0014    01BC
0015    01BD
009E    01BE
001A    01BF
009F    027F
```

対応する AWS DMS タスク用の入力マッピング:

```
{"SourceCharacterCodePoint": "0180","TargetCharacterCodePoint": "0180"}
```

```
,{"SourceCharacterCodePoint": "0001","TargetCharacterCodePoint": "0181"}
,{"SourceCharacterCodePoint": "0002","TargetCharacterCodePoint": "0182"}
,{"SourceCharacterCodePoint": "0003","TargetCharacterCodePoint": "0183"}
,{"SourceCharacterCodePoint": "009C","TargetCharacterCodePoint": "0184"}
,{"SourceCharacterCodePoint": "0009","TargetCharacterCodePoint": "0185"}
,{"SourceCharacterCodePoint": "0086","TargetCharacterCodePoint": "0186"}
,{"SourceCharacterCodePoint": "007F","TargetCharacterCodePoint": "0187"}
,{"SourceCharacterCodePoint": "0097","TargetCharacterCodePoint": "0188"}
,{"SourceCharacterCodePoint": "008D","TargetCharacterCodePoint": "0189"}
,{"SourceCharacterCodePoint": "008E","TargetCharacterCodePoint": "018A"}
,{"SourceCharacterCodePoint": "000B","TargetCharacterCodePoint": "018B"}
,{"SourceCharacterCodePoint": "000C","TargetCharacterCodePoint": "018C"}
,{"SourceCharacterCodePoint": "000D","TargetCharacterCodePoint": "018D"}
,{"SourceCharacterCodePoint": "000E","TargetCharacterCodePoint": "018E"}
,{"SourceCharacterCodePoint": "000F","TargetCharacterCodePoint": "018F"}
,{"SourceCharacterCodePoint": "0010","TargetCharacterCodePoint": "0190"}
,{"SourceCharacterCodePoint": "0011","TargetCharacterCodePoint": "0191"}
,{"SourceCharacterCodePoint": "0012","TargetCharacterCodePoint": "0192"}
,{"SourceCharacterCodePoint": "0013","TargetCharacterCodePoint": "0193"}
,{"SourceCharacterCodePoint": "009D","TargetCharacterCodePoint": "0194"}
,{"SourceCharacterCodePoint": "0085","TargetCharacterCodePoint": "0195"}
,{"SourceCharacterCodePoint": "0008","TargetCharacterCodePoint": "0196"}
,{"SourceCharacterCodePoint": "0087","TargetCharacterCodePoint": "0197"}
,{"SourceCharacterCodePoint": "0018","TargetCharacterCodePoint": "0198"}
,{"SourceCharacterCodePoint": "0019","TargetCharacterCodePoint": "0199"}
,{"SourceCharacterCodePoint": "0092","TargetCharacterCodePoint": "019A"}
,{"SourceCharacterCodePoint": "008F","TargetCharacterCodePoint": "019B"}
,{"SourceCharacterCodePoint": "001C","TargetCharacterCodePoint": "019C"}
,{"SourceCharacterCodePoint": "001D","TargetCharacterCodePoint": "019D"}
,{"SourceCharacterCodePoint": "001E","TargetCharacterCodePoint": "019E"}
,{"SourceCharacterCodePoint": "001F","TargetCharacterCodePoint": "019F"}
,{"SourceCharacterCodePoint": "0080","TargetCharacterCodePoint": "01A0"}
,{"SourceCharacterCodePoint": "0081","TargetCharacterCodePoint": "01A1"}
,{"SourceCharacterCodePoint": "0082","TargetCharacterCodePoint": "01A2"}
,{"SourceCharacterCodePoint": "0083","TargetCharacterCodePoint": "01A3"}
,{"SourceCharacterCodePoint": "0084","TargetCharacterCodePoint": "01A4"}
,{"SourceCharacterCodePoint": "000A","TargetCharacterCodePoint": "01A5"}
,{"SourceCharacterCodePoint": "0017","TargetCharacterCodePoint": "01A6"}
,{"SourceCharacterCodePoint": "001B","TargetCharacterCodePoint": "01A7"}
,{"SourceCharacterCodePoint": "0088","TargetCharacterCodePoint": "01A8"}
,{"SourceCharacterCodePoint": "0089","TargetCharacterCodePoint": "01A9"}
,{"SourceCharacterCodePoint": "008A","TargetCharacterCodePoint": "01AA"}
,{"SourceCharacterCodePoint": "008B","TargetCharacterCodePoint": "01AB"}
,{"SourceCharacterCodePoint": "008C","TargetCharacterCodePoint": "01AC"}
```

```
,{"SourceCharacterCodePoint": "0005","TargetCharacterCodePoint": "01AD"}
,{"SourceCharacterCodePoint": "0006","TargetCharacterCodePoint": "01AE"}
,{"SourceCharacterCodePoint": "0007","TargetCharacterCodePoint": "01AF"}
,{"SourceCharacterCodePoint": "0090","TargetCharacterCodePoint": "01B0"}
,{"SourceCharacterCodePoint": "0091","TargetCharacterCodePoint": "01B1"}
,{"SourceCharacterCodePoint": "0016","TargetCharacterCodePoint": "01B2"}
,{"SourceCharacterCodePoint": "0093","TargetCharacterCodePoint": "01B3"}
,{"SourceCharacterCodePoint": "0094","TargetCharacterCodePoint": "01B4"}
,{"SourceCharacterCodePoint": "0095","TargetCharacterCodePoint": "01B5"}
,{"SourceCharacterCodePoint": "0096","TargetCharacterCodePoint": "01B6"}
,{"SourceCharacterCodePoint": "0004","TargetCharacterCodePoint": "01B7"}
,{"SourceCharacterCodePoint": "0098","TargetCharacterCodePoint": "01B8"}
,{"SourceCharacterCodePoint": "0099","TargetCharacterCodePoint": "01B9"}
,{"SourceCharacterCodePoint": "009A","TargetCharacterCodePoint": "01BA"}
,{"SourceCharacterCodePoint": "009B","TargetCharacterCodePoint": "01BB"}
,{"SourceCharacterCodePoint": "0014","TargetCharacterCodePoint": "01BC"}
,{"SourceCharacterCodePoint": "0015","TargetCharacterCodePoint": "01BD"}
,{"SourceCharacterCodePoint": "009E","TargetCharacterCodePoint": "01BE"}
,{"SourceCharacterCodePoint": "001A","TargetCharacterCodePoint": "01BF"}
,{"SourceCharacterCodePoint": "009F","TargetCharacterCodePoint": "027F"}
```

de-DE-273b と de-DE-1141b

コードシフト:

```
0180    0180
0001    0181
0002    0182
0003    0183
009C    0184
0009    0185
0086    0186
007F    0187
0097    0188
008D    0189
008E    018A
000B    018B
000C    018C
000D    018D
000E    018E
000F    018F
0010    0190
```

```
0011    0191
0012    0192
0013    0193
009D    0194
0085    0195
0008    0196
0087    0197
0018    0198
0019    0199
0092    019A
008F    019B
001C    019C
001D    019D
001E    019E
001F    019F
0080    01A0
0081    01A1
0082    01A2
0083    01A3
0084    01A4
000A    01A5
0017    01A6
001B    01A7
0088    01A8
0089    01A9
008A    01AA
008B    01AB
008C    01AC
0005    01AD
0006    01AE
0007    01AF
0090    01B0
0091    01B1
0016    01B2
0093    01B3
0094    01B4
0095    01B5
0096    01B6
0004    01B7
0098    01B8
0099    01B9
009A    01BA
009B    01BB
0014    01BC
```

```
0015    01BD
009E    01BE
001A    01BF
009F    027F
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0180","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "0001","TargetCharacterCodePoint": "0181"}
,{"SourceCharacterCodePoint": "0002","TargetCharacterCodePoint": "0182"}
,{"SourceCharacterCodePoint": "0003","TargetCharacterCodePoint": "0183"}
,{"SourceCharacterCodePoint": "009C","TargetCharacterCodePoint": "0184"}
,{"SourceCharacterCodePoint": "0009","TargetCharacterCodePoint": "0185"}
,{"SourceCharacterCodePoint": "0086","TargetCharacterCodePoint": "0186"}
,{"SourceCharacterCodePoint": "007F","TargetCharacterCodePoint": "0187"}
,{"SourceCharacterCodePoint": "0097","TargetCharacterCodePoint": "0188"}
,{"SourceCharacterCodePoint": "008D","TargetCharacterCodePoint": "0189"}
,{"SourceCharacterCodePoint": "008E","TargetCharacterCodePoint": "018A"}
,{"SourceCharacterCodePoint": "000B","TargetCharacterCodePoint": "018B"}
,{"SourceCharacterCodePoint": "000C","TargetCharacterCodePoint": "018C"}
,{"SourceCharacterCodePoint": "000D","TargetCharacterCodePoint": "018D"}
,{"SourceCharacterCodePoint": "000E","TargetCharacterCodePoint": "018E"}
,{"SourceCharacterCodePoint": "000F","TargetCharacterCodePoint": "018F"}
,{"SourceCharacterCodePoint": "0010","TargetCharacterCodePoint": "0190"}
,{"SourceCharacterCodePoint": "0011","TargetCharacterCodePoint": "0191"}
,{"SourceCharacterCodePoint": "0012","TargetCharacterCodePoint": "0192"}
,{"SourceCharacterCodePoint": "0013","TargetCharacterCodePoint": "0193"}
,{"SourceCharacterCodePoint": "009D","TargetCharacterCodePoint": "0194"}
,{"SourceCharacterCodePoint": "0085","TargetCharacterCodePoint": "0195"}
,{"SourceCharacterCodePoint": "0008","TargetCharacterCodePoint": "0196"}
,{"SourceCharacterCodePoint": "0087","TargetCharacterCodePoint": "0197"}
,{"SourceCharacterCodePoint": "0018","TargetCharacterCodePoint": "0198"}
,{"SourceCharacterCodePoint": "0019","TargetCharacterCodePoint": "0199"}
,{"SourceCharacterCodePoint": "0092","TargetCharacterCodePoint": "019A"}
,{"SourceCharacterCodePoint": "008F","TargetCharacterCodePoint": "019B"}
,{"SourceCharacterCodePoint": "001C","TargetCharacterCodePoint": "019C"}
,{"SourceCharacterCodePoint": "001D","TargetCharacterCodePoint": "019D"}
,{"SourceCharacterCodePoint": "001E","TargetCharacterCodePoint": "019E"}
,{"SourceCharacterCodePoint": "001F","TargetCharacterCodePoint": "019F"}
,{"SourceCharacterCodePoint": "0080","TargetCharacterCodePoint": "01A0"}
,{"SourceCharacterCodePoint": "0081","TargetCharacterCodePoint": "01A1"}
,{"SourceCharacterCodePoint": "0082","TargetCharacterCodePoint": "01A2"}
```

```
,{"SourceCharacterCodePoint": "0083","TargetCharacterCodePoint": "01A3"}
,{"SourceCharacterCodePoint": "0084","TargetCharacterCodePoint": "01A4"}
,{"SourceCharacterCodePoint": "000A","TargetCharacterCodePoint": "01A5"}
,{"SourceCharacterCodePoint": "0017","TargetCharacterCodePoint": "01A6"}
,{"SourceCharacterCodePoint": "001B","TargetCharacterCodePoint": "01A7"}
,{"SourceCharacterCodePoint": "0088","TargetCharacterCodePoint": "01A8"}
,{"SourceCharacterCodePoint": "0089","TargetCharacterCodePoint": "01A9"}
,{"SourceCharacterCodePoint": "008A","TargetCharacterCodePoint": "01AA"}
,{"SourceCharacterCodePoint": "008B","TargetCharacterCodePoint": "01AB"}
,{"SourceCharacterCodePoint": "008C","TargetCharacterCodePoint": "01AC"}
,{"SourceCharacterCodePoint": "0005","TargetCharacterCodePoint": "01AD"}
,{"SourceCharacterCodePoint": "0006","TargetCharacterCodePoint": "01AE"}
,{"SourceCharacterCodePoint": "0007","TargetCharacterCodePoint": "01AF"}
,{"SourceCharacterCodePoint": "0090","TargetCharacterCodePoint": "01B0"}
,{"SourceCharacterCodePoint": "0091","TargetCharacterCodePoint": "01B1"}
,{"SourceCharacterCodePoint": "0016","TargetCharacterCodePoint": "01B2"}
,{"SourceCharacterCodePoint": "0093","TargetCharacterCodePoint": "01B3"}
,{"SourceCharacterCodePoint": "0094","TargetCharacterCodePoint": "01B4"}
,{"SourceCharacterCodePoint": "0095","TargetCharacterCodePoint": "01B5"}
,{"SourceCharacterCodePoint": "0096","TargetCharacterCodePoint": "01B6"}
,{"SourceCharacterCodePoint": "0004","TargetCharacterCodePoint": "01B7"}
,{"SourceCharacterCodePoint": "0098","TargetCharacterCodePoint": "01B8"}
,{"SourceCharacterCodePoint": "0099","TargetCharacterCodePoint": "01B9"}
,{"SourceCharacterCodePoint": "009A","TargetCharacterCodePoint": "01BA"}
,{"SourceCharacterCodePoint": "009B","TargetCharacterCodePoint": "01BB"}
,{"SourceCharacterCodePoint": "0014","TargetCharacterCodePoint": "01BC"}
,{"SourceCharacterCodePoint": "0015","TargetCharacterCodePoint": "01BD"}
,{"SourceCharacterCodePoint": "009E","TargetCharacterCodePoint": "01BE"}
,{"SourceCharacterCodePoint": "001A","TargetCharacterCodePoint": "01BF"}
,{"SourceCharacterCodePoint": "009F","TargetCharacterCodePoint": "027F"}
```

en-GB-285b と en-GB-1146b

コードシフト:

```
0180    0180
0001    0181
0002    0182
0003    0183
009C    0184
0009    0185
0086    0186
```

```
007F    0187
0097    0188
008D    0189
008E    018A
000B    018B
000C    018C
000D    018D
000E    018E
000F    018F
0010    0190
0011    0191
0012    0192
0013    0193
009D    0194
0085    0195
0008    0196
0087    0197
0018    0198
0019    0199
0092    019A
008F    019B
001C    019C
001D    019D
001E    019E
001F    019F
0080    01A0
0081    01A1
0082    01A2
0083    01A3
0084    01A4
000A    01A5
0017    01A6
001B    01A7
0088    01A8
0089    01A9
008A    01AA
008B    01AB
008C    01AC
0005    01AD
0006    01AE
0007    01AF
0090    01B0
0091    01B1
0016    01B2
```

```
0093    01B3
0094    01B4
0095    01B5
0096    01B6
0004    01B7
0098    01B8
0099    01B9
009A    01BA
009B    01BB
0014    01BC
0015    01BD
009E    01BE
001A    01BF
009F    027F
```

対応する AWS DMS タスク用の入力マッピング:

```
{"SourceCharacterCodePoint": "0180","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "0001","TargetCharacterCodePoint": "0181"}
,{"SourceCharacterCodePoint": "0002","TargetCharacterCodePoint": "0182"}
,{"SourceCharacterCodePoint": "0003","TargetCharacterCodePoint": "0183"}
,{"SourceCharacterCodePoint": "009C","TargetCharacterCodePoint": "0184"}
,{"SourceCharacterCodePoint": "0009","TargetCharacterCodePoint": "0185"}
,{"SourceCharacterCodePoint": "0086","TargetCharacterCodePoint": "0186"}
,{"SourceCharacterCodePoint": "007F","TargetCharacterCodePoint": "0187"}
,{"SourceCharacterCodePoint": "0097","TargetCharacterCodePoint": "0188"}
,{"SourceCharacterCodePoint": "008D","TargetCharacterCodePoint": "0189"}
,{"SourceCharacterCodePoint": "008E","TargetCharacterCodePoint": "018A"}
,{"SourceCharacterCodePoint": "000B","TargetCharacterCodePoint": "018B"}
,{"SourceCharacterCodePoint": "000C","TargetCharacterCodePoint": "018C"}
,{"SourceCharacterCodePoint": "000D","TargetCharacterCodePoint": "018D"}
,{"SourceCharacterCodePoint": "000E","TargetCharacterCodePoint": "018E"}
,{"SourceCharacterCodePoint": "000F","TargetCharacterCodePoint": "018F"}
,{"SourceCharacterCodePoint": "0010","TargetCharacterCodePoint": "0190"}
,{"SourceCharacterCodePoint": "0011","TargetCharacterCodePoint": "0191"}
,{"SourceCharacterCodePoint": "0012","TargetCharacterCodePoint": "0192"}
,{"SourceCharacterCodePoint": "0013","TargetCharacterCodePoint": "0193"}
,{"SourceCharacterCodePoint": "009D","TargetCharacterCodePoint": "0194"}
,{"SourceCharacterCodePoint": "0085","TargetCharacterCodePoint": "0195"}
,{"SourceCharacterCodePoint": "0008","TargetCharacterCodePoint": "0196"}
,{"SourceCharacterCodePoint": "0087","TargetCharacterCodePoint": "0197"}
,{"SourceCharacterCodePoint": "0018","TargetCharacterCodePoint": "0198"}
```

```
,{"SourceCharacterCodePoint": "0019","TargetCharacterCodePoint": "0199"}
,{"SourceCharacterCodePoint": "0092","TargetCharacterCodePoint": "019A"}
,{"SourceCharacterCodePoint": "008F","TargetCharacterCodePoint": "019B"}
,{"SourceCharacterCodePoint": "001C","TargetCharacterCodePoint": "019C"}
,{"SourceCharacterCodePoint": "001D","TargetCharacterCodePoint": "019D"}
,{"SourceCharacterCodePoint": "001E","TargetCharacterCodePoint": "019E"}
,{"SourceCharacterCodePoint": "001F","TargetCharacterCodePoint": "019F"}
,{"SourceCharacterCodePoint": "0080","TargetCharacterCodePoint": "01A0"}
,{"SourceCharacterCodePoint": "0081","TargetCharacterCodePoint": "01A1"}
,{"SourceCharacterCodePoint": "0082","TargetCharacterCodePoint": "01A2"}
,{"SourceCharacterCodePoint": "0083","TargetCharacterCodePoint": "01A3"}
,{"SourceCharacterCodePoint": "0084","TargetCharacterCodePoint": "01A4"}
,{"SourceCharacterCodePoint": "000A","TargetCharacterCodePoint": "01A5"}
,{"SourceCharacterCodePoint": "0017","TargetCharacterCodePoint": "01A6"}
,{"SourceCharacterCodePoint": "001B","TargetCharacterCodePoint": "01A7"}
,{"SourceCharacterCodePoint": "0088","TargetCharacterCodePoint": "01A8"}
,{"SourceCharacterCodePoint": "0089","TargetCharacterCodePoint": "01A9"}
,{"SourceCharacterCodePoint": "008A","TargetCharacterCodePoint": "01AA"}
,{"SourceCharacterCodePoint": "008B","TargetCharacterCodePoint": "01AB"}
,{"SourceCharacterCodePoint": "008C","TargetCharacterCodePoint": "01AC"}
,{"SourceCharacterCodePoint": "0005","TargetCharacterCodePoint": "01AD"}
,{"SourceCharacterCodePoint": "0006","TargetCharacterCodePoint": "01AE"}
,{"SourceCharacterCodePoint": "0007","TargetCharacterCodePoint": "01AF"}
,{"SourceCharacterCodePoint": "0090","TargetCharacterCodePoint": "01B0"}
,{"SourceCharacterCodePoint": "0091","TargetCharacterCodePoint": "01B1"}
,{"SourceCharacterCodePoint": "0016","TargetCharacterCodePoint": "01B2"}
,{"SourceCharacterCodePoint": "0093","TargetCharacterCodePoint": "01B3"}
,{"SourceCharacterCodePoint": "0094","TargetCharacterCodePoint": "01B4"}
,{"SourceCharacterCodePoint": "0095","TargetCharacterCodePoint": "01B5"}
,{"SourceCharacterCodePoint": "0096","TargetCharacterCodePoint": "01B6"}
,{"SourceCharacterCodePoint": "0004","TargetCharacterCodePoint": "01B7"}
,{"SourceCharacterCodePoint": "0098","TargetCharacterCodePoint": "01B8"}
,{"SourceCharacterCodePoint": "0099","TargetCharacterCodePoint": "01B9"}
,{"SourceCharacterCodePoint": "009A","TargetCharacterCodePoint": "01BA"}
,{"SourceCharacterCodePoint": "009B","TargetCharacterCodePoint": "01BB"}
,{"SourceCharacterCodePoint": "0014","TargetCharacterCodePoint": "01BC"}
,{"SourceCharacterCodePoint": "0015","TargetCharacterCodePoint": "01BD"}
,{"SourceCharacterCodePoint": "009E","TargetCharacterCodePoint": "01BE"}
,{"SourceCharacterCodePoint": "001A","TargetCharacterCodePoint": "01BF"}
,{"SourceCharacterCodePoint": "009F","TargetCharacterCodePoint": "027F"}
```

en-us-037b と en-us-1140b

コードシフト:

```
0180    0180
0001    0181
0002    0182
0003    0183
009C    0184
0009    0185
0086    0186
007F    0187
0097    0188
008D    0189
008E    018A
000B    018B
000C    018C
000D    018D
000E    018E
000F    018F
0010    0190
0011    0191
0012    0192
0013    0193
009D    0194
0085    0195
0008    0196
0087    0197
0018    0198
0019    0199
0092    019A
008F    019B
001C    019C
001D    019D
001E    019E
001F    019F
0080    01A0
0081    01A1
0082    01A2
0083    01A3
0084    01A4
000A    01A5
0017    01A6
001B    01A7
```

```
0088    01A8
0089    01A9
008A    01AA
008B    01AB
008C    01AC
0005    01AD
0006    01AE
0007    01AF
0090    01B0
0091    01B1
0016    01B2
0093    01B3
0094    01B4
0095    01B5
0096    01B6
0004    01B7
0098    01B8
0099    01B9
009A    01BA
009B    01BB
0014    01BC
0015    01BD
009E    01BE
001A    01BF
009F    027F
```

対応する AWS DMS タスク用の入力マッピング:

```
{"SourceCharacterCodePoint": "0180","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "0001","TargetCharacterCodePoint": "0181"}
,{"SourceCharacterCodePoint": "0002","TargetCharacterCodePoint": "0182"}
,{"SourceCharacterCodePoint": "0003","TargetCharacterCodePoint": "0183"}
,{"SourceCharacterCodePoint": "009C","TargetCharacterCodePoint": "0184"}
,{"SourceCharacterCodePoint": "0009","TargetCharacterCodePoint": "0185"}
,{"SourceCharacterCodePoint": "0086","TargetCharacterCodePoint": "0186"}
,{"SourceCharacterCodePoint": "007F","TargetCharacterCodePoint": "0187"}
,{"SourceCharacterCodePoint": "0097","TargetCharacterCodePoint": "0188"}
,{"SourceCharacterCodePoint": "008D","TargetCharacterCodePoint": "0189"}
,{"SourceCharacterCodePoint": "008E","TargetCharacterCodePoint": "018A"}
,{"SourceCharacterCodePoint": "000B","TargetCharacterCodePoint": "018B"}
,{"SourceCharacterCodePoint": "000C","TargetCharacterCodePoint": "018C"}
,{"SourceCharacterCodePoint": "000D","TargetCharacterCodePoint": "018D"}
```

```
,{"SourceCharacterCodePoint": "000E","TargetCharacterCodePoint": "018E"}
,{"SourceCharacterCodePoint": "000F","TargetCharacterCodePoint": "018F"}
,{"SourceCharacterCodePoint": "0010","TargetCharacterCodePoint": "0190"}
,{"SourceCharacterCodePoint": "0011","TargetCharacterCodePoint": "0191"}
,{"SourceCharacterCodePoint": "0012","TargetCharacterCodePoint": "0192"}
,{"SourceCharacterCodePoint": "0013","TargetCharacterCodePoint": "0193"}
,{"SourceCharacterCodePoint": "009D","TargetCharacterCodePoint": "0194"}
,{"SourceCharacterCodePoint": "0085","TargetCharacterCodePoint": "0195"}
,{"SourceCharacterCodePoint": "0008","TargetCharacterCodePoint": "0196"}
,{"SourceCharacterCodePoint": "0087","TargetCharacterCodePoint": "0197"}
,{"SourceCharacterCodePoint": "0018","TargetCharacterCodePoint": "0198"}
,{"SourceCharacterCodePoint": "0019","TargetCharacterCodePoint": "0199"}
,{"SourceCharacterCodePoint": "0092","TargetCharacterCodePoint": "019A"}
,{"SourceCharacterCodePoint": "008F","TargetCharacterCodePoint": "019B"}
,{"SourceCharacterCodePoint": "001C","TargetCharacterCodePoint": "019C"}
,{"SourceCharacterCodePoint": "001D","TargetCharacterCodePoint": "019D"}
,{"SourceCharacterCodePoint": "001E","TargetCharacterCodePoint": "019E"}
,{"SourceCharacterCodePoint": "001F","TargetCharacterCodePoint": "019F"}
,{"SourceCharacterCodePoint": "0080","TargetCharacterCodePoint": "01A0"}
,{"SourceCharacterCodePoint": "0081","TargetCharacterCodePoint": "01A1"}
,{"SourceCharacterCodePoint": "0082","TargetCharacterCodePoint": "01A2"}
,{"SourceCharacterCodePoint": "0083","TargetCharacterCodePoint": "01A3"}
,{"SourceCharacterCodePoint": "0084","TargetCharacterCodePoint": "01A4"}
,{"SourceCharacterCodePoint": "000A","TargetCharacterCodePoint": "01A5"}
,{"SourceCharacterCodePoint": "0017","TargetCharacterCodePoint": "01A6"}
,{"SourceCharacterCodePoint": "001B","TargetCharacterCodePoint": "01A7"}
,{"SourceCharacterCodePoint": "0088","TargetCharacterCodePoint": "01A8"}
,{"SourceCharacterCodePoint": "0089","TargetCharacterCodePoint": "01A9"}
,{"SourceCharacterCodePoint": "008A","TargetCharacterCodePoint": "01AA"}
,{"SourceCharacterCodePoint": "008B","TargetCharacterCodePoint": "01AB"}
,{"SourceCharacterCodePoint": "008C","TargetCharacterCodePoint": "01AC"}
,{"SourceCharacterCodePoint": "0005","TargetCharacterCodePoint": "01AD"}
,{"SourceCharacterCodePoint": "0006","TargetCharacterCodePoint": "01AE"}
,{"SourceCharacterCodePoint": "0007","TargetCharacterCodePoint": "01AF"}
,{"SourceCharacterCodePoint": "0090","TargetCharacterCodePoint": "01B0"}
,{"SourceCharacterCodePoint": "0091","TargetCharacterCodePoint": "01B1"}
,{"SourceCharacterCodePoint": "0016","TargetCharacterCodePoint": "01B2"}
,{"SourceCharacterCodePoint": "0093","TargetCharacterCodePoint": "01B3"}
,{"SourceCharacterCodePoint": "0094","TargetCharacterCodePoint": "01B4"}
,{"SourceCharacterCodePoint": "0095","TargetCharacterCodePoint": "01B5"}
,{"SourceCharacterCodePoint": "0096","TargetCharacterCodePoint": "01B6"}
,{"SourceCharacterCodePoint": "0004","TargetCharacterCodePoint": "01B7"}
,{"SourceCharacterCodePoint": "0098","TargetCharacterCodePoint": "01B8"}
,{"SourceCharacterCodePoint": "0099","TargetCharacterCodePoint": "01B9"}
```

```
,{"SourceCharacterCodePoint": "009A","TargetCharacterCodePoint": "01BA"}
,{"SourceCharacterCodePoint": "009B","TargetCharacterCodePoint": "01BB"}
,{"SourceCharacterCodePoint": "0014","TargetCharacterCodePoint": "01BC"}
,{"SourceCharacterCodePoint": "0015","TargetCharacterCodePoint": "01BD"}
,{"SourceCharacterCodePoint": "009E","TargetCharacterCodePoint": "01BE"}
,{"SourceCharacterCodePoint": "001A","TargetCharacterCodePoint": "01BF"}
,{"SourceCharacterCodePoint": "009F","TargetCharacterCodePoint": "027F"}
```

es-ES-284b と es-ES-1145b

コードシフト:

```
0180    0180
0001    0181
0002    0182
0003    0183
009C    0184
0009    0185
0086    0186
007F    0187
0097    0188
008D    0189
008E    018A
000B    018B
000C    018C
000D    018D
000E    018E
000F    018F
0010    0190
0011    0191
0012    0192
0013    0193
009D    0194
0085    0195
0008    0196
0087    0197
0018    0198
0019    0199
0092    019A
008F    019B
001C    019C
001D    019D
```

```
001E    019E
001F    019F
0080    01A0
0081    01A1
0082    01A2
0083    01A3
0084    01A4
000A    01A5
0017    01A6
001B    01A7
0088    01A8
0089    01A9
008A    01AA
008B    01AB
008C    01AC
0005    01AD
0006    01AE
0007    01AF
0090    01B0
0091    01B1
0016    01B2
0093    01B3
0094    01B4
0095    01B5
0096    01B6
0004    01B7
0098    01B8
0099    01B9
009A    01BA
009B    01BB
0014    01BC
0015    01BD
009E    01BE
001A    01BF
009F    027F
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0180","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "0001","TargetCharacterCodePoint": "0181"}
,{"SourceCharacterCodePoint": "0002","TargetCharacterCodePoint": "0182"}
,{"SourceCharacterCodePoint": "0003","TargetCharacterCodePoint": "0183"}
```

```
,{"SourceCharacterCodePoint": "009C","TargetCharacterCodePoint": "0184"}
,{"SourceCharacterCodePoint": "0009","TargetCharacterCodePoint": "0185"}
,{"SourceCharacterCodePoint": "0086","TargetCharacterCodePoint": "0186"}
,{"SourceCharacterCodePoint": "007F","TargetCharacterCodePoint": "0187"}
,{"SourceCharacterCodePoint": "0097","TargetCharacterCodePoint": "0188"}
,{"SourceCharacterCodePoint": "008D","TargetCharacterCodePoint": "0189"}
,{"SourceCharacterCodePoint": "008E","TargetCharacterCodePoint": "018A"}
,{"SourceCharacterCodePoint": "000B","TargetCharacterCodePoint": "018B"}
,{"SourceCharacterCodePoint": "000C","TargetCharacterCodePoint": "018C"}
,{"SourceCharacterCodePoint": "000D","TargetCharacterCodePoint": "018D"}
,{"SourceCharacterCodePoint": "000E","TargetCharacterCodePoint": "018E"}
,{"SourceCharacterCodePoint": "000F","TargetCharacterCodePoint": "018F"}
,{"SourceCharacterCodePoint": "0010","TargetCharacterCodePoint": "0190"}
,{"SourceCharacterCodePoint": "0011","TargetCharacterCodePoint": "0191"}
,{"SourceCharacterCodePoint": "0012","TargetCharacterCodePoint": "0192"}
,{"SourceCharacterCodePoint": "0013","TargetCharacterCodePoint": "0193"}
,{"SourceCharacterCodePoint": "009D","TargetCharacterCodePoint": "0194"}
,{"SourceCharacterCodePoint": "0085","TargetCharacterCodePoint": "0195"}
,{"SourceCharacterCodePoint": "0008","TargetCharacterCodePoint": "0196"}
,{"SourceCharacterCodePoint": "0087","TargetCharacterCodePoint": "0197"}
,{"SourceCharacterCodePoint": "0018","TargetCharacterCodePoint": "0198"}
,{"SourceCharacterCodePoint": "0019","TargetCharacterCodePoint": "0199"}
,{"SourceCharacterCodePoint": "0092","TargetCharacterCodePoint": "019A"}
,{"SourceCharacterCodePoint": "008F","TargetCharacterCodePoint": "019B"}
,{"SourceCharacterCodePoint": "001C","TargetCharacterCodePoint": "019C"}
,{"SourceCharacterCodePoint": "001D","TargetCharacterCodePoint": "019D"}
,{"SourceCharacterCodePoint": "001E","TargetCharacterCodePoint": "019E"}
,{"SourceCharacterCodePoint": "001F","TargetCharacterCodePoint": "019F"}
,{"SourceCharacterCodePoint": "0080","TargetCharacterCodePoint": "01A0"}
,{"SourceCharacterCodePoint": "0081","TargetCharacterCodePoint": "01A1"}
,{"SourceCharacterCodePoint": "0082","TargetCharacterCodePoint": "01A2"}
,{"SourceCharacterCodePoint": "0083","TargetCharacterCodePoint": "01A3"}
,{"SourceCharacterCodePoint": "0084","TargetCharacterCodePoint": "01A4"}
,{"SourceCharacterCodePoint": "000A","TargetCharacterCodePoint": "01A5"}
,{"SourceCharacterCodePoint": "0017","TargetCharacterCodePoint": "01A6"}
,{"SourceCharacterCodePoint": "001B","TargetCharacterCodePoint": "01A7"}
,{"SourceCharacterCodePoint": "0088","TargetCharacterCodePoint": "01A8"}
,{"SourceCharacterCodePoint": "0089","TargetCharacterCodePoint": "01A9"}
,{"SourceCharacterCodePoint": "008A","TargetCharacterCodePoint": "01AA"}
,{"SourceCharacterCodePoint": "008B","TargetCharacterCodePoint": "01AB"}
,{"SourceCharacterCodePoint": "008C","TargetCharacterCodePoint": "01AC"}
,{"SourceCharacterCodePoint": "0005","TargetCharacterCodePoint": "01AD"}
,{"SourceCharacterCodePoint": "0006","TargetCharacterCodePoint": "01AE"}
,{"SourceCharacterCodePoint": "0007","TargetCharacterCodePoint": "01AF"}
```

```
,{"SourceCharacterCodePoint": "0090","TargetCharacterCodePoint": "01B0"}
,{"SourceCharacterCodePoint": "0091","TargetCharacterCodePoint": "01B1"}
,{"SourceCharacterCodePoint": "0016","TargetCharacterCodePoint": "01B2"}
,{"SourceCharacterCodePoint": "0093","TargetCharacterCodePoint": "01B3"}
,{"SourceCharacterCodePoint": "0094","TargetCharacterCodePoint": "01B4"}
,{"SourceCharacterCodePoint": "0095","TargetCharacterCodePoint": "01B5"}
,{"SourceCharacterCodePoint": "0096","TargetCharacterCodePoint": "01B6"}
,{"SourceCharacterCodePoint": "0004","TargetCharacterCodePoint": "01B7"}
,{"SourceCharacterCodePoint": "0098","TargetCharacterCodePoint": "01B8"}
,{"SourceCharacterCodePoint": "0099","TargetCharacterCodePoint": "01B9"}
,{"SourceCharacterCodePoint": "009A","TargetCharacterCodePoint": "01BA"}
,{"SourceCharacterCodePoint": "009B","TargetCharacterCodePoint": "01BB"}
,{"SourceCharacterCodePoint": "0014","TargetCharacterCodePoint": "01BC"}
,{"SourceCharacterCodePoint": "0015","TargetCharacterCodePoint": "01BD"}
,{"SourceCharacterCodePoint": "009E","TargetCharacterCodePoint": "01BE"}
,{"SourceCharacterCodePoint": "001A","TargetCharacterCodePoint": "01BF"}
,{"SourceCharacterCodePoint": "009F","TargetCharacterCodePoint": "027F"}
```

fi_FI-278b と fi-FI-1143b

コードシフト:

```
0180    0180
0001    0181
0002    0182
0003    0183
009C    0184
0009    0185
0086    0186
007F    0187
0097    0188
008D    0189
008E    018A
000B    018B
000C    018C
000D    018D
000E    018E
000F    018F
0010    0190
0011    0191
0012    0192
0013    0193
```

```
009D    0194
0085    0195
0008    0196
0087    0197
0018    0198
0019    0199
0092    019A
008F    019B
001C    019C
001D    019D
001E    019E
001F    019F
0080    01A0
0081    01A1
0082    01A2
0083    01A3
0084    01A4
000A    01A5
0017    01A6
001B    01A7
0088    01A8
0089    01A9
008A    01AA
008B    01AB
008C    01AC
0005    01AD
0006    01AE
0007    01AF
0090    01B0
0091    01B1
0016    01B2
0093    01B3
0094    01B4
0095    01B5
0096    01B6
0004    01B7
0098    01B8
0099    01B9
009A    01BA
009B    01BB
0014    01BC
0015    01BD
009E    01BE
001A    01BF
```

```
009F    027F
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0180","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "0001","TargetCharacterCodePoint": "0181"}
,{"SourceCharacterCodePoint": "0002","TargetCharacterCodePoint": "0182"}
,{"SourceCharacterCodePoint": "0003","TargetCharacterCodePoint": "0183"}
,{"SourceCharacterCodePoint": "009C","TargetCharacterCodePoint": "0184"}
,{"SourceCharacterCodePoint": "0009","TargetCharacterCodePoint": "0185"}
,{"SourceCharacterCodePoint": "0086","TargetCharacterCodePoint": "0186"}
,{"SourceCharacterCodePoint": "007F","TargetCharacterCodePoint": "0187"}
,{"SourceCharacterCodePoint": "0097","TargetCharacterCodePoint": "0188"}
,{"SourceCharacterCodePoint": "008D","TargetCharacterCodePoint": "0189"}
,{"SourceCharacterCodePoint": "008E","TargetCharacterCodePoint": "018A"}
,{"SourceCharacterCodePoint": "000B","TargetCharacterCodePoint": "018B"}
,{"SourceCharacterCodePoint": "000C","TargetCharacterCodePoint": "018C"}
,{"SourceCharacterCodePoint": "000D","TargetCharacterCodePoint": "018D"}
,{"SourceCharacterCodePoint": "000E","TargetCharacterCodePoint": "018E"}
,{"SourceCharacterCodePoint": "000F","TargetCharacterCodePoint": "018F"}
,{"SourceCharacterCodePoint": "0010","TargetCharacterCodePoint": "0190"}
,{"SourceCharacterCodePoint": "0011","TargetCharacterCodePoint": "0191"}
,{"SourceCharacterCodePoint": "0012","TargetCharacterCodePoint": "0192"}
,{"SourceCharacterCodePoint": "0013","TargetCharacterCodePoint": "0193"}
,{"SourceCharacterCodePoint": "009D","TargetCharacterCodePoint": "0194"}
,{"SourceCharacterCodePoint": "0085","TargetCharacterCodePoint": "0195"}
,{"SourceCharacterCodePoint": "0008","TargetCharacterCodePoint": "0196"}
,{"SourceCharacterCodePoint": "0087","TargetCharacterCodePoint": "0197"}
,{"SourceCharacterCodePoint": "0018","TargetCharacterCodePoint": "0198"}
,{"SourceCharacterCodePoint": "0019","TargetCharacterCodePoint": "0199"}
,{"SourceCharacterCodePoint": "0092","TargetCharacterCodePoint": "019A"}
,{"SourceCharacterCodePoint": "008F","TargetCharacterCodePoint": "019B"}
,{"SourceCharacterCodePoint": "001C","TargetCharacterCodePoint": "019C"}
,{"SourceCharacterCodePoint": "001D","TargetCharacterCodePoint": "019D"}
,{"SourceCharacterCodePoint": "001E","TargetCharacterCodePoint": "019E"}
,{"SourceCharacterCodePoint": "001F","TargetCharacterCodePoint": "019F"}
,{"SourceCharacterCodePoint": "0080","TargetCharacterCodePoint": "01A0"}
,{"SourceCharacterCodePoint": "0081","TargetCharacterCodePoint": "01A1"}
,{"SourceCharacterCodePoint": "0082","TargetCharacterCodePoint": "01A2"}
,{"SourceCharacterCodePoint": "0083","TargetCharacterCodePoint": "01A3"}
,{"SourceCharacterCodePoint": "0084","TargetCharacterCodePoint": "01A4"}
,{"SourceCharacterCodePoint": "000A","TargetCharacterCodePoint": "01A5"}
```

```
,{"SourceCharacterCodePoint": "0017","TargetCharacterCodePoint": "01A6"}
,{"SourceCharacterCodePoint": "001B","TargetCharacterCodePoint": "01A7"}
,{"SourceCharacterCodePoint": "0088","TargetCharacterCodePoint": "01A8"}
,{"SourceCharacterCodePoint": "0089","TargetCharacterCodePoint": "01A9"}
,{"SourceCharacterCodePoint": "008A","TargetCharacterCodePoint": "01AA"}
,{"SourceCharacterCodePoint": "008B","TargetCharacterCodePoint": "01AB"}
,{"SourceCharacterCodePoint": "008C","TargetCharacterCodePoint": "01AC"}
,{"SourceCharacterCodePoint": "0005","TargetCharacterCodePoint": "01AD"}
,{"SourceCharacterCodePoint": "0006","TargetCharacterCodePoint": "01AE"}
,{"SourceCharacterCodePoint": "0007","TargetCharacterCodePoint": "01AF"}
,{"SourceCharacterCodePoint": "0090","TargetCharacterCodePoint": "01B0"}
,{"SourceCharacterCodePoint": "0091","TargetCharacterCodePoint": "01B1"}
,{"SourceCharacterCodePoint": "0016","TargetCharacterCodePoint": "01B2"}
,{"SourceCharacterCodePoint": "0093","TargetCharacterCodePoint": "01B3"}
,{"SourceCharacterCodePoint": "0094","TargetCharacterCodePoint": "01B4"}
,{"SourceCharacterCodePoint": "0095","TargetCharacterCodePoint": "01B5"}
,{"SourceCharacterCodePoint": "0096","TargetCharacterCodePoint": "01B6"}
,{"SourceCharacterCodePoint": "0004","TargetCharacterCodePoint": "01B7"}
,{"SourceCharacterCodePoint": "0098","TargetCharacterCodePoint": "01B8"}
,{"SourceCharacterCodePoint": "0099","TargetCharacterCodePoint": "01B9"}
,{"SourceCharacterCodePoint": "009A","TargetCharacterCodePoint": "01BA"}
,{"SourceCharacterCodePoint": "009B","TargetCharacterCodePoint": "01BB"}
,{"SourceCharacterCodePoint": "0014","TargetCharacterCodePoint": "01BC"}
,{"SourceCharacterCodePoint": "0015","TargetCharacterCodePoint": "01BD"}
,{"SourceCharacterCodePoint": "009E","TargetCharacterCodePoint": "01BE"}
,{"SourceCharacterCodePoint": "001A","TargetCharacterCodePoint": "01BF"}
,{"SourceCharacterCodePoint": "009F","TargetCharacterCodePoint": "027F"}
```

fr-FR-297b と fr-FR-1147b

コードシフト:

```
0180    0180
0001    0181
0002    0182
0003    0183
009C    0184
0009    0185
0086    0186
007F    0187
0097    0188
008D    0189
008E    018A
```

```
000B    018B
000C    018C
000D    018D
000E    018E
000F    018F
0010    0190
0011    0191
0012    0192
0013    0193
009D    0194
0085    0195
0008    0196
0087    0197
0018    0198
0019    0199
0092    019A
008F    019B
001C    019C
001D    019D
001E    019E
001F    019F
0080    01A0
0081    01A1
0082    01A2
0083    01A3
0084    01A4
000A    01A5
0017    01A6
001B    01A7
0088    01A8
0089    01A9
008A    01AA
008B    01AB
008C    01AC
0005    01AD
0006    01AE
0007    01AF
0090    01B0
0091    01B1
0016    01B2
0093    01B3
0094    01B4
0095    01B5
0096    01B6
```

```
0004    01B7
0098    01B8
0099    01B9
009A    01BA
009B    01BB
0014    01BC
0015    01BD
009E    01BE
001A    01BF
009F    027F
```

対応する AWS DMS タスク用の入力マッピング:

```
{"SourceCharacterCodePoint": "0180","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "0001","TargetCharacterCodePoint": "0181"}
,{"SourceCharacterCodePoint": "0002","TargetCharacterCodePoint": "0182"}
,{"SourceCharacterCodePoint": "0003","TargetCharacterCodePoint": "0183"}
,{"SourceCharacterCodePoint": "009C","TargetCharacterCodePoint": "0184"}
,{"SourceCharacterCodePoint": "0009","TargetCharacterCodePoint": "0185"}
,{"SourceCharacterCodePoint": "0086","TargetCharacterCodePoint": "0186"}
,{"SourceCharacterCodePoint": "007F","TargetCharacterCodePoint": "0187"}
,{"SourceCharacterCodePoint": "0097","TargetCharacterCodePoint": "0188"}
,{"SourceCharacterCodePoint": "008D","TargetCharacterCodePoint": "0189"}
,{"SourceCharacterCodePoint": "008E","TargetCharacterCodePoint": "018A"}
,{"SourceCharacterCodePoint": "000B","TargetCharacterCodePoint": "018B"}
,{"SourceCharacterCodePoint": "000C","TargetCharacterCodePoint": "018C"}
,{"SourceCharacterCodePoint": "000D","TargetCharacterCodePoint": "018D"}
,{"SourceCharacterCodePoint": "000E","TargetCharacterCodePoint": "018E"}
,{"SourceCharacterCodePoint": "000F","TargetCharacterCodePoint": "018F"}
,{"SourceCharacterCodePoint": "0010","TargetCharacterCodePoint": "0190"}
,{"SourceCharacterCodePoint": "0011","TargetCharacterCodePoint": "0191"}
,{"SourceCharacterCodePoint": "0012","TargetCharacterCodePoint": "0192"}
,{"SourceCharacterCodePoint": "0013","TargetCharacterCodePoint": "0193"}
,{"SourceCharacterCodePoint": "009D","TargetCharacterCodePoint": "0194"}
,{"SourceCharacterCodePoint": "0085","TargetCharacterCodePoint": "0195"}
,{"SourceCharacterCodePoint": "0008","TargetCharacterCodePoint": "0196"}
,{"SourceCharacterCodePoint": "0087","TargetCharacterCodePoint": "0197"}
,{"SourceCharacterCodePoint": "0018","TargetCharacterCodePoint": "0198"}
,{"SourceCharacterCodePoint": "0019","TargetCharacterCodePoint": "0199"}
,{"SourceCharacterCodePoint": "0092","TargetCharacterCodePoint": "019A"}
,{"SourceCharacterCodePoint": "008F","TargetCharacterCodePoint": "019B"}
,{"SourceCharacterCodePoint": "001C","TargetCharacterCodePoint": "019C"}
```

```
,{"SourceCharacterCodePoint": "001D","TargetCharacterCodePoint": "019D"}
,{"SourceCharacterCodePoint": "001E","TargetCharacterCodePoint": "019E"}
,{"SourceCharacterCodePoint": "001F","TargetCharacterCodePoint": "019F"}
,{"SourceCharacterCodePoint": "0080","TargetCharacterCodePoint": "01A0"}
,{"SourceCharacterCodePoint": "0081","TargetCharacterCodePoint": "01A1"}
,{"SourceCharacterCodePoint": "0082","TargetCharacterCodePoint": "01A2"}
,{"SourceCharacterCodePoint": "0083","TargetCharacterCodePoint": "01A3"}
,{"SourceCharacterCodePoint": "0084","TargetCharacterCodePoint": "01A4"}
,{"SourceCharacterCodePoint": "000A","TargetCharacterCodePoint": "01A5"}
,{"SourceCharacterCodePoint": "0017","TargetCharacterCodePoint": "01A6"}
,{"SourceCharacterCodePoint": "001B","TargetCharacterCodePoint": "01A7"}
,{"SourceCharacterCodePoint": "0088","TargetCharacterCodePoint": "01A8"}
,{"SourceCharacterCodePoint": "0089","TargetCharacterCodePoint": "01A9"}
,{"SourceCharacterCodePoint": "008A","TargetCharacterCodePoint": "01AA"}
,{"SourceCharacterCodePoint": "008B","TargetCharacterCodePoint": "01AB"}
,{"SourceCharacterCodePoint": "008C","TargetCharacterCodePoint": "01AC"}
,{"SourceCharacterCodePoint": "0005","TargetCharacterCodePoint": "01AD"}
,{"SourceCharacterCodePoint": "0006","TargetCharacterCodePoint": "01AE"}
,{"SourceCharacterCodePoint": "0007","TargetCharacterCodePoint": "01AF"}
,{"SourceCharacterCodePoint": "0090","TargetCharacterCodePoint": "01B0"}
,{"SourceCharacterCodePoint": "0091","TargetCharacterCodePoint": "01B1"}
,{"SourceCharacterCodePoint": "0016","TargetCharacterCodePoint": "01B2"}
,{"SourceCharacterCodePoint": "0093","TargetCharacterCodePoint": "01B3"}
,{"SourceCharacterCodePoint": "0094","TargetCharacterCodePoint": "01B4"}
,{"SourceCharacterCodePoint": "0095","TargetCharacterCodePoint": "01B5"}
,{"SourceCharacterCodePoint": "0096","TargetCharacterCodePoint": "01B6"}
,{"SourceCharacterCodePoint": "0004","TargetCharacterCodePoint": "01B7"}
,{"SourceCharacterCodePoint": "0098","TargetCharacterCodePoint": "01B8"}
,{"SourceCharacterCodePoint": "0099","TargetCharacterCodePoint": "01B9"}
,{"SourceCharacterCodePoint": "009A","TargetCharacterCodePoint": "01BA"}
,{"SourceCharacterCodePoint": "009B","TargetCharacterCodePoint": "01BB"}
,{"SourceCharacterCodePoint": "0014","TargetCharacterCodePoint": "01BC"}
,{"SourceCharacterCodePoint": "0015","TargetCharacterCodePoint": "01BD"}
,{"SourceCharacterCodePoint": "009E","TargetCharacterCodePoint": "01BE"}
,{"SourceCharacterCodePoint": "001A","TargetCharacterCodePoint": "01BF"}
,{"SourceCharacterCodePoint": "009F","TargetCharacterCodePoint": "027F"}
```

it-IT-280b と it-IT-1144b

コードシフト:

```
0180    0180
```

```
0001    0181
0002    0182
0003    0183
009C    0184
0009    0185
0086    0186
007F    0187
0097    0188
008D    0189
008E    018A
000B    018B
000C    018C
000D    018D
000E    018E
000F    018F
0010    0190
0011    0191
0012    0192
0013    0193
009D    0194
0085    0195
0008    0196
0087    0197
0018    0198
0019    0199
0092    019A
008F    019B
001C    019C
001D    019D
001E    019E
001F    019F
0080    01A0
0081    01A1
0082    01A2
0083    01A3
0084    01A4
000A    01A5
0017    01A6
001B    01A7
0088    01A8
0089    01A9
008A    01AA
008B    01AB
008C    01AC
```

```
0005    01AD
0006    01AE
0007    01AF
0090    01B0
0091    01B1
0016    01B2
0093    01B3
0094    01B4
0095    01B5
0096    01B6
0004    01B7
0098    01B8
0099    01B9
009A    01BA
009B    01BB
0014    01BC
0015    01BD
009E    01BE
001A    01BF
009F    027F
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0180","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "0001","TargetCharacterCodePoint": "0181"}
,{"SourceCharacterCodePoint": "0002","TargetCharacterCodePoint": "0182"}
,{"SourceCharacterCodePoint": "0003","TargetCharacterCodePoint": "0183"}
,{"SourceCharacterCodePoint": "009C","TargetCharacterCodePoint": "0184"}
,{"SourceCharacterCodePoint": "0009","TargetCharacterCodePoint": "0185"}
,{"SourceCharacterCodePoint": "0086","TargetCharacterCodePoint": "0186"}
,{"SourceCharacterCodePoint": "007F","TargetCharacterCodePoint": "0187"}
,{"SourceCharacterCodePoint": "0097","TargetCharacterCodePoint": "0188"}
,{"SourceCharacterCodePoint": "008D","TargetCharacterCodePoint": "0189"}
,{"SourceCharacterCodePoint": "008E","TargetCharacterCodePoint": "018A"}
,{"SourceCharacterCodePoint": "000B","TargetCharacterCodePoint": "018B"}
,{"SourceCharacterCodePoint": "000C","TargetCharacterCodePoint": "018C"}
,{"SourceCharacterCodePoint": "000D","TargetCharacterCodePoint": "018D"}
,{"SourceCharacterCodePoint": "000E","TargetCharacterCodePoint": "018E"}
,{"SourceCharacterCodePoint": "000F","TargetCharacterCodePoint": "018F"}
,{"SourceCharacterCodePoint": "0010","TargetCharacterCodePoint": "0190"}
,{"SourceCharacterCodePoint": "0011","TargetCharacterCodePoint": "0191"}
,{"SourceCharacterCodePoint": "0012","TargetCharacterCodePoint": "0192"}
```

```
,{"SourceCharacterCodePoint": "0013","TargetCharacterCodePoint": "0193"}
,{"SourceCharacterCodePoint": "009D","TargetCharacterCodePoint": "0194"}
,{"SourceCharacterCodePoint": "0085","TargetCharacterCodePoint": "0195"}
,{"SourceCharacterCodePoint": "0008","TargetCharacterCodePoint": "0196"}
,{"SourceCharacterCodePoint": "0087","TargetCharacterCodePoint": "0197"}
,{"SourceCharacterCodePoint": "0018","TargetCharacterCodePoint": "0198"}
,{"SourceCharacterCodePoint": "0019","TargetCharacterCodePoint": "0199"}
,{"SourceCharacterCodePoint": "0092","TargetCharacterCodePoint": "019A"}
,{"SourceCharacterCodePoint": "008F","TargetCharacterCodePoint": "019B"}
,{"SourceCharacterCodePoint": "001C","TargetCharacterCodePoint": "019C"}
,{"SourceCharacterCodePoint": "001D","TargetCharacterCodePoint": "019D"}
,{"SourceCharacterCodePoint": "001E","TargetCharacterCodePoint": "019E"}
,{"SourceCharacterCodePoint": "001F","TargetCharacterCodePoint": "019F"}
,{"SourceCharacterCodePoint": "0080","TargetCharacterCodePoint": "01A0"}
,{"SourceCharacterCodePoint": "0081","TargetCharacterCodePoint": "01A1"}
,{"SourceCharacterCodePoint": "0082","TargetCharacterCodePoint": "01A2"}
,{"SourceCharacterCodePoint": "0083","TargetCharacterCodePoint": "01A3"}
,{"SourceCharacterCodePoint": "0084","TargetCharacterCodePoint": "01A4"}
,{"SourceCharacterCodePoint": "000A","TargetCharacterCodePoint": "01A5"}
,{"SourceCharacterCodePoint": "0017","TargetCharacterCodePoint": "01A6"}
,{"SourceCharacterCodePoint": "001B","TargetCharacterCodePoint": "01A7"}
,{"SourceCharacterCodePoint": "0088","TargetCharacterCodePoint": "01A8"}
,{"SourceCharacterCodePoint": "0089","TargetCharacterCodePoint": "01A9"}
,{"SourceCharacterCodePoint": "008A","TargetCharacterCodePoint": "01AA"}
,{"SourceCharacterCodePoint": "008B","TargetCharacterCodePoint": "01AB"}
,{"SourceCharacterCodePoint": "008C","TargetCharacterCodePoint": "01AC"}
,{"SourceCharacterCodePoint": "0005","TargetCharacterCodePoint": "01AD"}
,{"SourceCharacterCodePoint": "0006","TargetCharacterCodePoint": "01AE"}
,{"SourceCharacterCodePoint": "0007","TargetCharacterCodePoint": "01AF"}
,{"SourceCharacterCodePoint": "0090","TargetCharacterCodePoint": "01B0"}
,{"SourceCharacterCodePoint": "0091","TargetCharacterCodePoint": "01B1"}
,{"SourceCharacterCodePoint": "0016","TargetCharacterCodePoint": "01B2"}
,{"SourceCharacterCodePoint": "0093","TargetCharacterCodePoint": "01B3"}
,{"SourceCharacterCodePoint": "0094","TargetCharacterCodePoint": "01B4"}
,{"SourceCharacterCodePoint": "0095","TargetCharacterCodePoint": "01B5"}
,{"SourceCharacterCodePoint": "0096","TargetCharacterCodePoint": "01B6"}
,{"SourceCharacterCodePoint": "0004","TargetCharacterCodePoint": "01B7"}
,{"SourceCharacterCodePoint": "0098","TargetCharacterCodePoint": "01B8"}
,{"SourceCharacterCodePoint": "0099","TargetCharacterCodePoint": "01B9"}
,{"SourceCharacterCodePoint": "009A","TargetCharacterCodePoint": "01BA"}
,{"SourceCharacterCodePoint": "009B","TargetCharacterCodePoint": "01BB"}
,{"SourceCharacterCodePoint": "0014","TargetCharacterCodePoint": "01BC"}
,{"SourceCharacterCodePoint": "0015","TargetCharacterCodePoint": "01BD"}
,{"SourceCharacterCodePoint": "009E","TargetCharacterCodePoint": "01BE"}
```

```
,{"SourceCharacterCodePoint": "001A","TargetCharacterCodePoint": "01BF"}
,{"SourceCharacterCodePoint": "009F","TargetCharacterCodePoint": "027F"}
```

nl-BE-500b と nl-BE-1148b

コードシフト:

```
0180    0180
0001    0181
0002    0182
0003    0183
009C    0184
0009    0185
0086    0186
007F    0187
0097    0188
008D    0189
008E    018A
000B    018B
000C    018C
000D    018D
000E    018E
000F    018F
0010    0190
0011    0191
0012    0192
0013    0193
009D    0194
0085    0195
0008    0196
0087    0197
0018    0198
0019    0199
0092    019A
008F    019B
001C    019C
001D    019D
001E    019E
001F    019F
0080    01A0
0081    01A1
0082    01A2
```

```
0083    01A3
0084    01A4
000A    01A5
0017    01A6
001B    01A7
0088    01A8
0089    01A9
008A    01AA
008B    01AB
008C    01AC
0005    01AD
0006    01AE
0007    01AF
0090    01B0
0091    01B1
0016    01B2
0093    01B3
0094    01B4
0095    01B5
0096    01B6
0004    01B7
0098    01B8
0099    01B9
009A    01BA
009B    01BB
0014    01BC
0015    01BD
009E    01BE
001A    01BF
009F    027F
```

対応する AWS DMS タスク用の入力マッピング:

```
 {"SourceCharacterCodePoint": "0180","TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "0001","TargetCharacterCodePoint": "0181"}
,{"SourceCharacterCodePoint": "0002","TargetCharacterCodePoint": "0182"}
,{"SourceCharacterCodePoint": "0003","TargetCharacterCodePoint": "0183"}
,{"SourceCharacterCodePoint": "009C","TargetCharacterCodePoint": "0184"}
,{"SourceCharacterCodePoint": "0009","TargetCharacterCodePoint": "0185"}
,{"SourceCharacterCodePoint": "0086","TargetCharacterCodePoint": "0186"}
,{"SourceCharacterCodePoint": "007F","TargetCharacterCodePoint": "0187"}
,{"SourceCharacterCodePoint": "0097","TargetCharacterCodePoint": "0188"}
```

```
,{"SourceCharacterCodePoint": "008D","TargetCharacterCodePoint": "0189"}
,{"SourceCharacterCodePoint": "008E","TargetCharacterCodePoint": "018A"}
,{"SourceCharacterCodePoint": "000B","TargetCharacterCodePoint": "018B"}
,{"SourceCharacterCodePoint": "000C","TargetCharacterCodePoint": "018C"}
,{"SourceCharacterCodePoint": "000D","TargetCharacterCodePoint": "018D"}
,{"SourceCharacterCodePoint": "000E","TargetCharacterCodePoint": "018E"}
,{"SourceCharacterCodePoint": "000F","TargetCharacterCodePoint": "018F"}
,{"SourceCharacterCodePoint": "0010","TargetCharacterCodePoint": "0190"}
,{"SourceCharacterCodePoint": "0011","TargetCharacterCodePoint": "0191"}
,{"SourceCharacterCodePoint": "0012","TargetCharacterCodePoint": "0192"}
,{"SourceCharacterCodePoint": "0013","TargetCharacterCodePoint": "0193"}
,{"SourceCharacterCodePoint": "009D","TargetCharacterCodePoint": "0194"}
,{"SourceCharacterCodePoint": "0085","TargetCharacterCodePoint": "0195"}
,{"SourceCharacterCodePoint": "0008","TargetCharacterCodePoint": "0196"}
,{"SourceCharacterCodePoint": "0087","TargetCharacterCodePoint": "0197"}
,{"SourceCharacterCodePoint": "0018","TargetCharacterCodePoint": "0198"}
,{"SourceCharacterCodePoint": "0019","TargetCharacterCodePoint": "0199"}
,{"SourceCharacterCodePoint": "0092","TargetCharacterCodePoint": "019A"}
,{"SourceCharacterCodePoint": "008F","TargetCharacterCodePoint": "019B"}
,{"SourceCharacterCodePoint": "001C","TargetCharacterCodePoint": "019C"}
,{"SourceCharacterCodePoint": "001D","TargetCharacterCodePoint": "019D"}
,{"SourceCharacterCodePoint": "001E","TargetCharacterCodePoint": "019E"}
,{"SourceCharacterCodePoint": "001F","TargetCharacterCodePoint": "019F"}
,{"SourceCharacterCodePoint": "0080","TargetCharacterCodePoint": "01A0"}
,{"SourceCharacterCodePoint": "0081","TargetCharacterCodePoint": "01A1"}
,{"SourceCharacterCodePoint": "0082","TargetCharacterCodePoint": "01A2"}
,{"SourceCharacterCodePoint": "0083","TargetCharacterCodePoint": "01A3"}
,{"SourceCharacterCodePoint": "0084","TargetCharacterCodePoint": "01A4"}
,{"SourceCharacterCodePoint": "000A","TargetCharacterCodePoint": "01A5"}
,{"SourceCharacterCodePoint": "0017","TargetCharacterCodePoint": "01A6"}
,{"SourceCharacterCodePoint": "001B","TargetCharacterCodePoint": "01A7"}
,{"SourceCharacterCodePoint": "0088","TargetCharacterCodePoint": "01A8"}
,{"SourceCharacterCodePoint": "0089","TargetCharacterCodePoint": "01A9"}
,{"SourceCharacterCodePoint": "008A","TargetCharacterCodePoint": "01AA"}
,{"SourceCharacterCodePoint": "008B","TargetCharacterCodePoint": "01AB"}
,{"SourceCharacterCodePoint": "008C","TargetCharacterCodePoint": "01AC"}
,{"SourceCharacterCodePoint": "0005","TargetCharacterCodePoint": "01AD"}
,{"SourceCharacterCodePoint": "0006","TargetCharacterCodePoint": "01AE"}
,{"SourceCharacterCodePoint": "0007","TargetCharacterCodePoint": "01AF"}
,{"SourceCharacterCodePoint": "0090","TargetCharacterCodePoint": "01B0"}
,{"SourceCharacterCodePoint": "0091","TargetCharacterCodePoint": "01B1"}
,{"SourceCharacterCodePoint": "0016","TargetCharacterCodePoint": "01B2"}
,{"SourceCharacterCodePoint": "0093","TargetCharacterCodePoint": "01B3"}
,{"SourceCharacterCodePoint": "0094","TargetCharacterCodePoint": "01B4"}
```

```
,{"SourceCharacterCodePoint": "0095","TargetCharacterCodePoint": "01B5"}
,{"SourceCharacterCodePoint": "0096","TargetCharacterCodePoint": "01B6"}
,{"SourceCharacterCodePoint": "0004","TargetCharacterCodePoint": "01B7"}
,{"SourceCharacterCodePoint": "0098","TargetCharacterCodePoint": "01B8"}
,{"SourceCharacterCodePoint": "0099","TargetCharacterCodePoint": "01B9"}
,{"SourceCharacterCodePoint": "009A","TargetCharacterCodePoint": "01BA"}
,{"SourceCharacterCodePoint": "009B","TargetCharacterCodePoint": "01BB"}
,{"SourceCharacterCodePoint": "0014","TargetCharacterCodePoint": "01BC"}
,{"SourceCharacterCodePoint": "0015","TargetCharacterCodePoint": "01BD"}
,{"SourceCharacterCodePoint": "009E","TargetCharacterCodePoint": "01BE"}
,{"SourceCharacterCodePoint": "001A","TargetCharacterCodePoint": "01BF"}
,{"SourceCharacterCodePoint": "009F","TargetCharacterCodePoint": "027F"}
```

「データ移行のターゲット」

AWS Database Migration Service (AWS DMS) では、最もよく利用されているデータベースの多くをデータレプリケーションのターゲットとして使用できます。ターゲットは Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、Amazon Relational Database Service (Amazon RDS) DB インスタンス、またはオンプレミス データベースが可能です。

有効なターゲットの包括的なリストについては、「AWS DMSのターゲット」をご参照ください。

> **Note**
>
> AWS DMS は、以下のターゲット エンドポイントタイプの AWS リージョン間の移行をサポートしていません。
>
> - 「Amazon DynamoDB」
> - Amazon OpenSearch Service
> - Amazon Kinesis Data Streams

トピック

- AWS Database Migration Serviceのターゲットとしての Oracle データベースの使用
- Microsoft SQL Server データベースの AWS Database Migration Service のターゲットとしての使用
- PostgreSQL データベースの AWS Database Migration Serviceのターゲットとしての使用

- MySQL 互換データベースの AWS Database Migration Serviceのターゲットとしての使用
- AWS Database Migration Service のターゲットとしての Amazon Redshift データベースの使用
- AWS Database Migration Service のターゲットとしての SAP ASE データベースの使用
- AWS Database Migration Service のターゲットに Amazon S3 を使用する
- AWS Database Migration Service のターゲットとしての Amazon DynamoDB データベースの使用
- のターゲットとしての Amazon Kinesis Data Streams の使用 AWS Database Migration Service
- のターゲットとしての Apache Kafka の使用 AWS Database Migration Service
- AWS Database Migration Service のターゲットとしての Amazon OpenSearch Service クラスターの使用
- AWS Database Migration Service のターゲットとしての Amazon DocumentDB の使用
- AWS Database Migration Service のターゲットとしての Amazon Neptune の使用
- AWS Database Migration Service のターゲットとしての Redis の使用
- AWS Database Migration Service のターゲットとしての Babelfish の使用
- Amazon Timestream を AWS Database Migration Service のターゲットとして使用する
- AWS DMS のターゲットとしての Amazon RDS for Db2 および IBM Db2 LUW の使用

AWS Database Migration Serviceのターゲットとしての Oracle データベースの使用

別の Oracle データベースまたはサポートされている他のデータベースのいずれかを使用して AWS DMS Oracle データベースターゲットにデータを移行できます。Secure Sockets Layer (SSL) を使用して、Oracle エンドポイントとレプリケーションインスタンスとの接続を暗号化できます。Oracle エンドポイントで SSL を使用する方法の詳細については、を参照してくださいでの SSL の使用 AWS Database Migration Service。 AWS DMS Oracle TDE ではデータベースに書き込む際に暗号化キーやパスワードを必要としないため、Oracle 透過的データ暗号化 (TDE) を使用してターゲットデータベースに保存されているデータを暗号化することもサポートされています。

AWS DMS ターゲットとしてサポートされている Oracle のバージョンについては、を参照してください。のターゲット AWS DMS

Oracle をターゲットとして使用するときは、ターゲット接続に使用されるスキーマまたはユーザーにデータを移行することを前提とします。別のスキーマにデータを移行する場合は、スキーマ変換を使用します。たとえば、ターゲットエンドポイントがユーザー RDSMASTER に接続しており、ユー

ザー PERFDATA1 から PERFDATA2 に移行したいとします。この場合、次のように変換を作成します。

```
{
   "rule-type": "transformation",
   "rule-id": "2",
   "rule-name": "2",
   "rule-action": "rename",
   "rule-target": "schema",
   "object-locator": {
   "schema-name": "PERFDATA1"
},
"value": "PERFDATA2"
}
```

Oracle をターゲットとして使用すると、 AWS DMS すべてのテーブルとインデックスがターゲットのデフォルトのテーブルスペースとインデックステーブルスペースに移行されます。テーブルとインデックスを別のテーブルとインデックスのテーブルスペースに移行する場合は、テーブルスペース変換を使用してこれを実行します。たとえば、Oracle ソース内の一部のテーブルスペースに割り当てられた INVENTORY スキーマに一連のテーブルがあるとします。移行については、このすべてのテーブルをターゲットの単一の INVENTORYSPACE テーブルスペースに割り当てるとします。この場合、次のように変換を作成します。

```
{
   "rule-type": "transformation",
   "rule-id": "3",
   "rule-name": "3",
   "rule-action": "rename",
   "rule-target": "table-tablespace",
   "object-locator": {
      "schema-name": "INVENTORY",
      "table-name": "%",
      "table-tablespace-name": "%"
   },
   "value": "INVENTORYSPACE"
}
```

変換の詳細については、「 JSON を使用するテーブル選択および変換を指定する」をご参照ください。

Oracle がソースとターゲットの両方である場合、Oracle ソースの追加の接続属性 `enableHomogenousTablespace=true` を設定することで、既存のテーブルまたはインデックステーブルスペースの割り当てを保持できます。詳細については、「のソースとして Oracle を使用する場合のエンドポイント設定 AWS DMS」をご参照ください。

Oracle データベースをのターゲットとして使用する方法の詳細については AWS DMS、以下のセクションを参照してください。

トピック

- Oracle をターゲットとする場合の制限事項 AWS Database Migration Service
- ターゲットとして Oracle を使用する場合に必要なユーザーアカウント権限
- Oracle データベースをターゲットとして設定する AWS Database Migration Service
- Oracle をターゲットとして使用する場合のエンドポイント設定 AWS DMS
- Oracle のターゲットデータ型

Oracle をターゲットとする場合の制限事項 AWS Database Migration Service

データ移行のターゲットとして Oracle を使用する場合の制限は、以下のとおりです。

- AWS DMS ターゲット Oracle データベースにはスキーマを作成しません。必要なすべてのスキーマをターゲット Oracle データベースで作成する必要があります。Oracle ターゲットのスキーマ名がすでに存在している必要があります。ソーススキーマのテーブルは、 AWS DMS ターゲットインスタンスへの接続に使用するユーザーまたはスキーマにインポートされます。複数のスキーマを移行するには、複数のレプリケーションタスクを作成します。データをターゲット上の別のスキーマに移行することもできます。そのためには、 AWS DMS テーブルマッピングにスキーマ変換ルールを使用する必要があります。
- AWS DMS INDEXTYPE CONTEXT `Use direct path full load` を含むテーブルのオプションはサポートされていません。回避策として、配列ロードを使用できます。
- バッチ最適化適用オプションでは、差分変更テーブルへのロードに直接パスが使用されるため、XML タイプはサポートされていません。回避策として、トランザクション適用モードを使用できます。
- ソースデータベースから移行された空の文字列は、Oracle ターゲットによって異なる方法で処理できます (たとえば、1 つのスペース文字列に変換されます)。これにより、 AWS DMS 検証で不一致が報告されることがあります。
- 次の式を使用して、バッチ最適化の適用モードでサポートされるテーブルごとの列の合計数を表すことができます。

```
2 * columns_in_original_table + columns_in_primary_key <= 999
```

例えば、元のテーブルに 25 列があり、そのプライマリキーが 5 列で構成されている場合、列の合計数は 55 になります。テーブルがサポートされている列数を超えると、 one-by-one すべての変更がモードで適用されます。

- AWS DMS オラクル・クラウド・インフラストラクチャ (OCI) 上のAutonomous DBをサポートしていません。

ターゲットとして Oracle を使用する場合に必要なユーザーアカウント権限

Oracle AWS Database Migration Service ターゲットをタスクで使用するには、Oracle データベースで次の権限を付与します。 AWS DMSへの Oracle データベース定義で指定されたユーザーアカウントにこの権限を付与します。

- SELECT ANY TRANSACTION
- V$NLS_PARAMETERS での SELECT
- V$TIMEZONE_NAMES での SELECT
- ALL_INDEXES での SELECT
- ALL_OBJECTS での SELECT
- DBA_OBJECTS での SELECT
- ALL_TABLES での SELECT
- ALL_USERS での SELECT
- ALL_CATALOG での SELECT
- ALL_CONSTRAINTS での SELECT
- ALL_CONS_COLUMNS での SELECT
- ALL_TAB_COLS での SELECT
- ALL_IND_COLUMNS での SELECT
- DROP ANY TABLE
- SELECT ANY TABLE
- INSERT ANY TABLE
- UPDATE ANY TABLE
- CREATE ANY VIEW

- DROP ANY VIEW
- CREATE ANY PROCEDURE
- ALTER ANY PROCEDURE
- DROP ANY PROCEDURE
- CREATE ANY SEQUENCE
- ALTER ANY SEQUENCE
- DROP ANY SEQUENCE
- テーブルの削除

以下に指定された要件のために、上記の追加の権限を付与します。

- 特定のテーブルリストを使用するには、レプリケートされたすべてのテーブルに SELECT を付与し、ALTER も付与します。
- ユーザーがデフォルトテーブルスペースにテーブルを作成することを許可するには、GRANT UNLIMITED TABLESPACE 権限を付与します。
- ログオンのために、CREATE SESSION 権限を付与します。
- 直接パス (全ロードのデフォルト) `GRANT LOCK ANY TABLE to dms_user;` を使用している場合。
- [DROP and CREATE] (ドロップして作成) テーブル準備モードを使用するときにスキーマが異なる場合は、`GRANT CREATE ANY INDEX to dms_user;`。
- 一部の全ロードシナリオでは、ターゲットテーブルスキーマが DMS ユーザーとは異なる場合、「DROP and CREATE table」または「TRUNCATE before loading」オプションを選択できます。この場合、DROP ANY TABLE を付与します。
- ターゲットテーブルスキーマが DMS ユーザーとは異なる変更テーブルまたは監査テーブルに変更を保存するには、CREATE ANY TABLE および CREATE ANY INDEX を付与します。

AWS Database Migration Service ターゲットデータベースに必要な読み取り権限

AWS DMS ユーザーアカウントには、次の DBA テーブルに対する読み取り権限が付与されている必要があります。

- DBA_USERS での SELECT
- DBA_TAB_PRIVS での SELECT
- DBA_OBJECTS での SELECT

- DBA_SYNONYMS での SELECT
- DBA_SEQUENCES での SELECT
- DBA_TYPES での SELECT
- DBA_INDEXES での SELECT
- DBA_TABLES での SELECT
- DBA_TRIGGERS での SELECT
- SELECT on SYS.DBA_REGISTRY

必要な権限のいずれかを V$xxx に付与できない場合は、V_$xxx に付与します。

移行前評価

Oracle の評価に記載されている移行前評価を Oracle を対象として使用するには、dms_userターゲットデータベースのデータベースユーザーに次の権限を追加する必要があります。

```
GRANT SELECT ON V_$INSTANCE TO dms_user;
```

Oracle データベースをターゲットとして設定する AWS Database Migration Service

Oracle データベースをデータ移行ターゲットとして使用する前に、に Oracle ユーザーアカウントを提供する必要があります AWS DMS。ユーザーアカウントには、ターゲットとして Oracle を使用する場合に必要なユーザーアカウント権限 で指定されているように、Oracle データベースでの読み取り/書き込み権限が必要です。

Oracle をターゲットとして使用する場合のエンドポイント設定 AWS DMS

エンドポイントの設定を使用して、追加の接続属性の使用する場合と同様に、ターゲットの Oracle のデータベースを設定できます。 AWS DMS コンソールを使用するか、`--oracle-settings '{"EndpointSetting": "value", ...}'` JSON `create-endpoint` 構文を使用してのコマンドを使用して、ターゲットエンドポイントを作成するときに設定を指定します。AWS CLI

次の表は、Oracle をターゲットとして使用できるエンドポイント設定を説明しています。

名前	説明
EscapeCharacter	この属性はエスケープ文字に設定する。このエスケープ文字を使うと、単一のワイルドカード文字をテーブル

名前	説明
	マッピング式で通常の文字のように動作させることができる。詳細については、「テーブルマッピングのワイルドカード」を参照してください。 デフォルト値: Null 有効値: ワイルドカード文字以外の任意の文字 例: `--oracle-settings '{"EscapeCharacter": "#"}'`
`UseDirectPathFullLoad`	に設定するとY、 AWS DMS ダイレクトパスフルロードを使用します。Oracle Call Interface (OCI) で直接パスプロトコルを使用可能にするには、この値を指定します。この OCI プロトコルを使用すると、完全ロード時に Oracle ターゲットテーブルを一括ロードできます。 デフォルト値: `true` 有効な値: `true/false` 例: `--oracle-settings '{"UseDirectPathFullLoad": false}'`

名前	説明
DirectPathParallelLoad	true に設定した場合、この属性は、UseDirectPathFullLoad が Y に設定されている場合に並列ロードを指定します。この属性は、 AWS DMS parallel ロード機能を使用する場合のみ適用されます。詳細については、parallel-load にある「テーブルとコレクション設定のルールとオペレーション オペレーションの説明」をご参照ください。 この並列ロード設定を指定するとき、ターゲットテーブルに制約やインデックスを設定できないという制限があります。この制限の詳細については、「Enabling Constraints After a Parallel Direct Path Load」をご参照ください。制約またはインデックスが有効になっている場合、この属性を true に設定しても効果はありません。 デフォルト値: false 有効な値: true/false 例: --oracle-settings '{"DirectPathParallelLoad": true}'
DirectPathNoLog	この属性を true に設定すると、データベースログに証跡を書き込まずにテーブルに直接書き込むことで、Oracle ターゲットデータベースのコミットレートを上げることができます。詳細については、「Direct-Load INSERT」をご参照ください。この属性は、UseDirectPathFullLoad を Y に設定した場合のみ適用されます。 デフォルト値: false 有効な値: true/false 例: --oracle-settings '{"DirectPathNoLog": true}'

名前	説明
CharLengthSemantics	文字の列の長さをバイト単位または文字単位で指定します。文字の列の長さが文字単位であることを示すには、この属性を CHAR に設定します。それ以外の場合、文字の列の長さはバイト単位です。 デフォルト値: CHAR に設定されていません 有効な値: CHAR 例: --oracle-settings '{"CharLengthSemantics": "CHAR"}'
AlwaysReplaceEmptyString	AWS DMS Oracle ターゲットへの移行時に、空の文字列を複製するための余分なスペースが追加されます。一般に、Oracle には空の文字列の表記はありません。varchar2 に空の文字列を挿入すると、空の文字列を NULL としてロードします。Oracle にデータを NULL として挿入する場合は、この属性を FALSE に設定します。 デフォルト値: true 有効な値: true/false 例: --oracle-settings '{"AlwaysReplaceEmptyString": false}'

Oracle のターゲットデータ型

で使用されるターゲット Oracle データベースは、ほとんどの Oracle AWS DMS データ型をサポートします。次の表は、使用時にサポートされる Oracle AWS DMS ターゲットデータ型と、 AWS DMS データ型からのデフォルトマッピングを示しています。ソースからマッピングされるデータ型を表示する方法の詳細については、使用しているソースのセクションをご参照ください。

AWS DMS データ型	Oracle のデータ型
BOOLEAN	NUMBER (1)

AWS DMS データ型	Oracle のデータ型
BYTES	RAW (長さ)
DATE	DATETIME
TIME	タイムスタンプ (0)
DATETIME	タイムスタンプ (スケール)
INT1	NUMBER (3)
INT2	NUMBER (5)
INT4	NUMBER (10)
INT8	NUMBER (19)
NUMERIC	NUMBER (p,s)
REAL4	FLOAT
REAL8	FLOAT
STRING	date を指定: DATE time を指定: タイムスタンプ timestamp を指定: タイムスタンプ timestamp_with_timezone を指定: タイムスタンプ時間帯あり timestamp_with_local_timezone を指定: タイムスタンプ WITH LOCAL TIMEZONE interval_year_to_month を指定: INTERVAL YEAR TO MONTH interval_day_to_second を指定: INTERVAL DAY TO SECOND 長さ > 4000 の場合: CLOB 他のすべての場合: VARCHAR2 (長さ)

AWS DMS データ型	Oracle のデータ型
UINT1	NUMBER (3)
UINT2	NUMBER (5)
UINT4	NUMBER (10)
UINT8	NUMBER (19)
WSTRING	長さ > 2000 の場合: NCLOB 他のすべての場合: NVARCHAR2 (長さ)
BLOB	BLOB このデータ型を使用するには AWS DMS、特定のタスクで BLOB の使用を有効にする必要があります。BLOB データ型は、プライマリキーを含むテーブルでのみサポートされます。
CLOB	CLOB このデータ型をと共に使用するには AWS DMS、特定のタスクで CLOB の使用を有効にする必要があります。変更データキャプチャ (CDC) 中は、プライマリキーを含むテーブルでのみ CLOB データ型がサポートされます。 STRING 宣言されたサイズが 4000 バイトを超えるソースの Oracle VARCHAR2 データ型は、 AWS DMS CLOB を介して Oracle ターゲットの文字列にマップされます。

AWS DMS データ型	Oracle のデータ型
NCLOB	NCLOB このデータ型をで使用するには AWS DMS、特定のタスクで NCLOB の使用を有効にする必要があります。CDC 中、プライマリキーを含むテーブルでのみ NCLOB データ型がサポートされます。 WSTRING 宣言されたサイズが 4000 バイトを超えるソース上の Oracle VARCHAR2 データ型は、 AWS DMS NCLOB を介して Oracle ターゲット上の WSTRING にマップされます。
XMLTYPE	XMLTYPE ターゲットデータ型は、Oracle 間レプリケーションタスクにのみ関連しています。 ソースデータベースが Oracle の場合、ソースデータ型はそのままの状態で Oracle ターゲットにレプリケートされます。たとえば、ソースにおける XMLTYPE データ型は、ターゲットでは XMLTYPE データ型として作成されます。

Microsoft SQL Server データベースの AWS Database Migration Service のターゲットとしての使用

AWS DMS を使用して、Microsoft SQL Server データベースにデータを移行できます。SQL Server データベースをターゲットとして使用すると、別の SQL Server データベースまたはサポートされている他のデータベースのいずれかからデータを移行できます。

AWS DMS がターゲットとしてサポートする SQL Server のバージョンについては、「のターゲット AWS DMS」を参照してください。

AWS DMS は、Enterprise、Standard、Workgroup、Developer のオンプレミスおよび Amazon RDS エディションをサポートします。

AWS DMS および SQL Server ターゲットデータベースを使用する方法の詳細については、以下をご参照ください。

トピック

- SQL Server を AWS Database Migration Service のターゲットとして使用する場合の制限
- AWS Database Migration Service のターゲットとして SQL Server を使用する場合のセキュリティ要件
- SQL Server を AWS DMS のターゲットとして使用する場合のエンドポイントの設定
- Microsoft SQL Server のターゲットデータ型

SQL Server を AWS Database Migration Service のターゲットとして使用する場合の制限

SQL Server データベースを AWS DMS のターゲットとして使用する場合、以下の制限が適用されます。

- 計算列を含む SQL Server ターゲットテーブルを手動で作成する場合、BCP 一括コピーユーティリティを使用する全ロードレプリケーションがサポートされません。全ロードレプリケーションを使用するには、追加接続属性 (ECA) を設定して`'useBCPFullLoad=false'` エンドポイントで BCP ロードを無効にします。エンドポイントでの ECA 設定の詳細については、「ソースおよびターゲットエンドポイントの作成」をご参照ください。BCP の詳細な操作方法については、「Microsoft SQL Server ドキュメント」をご参照ください。
- SQL Server 空間データ型 (GEOMETRY と GEOGRAPHY) を持つテーブルをレプリケートすると、挿入した空間参照識別子 (SRID) がすべて AWS DMS によりデフォルトの SRID に置き換えられます。デフォルトの SRID は、GEOMETRY は 0、GEOGRAPHY は 4326 です。
- 一時テーブルはサポートされていません。ターゲット上に手動で作成されている場合に、一時テーブルを移行すると、トランザクション適用モードのレプリケーションのみのタスクで動作することがあります。
- 現在のところ、PostgreSQL ソースの `boolean` データ型は、一貫性のない値を持つ `bit` データ型として SQLServer ターゲットに移行されます。

  回避策として、次を実行します。

  - 列の `VARCHAR(1)` データ型を使用してテーブルを事前に作成します (または、AWS DMS で自動的にテーブルを作成します)。次に、ダウンストリーム処理で「F」を False、「T」を True として扱います。
  - ダウンストリーム処理を変更する必要がないようにするには、「F」の値を「0」に、「T」の値を「1」に変更する変換ルールをタスクに追加して、SQL Server のビットデータ型として格納します。

- AWS DMS では、列の null 機能を設定するための変更処理はサポートされていません (ALTER COLUMN [SET|DROP] NOT NULL 句を使用した ALTER TABLE ステートメント)。
- Windows 認証はサポートされていません。

AWS Database Migration Service のターゲットとして SQL Server を使用する場合のセキュリティ要件

以下では、Microsoft SQL Server ターゲットとともに AWS DMS を使用する場合のセキュリティ要件について説明します。

- AWS DMS ユーザーアカウントには、接続先の SQL Server データベースに対する db_owner ユーザーロールが最低限必要です。
- SQL Server システム管理者は、すべての AWS DMS ユーザーアカウントにこのアクセス許可を付与する必要があります。

SQL Server を AWS DMS のターゲットとして使用する場合のエンドポイントの設定

追加の接続属性の使用と同様、エンドポイントの設定を使用して、ターゲットの SQL Server データベースを設定できます。ターゲットエンドポイントを作成する際に、AWS DMS コンソールを使用するか、AWS CLI で --microsoft-sql-server-settings '{"*EndpointSetting*": *"value"*, *...*}' の JSON 構文を指定して create-endpoint コマンドを使用して設定を指定します。

次の表は、ターゲットとして SQL Server を使用できるエンドポイント設定を説明しています。

名前	説明
ControlTablesFileGroup	AWS DMS 内部テーブルのファイルグループを指定します。レプリケーションタスクが開始されると、すべての内部 AWS DMS 制御テーブル (awsdms_ apply_exception、awsdms_apply、awsdms_changes) が、指定したファイルグループで作成されます。 デフォルト値: 該当なし 有効な値: 文字列

名前	説明

	例: `--microsoft-sql-server-settings '{"ControlTablesFileGroup": "filegroup1"}'`
	
	ファイルグループを作成するコマンドの例を次に示します。
	
	`ALTER DATABASE replicate ADD FILEGROUP Test1FG1;`
	`GO ALTER DATABASE replicate`
	`ADD FILE (`
	`NAME = test1dat5,`
	`FILENAME = 'C:\temp\DATA\t1dat5.ndf',`
	
	`SIZE = 5MB,`
	`MAXSIZE = 100MB,`
	`FILEGROWTH = 5MB`
	`)`
	`TO FILEGROUP Test1FG1;`
	`GO`

`ExecuteTimeout`	この追加の接続属性 (ECA) を使用して、SQL Server インスタンスのクライアントステートメントのタイムアウトを秒単位で設定する。デフォルト値は 60 秒です。 例: `'{"ExecuteTimeout": 100}'`

名前	説明
UseBCPFullLoad	BCP を使用して全ロードオペレーション用のデータを転送するには、この属性を使用します。ターゲットテーブルに、ソーステーブル内に存在しない IDENTITY 列が含まれている場合、[use BCP for loading table] (テーブルのロードに BCP を使用) オプションを無効にする必要があります。 デフォルト値: true 有効な値: true/false 例: `--microsoft-sql-server-settings '{"UseBCPFullLoad": false}'`

Microsoft SQL Server のターゲットデータ型

次の表に、AWS DMS を使用する場合にサポートされる Microsoft SQL Server のターゲットデータ型と、AWS DMS のデータ型からのデフォルトマッピングを示します。AWS DMS のデータ型の詳細については、「AWS Database Migration Service のデータ型」をご参照ください。

AWS DMS データ型	SQL Server のデータ型
BOOLEAN	TINYINT
BYTES	VARBINARY(長さ)
DATE	SQL Server 2008 以降の場合、DATE を使用する。 それより前のバージョンでは、スケールが 3 以下の場合は DATETIME を使用します。他のすべての場合は、VARCHAR (37) を使用します。
TIME	SQL Server 2008 以降の場合、DATETIME2 (%d) を使用する。

AWS DMS データ型	SQL Server のデータ型
	それより前のバージョンでは、スケールが 3 以下の場合は DATETIME を使用します。他のすべての場合は、VARCHAR (37) を使用します。
DATETIME	SQL Server 2008 以降では、DATETIME2 (位取り) を使用する。 それより前のバージョンでは、スケールが 3 以下の場合は DATETIME を使用します。他のすべての場合は、VARCHAR (37) を使用します。
INT1	SMALLINT
INT2	SMALLINT
INT4	INT
INT8	BIGINT
NUMERIC	NUMERIC (p,s)
REAL4	REAL
REAL8	FLOAT
STRING	列が日付または時刻列の場合、次の操作を行います。 • SQL Server 2008 以降の場合、DATETIME2 を使用する。 • それより前のバージョンでは、スケールが 3 以下の場合は DATETIME を使用します。他のすべての場合は、VARCHAR (37) を使用します。 列が日付または時刻列ではない場合、VARCHAR (長さ) を使用します。
UINT1	TINYINT
UINT2	SMALLINT

AWS DMS データ型	SQL Server のデータ型
UINT4	INT
UINT8	BIGINT
WSTRING	NVARCHAR (長さ)
BLOB	VARBINARY(最大) IMAGE AWS DMS でこのデータ型を使用する場合は、特定のタスク用に BLOB の使用を有効にする必要があります。AWS DMS では、プライマリキーを含むテーブルでのみ BLOB データ型がサポートされます。
CLOB	VARCHAR(max) AWS DMS でこのデータ型を使用するには、特定のタスク用に CLOB の使用を有効にする必要があります。変更データキャプチャ (CDC) 中、AWS DMS はプライマリキーを含むテーブルでのみ CLOB データ型をサポートしています。
NCLOB	NVARCHAR(最大) AWS DMS でこのデータ型を使用するには、特定のタスク用に NCLOB の使用を有効にする必要があります。CDC 中、AWS DMS はプライマリキーを含むテーブルでのみ NCLOB データ型をサポートします。

PostgreSQL データベースの AWS Database Migration Serviceのターゲットとしての使用

別の PostgreSQL データベースまたはサポートされている他のデータベースのいずれかから AWS DMS、 を使用して PostgreSQL データベースにデータを移行できます。

がターゲットとして AWS DMS サポートする PostgreSQL のバージョンについては、「」を参照してくださいのターゲット AWS DMS。

Note

- Amazon Aurora Serverless は、PostgreSQL との互換性を持つ Amazon Aurora のターゲットとして利用できます。Amazon Aurora Serverless の詳細については、「Amazon Aurora ユーザーガイド」の「Amazon Aurora Serverless v2 の使用」を参照してください。
- Aurora Serverless DB クラスターは Amazon VPC からのみアクセスでき、[public IP address] (パブリック IP アドレス) を使用することはできません。そのため、レプリケーション インスタンスを Aurora PostgreSQL Serverless とは異なるリージョンに配置する場合は、[vpc peering] (VPC ピアリング接続) を構成します。それ以外の場合は、Aurora PostgreSQL Serverless [regions] (リージョン) の可用性をチェックし、これらのリージョンのいずれかを Aurora PostgreSQL Serverless とレプリケーション インスタンスの両方に使用しました。
- Babelfish の機能は Amazon Aurora に組み込まれており、追加のコストは発生しません。詳細については、「Using Babelfish for Aurora PostgreSQL as a target for AWS Database Migration Service」を参照してください。

AWS DMS は、フルロードフェーズでソースからターゲットにデータを移行するときに table-by-table アプローチを取ります。全ロードフェーズ中のテーブルの順序は保証されません。テーブル全ロードフェーズ中、および個々のテーブルのキャッシュしたトランザクションが適用されている間は、テーブルは同期されません。その結果、アクティブな参照整合性制約により、全ロードフェーズ中にタスクが失敗する可能性があります。

PostgreSQL では、外部キー (参照整合性制約) はトリガーを使用して実装されます。全ロードフェーズでは、 は各テーブルを一度に 1 つずつ AWS DMS ロードします。次のいずれかの方法を使用して、全ロード中に外部キーの制約を無効にすることを強くお勧めします。

- インスタンスからすべてのトリガーを一時的に無効にして、全ロードを終了します。
- PostgreSQL では、`session_replication_role` パラメータを使用します。

特定の時間において、トリガーは `origin`、`replica`、`always`、または `disabled` のいずれかの状態になります。`session_replication_role` パラメータが `replica` に設定されている場合、`replica` 状態のトリガーのみがアクティブになり、呼び出されると実行されます。それ以外の場合、トリガーは非アクティブなままです。

PostgreSQL には、session_replication_role が設定されている場合でも、テーブルの切り捨てを防止するフェールセーフメカニズムが備わっています。これを、トリガーを無効にする代わりに使用して、全ロードの完了を支援できます。これを行うには、ターゲットテーブルの準備モードを DO_NOTHING に設定します それ以外の場合、外部キーの制約があると DROP および TRUNCATE オペレーションは失敗します。

Amazon RDS では、パラメータグループを使用してこのパラメータの設定を管理できます。Amazon EC2 で実行されている PostgreSQL インスタンスの場合、パラメータを直接設定できます。

のターゲットとして PostgreSQL データベースを使用する方法の詳細については AWS DMS、以下のセクションを参照してください。

トピック

- のターゲットとしての PostgreSQL の使用に関する制限 AWS Database Migration Service
- のターゲットとして PostgreSQL データベースを使用する場合のセキュリティ要件 AWS Database Migration Service
- のターゲットとして PostgreSQL を使用する場合のエンドポイント設定と追加の接続属性 (ECAs) AWS DMS
- PostgreSQL のターゲットデータ型
- のターゲットとしての Babelfish for Aurora PostgreSQL の使用 AWS Database Migration Service

のターゲットとしての PostgreSQL の使用に関する制限 AWS Database Migration Service

PostgreSQL データベースを AWS DMSのターゲットとして使用する場合、以下の制限が適用されます。

- 異種の移行の場合、JSON データ型は内部でネイティブな CLOB データ型に変換されます。
- Oracle から PostgreSQL への移行では、Oracle の列に NULL 文字 (16 進値 U+0000) が含まれている場合、 は NULL 文字をスペース (16 進値 U+0020) AWS DMS に変換します。これは、PostgreSQL の制限によるものです。
- AWS DMS は、coalesce 関数で作成された一意のインデックスを持つテーブルへのレプリケーションをサポートしていません。
- テーブルがシーケンスを使用している場合は、ソースデータベースからレプリケーションを停止した後、ターゲットデータベース内の各シーケンスNEXTVALの の値を更新します。 はソースデー

タベースからデータ AWS DMS をコピーしますが、進行中のレプリケーション中はシーケンスをターゲットに移行しません。

のターゲットとして PostgreSQL データベースを使用する場合のセキュリティ要件 AWS Database Migration Service

セキュリティ上の観点から、データ移行に使用されるユーザーアカウントは、ターゲットとして使用する PostgreSQL データベースにおける登録済みユーザーにする必要があります。

PostgreSQL ターゲットエンドポイントでは、 AWS DMS 移行を実行するために最小限のユーザーアクセス許可が必要です。以下の例を参照してください。

```
    CREATE USER newuser WITH PASSWORD 'your-password';
    ALTER SCHEMA schema_name OWNER TO newuser;
```

または

```
    GRANT USAGE ON SCHEMA schema_name TO myuser;
    GRANT CONNECT ON DATABASE postgres to myuser;
    GRANT CREATE ON DATABASE postgres TO myuser;
    GRANT CREATE ON SCHEMA schema_name TO myuser;
    GRANT UPDATE, INSERT, SELECT, DELETE, TRUNCATE ON ALL TABLES IN SCHEMA schema_name
TO myuser;
    GRANT TRUNCATE ON schema_name."BasicFeed" TO myuser;
```

のターゲットとして PostgreSQL を使用する場合のエンドポイント設定と追加の接続属性 (ECAs) AWS DMS

エンドポイント設定と追加の接続属性 (ECAs) を使用して、PostgreSQL ターゲットデータベースを設定できます。

AWS DMS コンソールを使用するか、`--postgre-sql-settings '{"EndpointSetting": "value", ...}'`JSON 構文で の `create-endpoint` コマンドを使用してAWS CLI、ターゲットエンドポイントを作成するときに設定を指定します。

エンドポイントの `ExtraConnectionAttributes`パラメータを使用して ECAs を指定します。

次の表は、PostgreSQL をターゲットとして使用できるエンドポイント設定を説明しています。

名前	説明
MaxFileSize	PostgreSQL へのデータ転送に使用される .csv ファイルの最大サイズ (KB 単位) を指定します。 デフォルト値: 32768 KB (32 MB) 有効な値:1 ~ 1,048,576 KB (最大 1.1 GB) 例: `--postgre-sql-settings '{"MaxFileSize": 512}'`
ExecuteTimeout	PostgreSQL インスタンスのクライアントステートメントタイムアウト (秒単位) を設定します。デフォルト値は 60 秒です。 例: `--postgre-sql-settings '{"ExecuteTimeout": 100}'`
AfterConnectScript= SET session_replication_role = replica	この属性は、外部キーとユーザートリガーを AWS DMS バイパスして、データの一括ロードにかかる時間を短縮します。
MapUnboundedNumericAsString	このパラメータは、数値の精度を失うことなく正常に移行するために、境界なしの NUMERIC データ型の列を文字列として扱います。このパラメーターは、PostgreSQL ソースから PostgreSQL ターゲットへのレプリケーション、または PostgreSQL 互換のデータベースにのみ使用します。 デフォルト値: false 有効な値: false/true 例: `--postgre-sql-settings '{"MapUnboundedNumericAsString": "true"}`

名前	説明
	このパラメータを使用すると、数値から文字列への変換、および数値への変換により、レプリケーションのパフォーマンスが低下する可能性があります。このパラメータは、DMS バージョン 3.4.4 以降で使用するためにサポートされています **Note** PostgreSQL のソースとターゲット エンドポイントを一緒にのみ `MapUnboundedNumericAsString` を使用します。 ソース PostgreSQL エンドポイントで `MapUnboundedNumericAsString` を使用すると、CDC 中に精度が 28 に制限されます。ターゲット エンドポイントで `MapUnboundedNumericAsString` を使用すると、精度 28 スケール 6 でデータを移行します。 PostgreSQL 以外のターゲットとは `MapUnboundedNumericAsString` を使用しません。
`loadUsingCSV`	この追加接続属性 (ECA) を使用して、\COPY コマンドを使用して全ロードオペレーションのデータを転送します。 デフォルト値: `true` 有効な値: true/false ECA の例:`loadUsingCSV=true;` 注： この ECA を false に設定すると、INSERTs が直接実行されるため、レプリケーションのパフォーマンスが低下する可能性があります。

名前	説明
DatabaseMode	この属性を使用して、Babelfish エンドポイントなど、追加の設定を必要とする PostgreSQL 互換エンドポイントのレプリケーション処理のデフォルトの動作を変更する。 デフォルト値: DEFAULT 有効な値: DEFAULT、BABELFISH 例: DatabaseMode=default;
BabelfishDatabaseName	この属性を使用して、移行先のターゲット Babelfish T-SQL データベース名を指定する。これは、DatabaseMode が Babelfish に設定されている場合に必要です。これは予約済みの babelfish_db データベースではない。 例: BabelfishDatabaseName=TargetDb;

PostgreSQL のターゲットデータ型

の PostgreSQL データベースエンドポイントは、ほとんどの PostgreSQL データベースデータ型 AWS DMS をサポートしています。次の表は、 の使用時にサポートされる PostgreSQL データベースターゲットデータ型 AWS DMS と、 AWS DMS データ型からのデフォルトのマッピングを示しています。

AWS DMS データ型の詳細については、「」を参照してくださいAWS Database Migration Service のデータ型。

AWS DMS データ型	PostgreSQL のデータ型
BOOLEAN	BOOLEAN
BLOB	BYTEA
BYTES	BYTEA

AWS DMS データ型	PostgreSQL のデータ型
DATE	DATE
TIME	TIME
DATETIME	スケールが 0 ~ 6 の場合、タイムスタンプ を使用します。 スケールが 7 ~ 9 の場合、VARCHAR (37) を使用します。
INT1	SMALLINT
INT2	SMALLINT
INT4	INTEGER
INT8	BIGINT
NUMERIC	DECIMAL (P,S)
REAL4	FLOAT4
REAL8	FLOAT8
STRING	長さが 1 ~ 21,845 の場合、VARCHAR (バイト単位の長さ) を使用します。 長さが 21,846 ~ 2,147,483,647 の場合、VARCHAR (65535) を使用します。
UINT1	SMALLINT
UINT2	INTEGER
UINT4	BIGINT
UINT8	BIGINT

AWS DMS データ型	PostgreSQL のデータ型
WSTRING	長さが 1 ~ 21,845 の場合、VARCHAR (バイト単位の長さ) を使用します。 長さが 21,846 ~ 2,147,483,647 の場合、VARCHAR (65535) を使用します。
NCLOB	TEXT
CLOB	TEXT

> **Note**
>
> PostgreSQL ソースからレプリケートする場合、 は、ユーザー定義のデータ型の列を除き、すべての列に対して同じデータ型のターゲットテーブル AWS DMS を作成します。このような場合、データ型はターゲットで「character varying」として作成されます。

のターゲットとしての Babelfish for Aurora PostgreSQL の使用 AWS Database Migration Service

AWS Database Migration Serviceを使用して SQL Server ソーステーブルを Babelfish for Amazon Aurora PostgreSQL のターゲットに移行できます。Babelfishを使用すると、Aurora PostgreSQL は Microsoft SQL Server 独自の SQL 言語である T-SQL を認識し、同じ通信プロトコルをサポートします。これにより、SQL Server 向けに作成されたアプリケーションのコード変更量が低減し、Aurora と連携できるようになります。Babelfish の機能は Amazon Aurora に組み込まれており、追加のコストは発生しません。Amazon Aurora クラスターのBabelfish は、Amazon RDS コンソールで有効化できます。

AWS DMS コンソール、API、または CLI コマンドを使用して AWS DMS ターゲットエンドポイントを作成するときは、ターゲットエンジンを Amazon Aurora PostgreSQL として指定し、データベースに babelfish_db という名前を付けます。[エンドポイント設定] セクションで、`DatabaseMode` を `Babelfish` に設定して、`BabelfishDatabaseName` をターゲットの Babelfish T-SQL データベース名に指定する設定を追加します。

移行タスクへの変換ルールの追加

Babelfish のターゲットの移行タスクを定義する場合、DMS がターゲットデータベース内で事前に作成された T-SQL Babelfish テーブルを使用するようにする変換ルールを含める必要があります。

まず、すべてのテーブル名を小文字にする変換ルールを移行タスクに追加します。Babelfish は、T-SQLを使用して作成したテーブル名前 PostgreSQL の `pg_class` カタログに小文字で保存します。ただし、大文字と小文字が混在する名前の SQL Server テーブルがある場合、DMS は T-SQL 互換のデータ型ではなく PostgreSQL ネイティブのデータ型を使用してテーブルを作成します。このため、すべてのテーブル名を小文字にする変換ルールを必ず追加します。列名は小文字に変換しないように注意してください。

次に、クラスターを定義する際にマルチデータベース移行モードを使用した場合は、元の SQL Server スキーマ名を変更する変換ルールを追加します。T-SQL データベース名が含まれるように SQL Server スキーマ名は必ず変更します。例えば、元の SQL Server のスキーマ名が dbo で、T-SQL データベース名が `mydb` の場合、変換ルールを使用してスキーマ名を `mydb_dbo` と変更します。

単一のデータベースモードを使用する場合、スキーマ名を変更するための変換ルールは必要ありません。スキーマ名には、Babelfish のターゲット T-SQL データベースとの one-to-one マッピングがあります。

次の変換ルールサンプルでは、すべてのテーブル名を小文字にして、元の SQL Server スキーマ名を dbo から `mydb_dbo` に変更しています。

```
{
   "rules": [
   {
      "rule-type": "transformation",
      "rule-id": "566251737",
      "rule-name": "566251737",
      "rule-target": "schema",
      "object-locator": {
         "schema-name": "dbo"
      },
      "rule-action": "rename",
      "value": "mydb_dbo",
      "old-value": null
   },
   {
      "rule-type": "transformation",
```

```
      "rule-id": "566139410",
      "rule-name": "566139410",
      "rule-target": "table",
      "object-locator": {
         "schema-name": "%",
         "table-name": "%"
      },
      "rule-action": "convert-lowercase",
      "value": null,
      "old-value": null
   },
   {
      "rule-type": "selection",
      "rule-id": "566111704",
      "rule-name": "566111704",
      "object-locator": {
         "schema-name": "dbo",
         "table-name": "%"
      },
      "rule-action": "include",
      "filters": []
   }
]
}
```

Babelfish テーブルで PostgreSQL のターゲットエンドポイントを使用する場合の制限

Babelfish テーブルで PostgreSQL のターゲットエンドポイントを使用する場合、次の制限が適用されます。

- [ターゲットテーブル作成モード] では、[何もしない] モードまたは [切り捨て] モードのみを使用します。[ターゲット上のテーブルを削除] モードは使用しないでください。このモードの場合、DMS は T-SQL が認識しない可能性のある PostgreSQL テーブルとしてテーブルを作成します。
- AWS DMS は sql_variant データ型をサポートしていません。
- Babelfish は、`HEIRARCHYID` データ型、`GEOMETRY` データ型、`GEOGRAPHY` データ型をサポートしていません。上記のデータ型を移行するには、変換ルールを追加してデータ型を `wstring(250)` に変換します。
- Babelfish での `BINARY` データ型、`VARBINARY` データ型、`IMAGE` データ型の以降は、`BYTEA` データ型を使用することでのみサポートされます。Aurora PostgreSQL の以前のバージョンの場

合、DMS を使用してこのようなテーブルを Babelfish のターゲットエンドポイント に移行できます。次の例のとおり、BYTEA データ型で長さを指定する必要はありません。

```
[Picture] [VARBINARY](max) NULL
```

上記の T-SQL データ型を T-SQL がサポートする BYTEA データ型に変更します。

```
[Picture] BYTEA NULL
```

- Aurora PostgreSQL Babelfish の以前のバージョンでは、PostgreSQL ターゲットエンドポイントを使用して SQL Server から Babelfish への継続的なレプリケーションのための移行タスクを作成する場合、IDENTITY 列を使用するすべてのテーブルに SERIAL データ型を割り当てる必要があります。Aurora PostgreSQL (バージョン 15.3、14.8 以降) と Babelfish (バージョン 3.2.0 以降) 以降では、アイデンティティ列がサポートされるようになったため、SERIAL データ型を割り当てる必要はありません。詳細については、「SQL Server to Aurora PostgreSQL 移行プレイブック」の「Sequences and Identity」セクションの「SERIAL Usage」を参照してください。Babelfish でテーブルを作成する際は、列の定義を次のとおり変更します。

```
    [IDCol] [INT] IDENTITY(1,1) NOT NULL PRIMARY KEY
```

上記の内容を次のとおり変更します。

```
    [IDCol] SERIAL PRIMARY KEY
```

Babelfish 互換の Aurora PostgreSQL では、デフォルト設定を使用してシーケンスを作成し、列に NOT NULL 制約を追加します。新たに作成されるシーケンスは通常のシーケンス (1 でインクリメント) と同様に動作し、複合型の SERIAL のオプションはありません。

- IDENTITY 列または SERIAL データ型を使用するテーブルのデータを移行した後、列の最大値に基づいて PostgreSQL ベースのシーケンスオブジェクトをリセットします。テーブルのフルロード実行後、次の T-SQL クエリを使用してステートメントを生成し、関連づけられたシーケンスオブジェクトをシードします。

```
DECLARE @schema_prefix NVARCHAR(200) = ''

IF current_setting('babelfishpg_tsql.migration_mode') = 'multi-db'
        SET @schema_prefix = db_name() + '_'
```

```
SELECT 'SELECT setval(pg_get_serial_sequence(''' + @schema_prefix +
 schema_name(tables.schema_id) + '.' + tables.name + ''', ''' + columns.name + ''')
               ,(select max(' + columns.name + ') from ' +
 schema_name(tables.schema_id) + '.' + tables.name + '));'
FROM sys.tables tables
JOIN sys.columns columns ON tables.object_id = columns.object_id
WHERE columns.is_identity = 1

UNION ALL

SELECT 'SELECT setval(pg_get_serial_sequence(''' + @schema_prefix + table_schema +
 '.' + table_name + ''',
''' + column_name + '''),(select max(' + column_name + ') from ' + table_schema + '.'
 + table_name + '));'
FROM information_schema.columns
WHERE column_default LIKE 'nextval(%';
```

このクエリは、最大 IDENTITY 値と SERIAL 値を更新する SELECT ステートメントのセットを生成します。

- バージョン 3.2 以前の Babelfish では、[完全 LOB モード] の場合、テーブルエラーが発生する可能性があります。エラーが発生した場合は、ロードに失敗したテーブルのために別のタスクを作成します。その後、[制限付き LOB モード] を使用して [最大 LOB サイズ (KB)] に適切な値を指定します。もう 1 つのオプションとして、SQL Server エンドポイント接続属性設定を `ForceFullLob=True` と指定する方法があります。
- バージョン 3.2 以前の Babelfish では、整数ベースのプライマリキーを使用しない Babelfish テーブルでデータ検証を実行すると、適切な一意のキーが見つからないというメッセージが表示されます。整数以外のプライマリキーのデータ検証は、Aurora PostgreSQL (バージョン 15.3、14.8 以降) と Babelfish (バージョン 3.2.0 以降) 以降ではサポートされています。
- 秒の小数点以下の桁数の精度の違いにより、DMS は `DATETIME` データ型を使用する Babelfish テーブルでデータ検証エラーを報告します。このようなエラーが表示されないようにするには、`DATETIME` データ型に次のとおりの検証ルールタイプを追加します。

```
{
         "rule-type": "validation",
         "rule-id": "3",
         "rule-name": "3",
         "rule-target": "column",
         "object-locator": {
             "schema-name": "dbo",
             "table-name": "%",
```

```
            "column-name": "%",
            "data-type": "datetime"
        },
        "rule-action": "override-validation-function",
        "source-function": "case when ${column-name} is NULL then NULL else 0 end",
        "target-function": "case when ${column-name} is NULL then NULL else 0 end"
    }
```

MySQL 互換データベースの AWS Database Migration Serviceのターゲットとしての使用

をサポートしている任意のソースデータエンジンから AWS DMS、を使用して MySQL 互換データベースにデータを移行できます。 AWS DMS オンプレミスの MySQL 互換データベースに移行する場合は、 AWS DMS ソースエンジンがエコシステム内にある必要があります。 AWS エンジンは、Amazon RDS、Amazon Aurora、Amazon S3 AWSなどのマネージド型サービスに搭載できます。または、エンジンは Amazon EC2 の自己管理型データベース上に存在していてもかまいません。

SSL を使用して、MySQL 互換のエンドポイントとレプリケーションインスタンスとの接続を暗号化できます。MySQL 互換のエンドポイントで SSL を使用する方法の詳細については、「での SSL の使用 AWS Database Migration Service」をご参照ください。

AWS DMS ターゲットとしてサポートする MySQL のバージョンについては、を参照してくださいのターゲット AWS DMS。

次の MySQL 互換データベースをターゲットとして使用できます。 AWS DMS

- MySQL Community Edition
- MySQL Standard Edition
- MySQL Enterprise Edition
- MySQL Cluster Carrier Grade Edition
- MariaDB Community Edition
- MariaDB Enterprise Edition
- MariaDB Column Store
- Amazon Aurora MySQL

> **Note**
>
> ソースストレージエンジン (MyISAM、MEMORY など) にかかわらず、 AWS DMS によってデフォルトで InnoDB テーブルとして MySQL 互換のターゲットテーブルが作成されます。InnoDB 以外のストレージエンジンのテーブルが必要な場合は、手動でテーブルを MySQL 互換のターゲットで作成し、[何もしない] オプションを使用して移行できます。詳細については、「全ロードタスク設定」をご参照ください。

AWS DMSのターゲットとしての MySQL 互換データベースの使用の詳細については、以下のセクションをご参照ください。

トピック

- 任意の MySQL 互換データベースをターゲットとして使用する AWS Database Migration Service
- MySQL 互換データベースをターゲットとして使用する場合の制限事項 AWS Database Migration Service
- MySQL 互換データベースをターゲットとして使用する場合のエンドポイント設定 AWS DMS
- MySQL のターゲットデータ型

任意の MySQL 互換データベースをターゲットとして使用する AWS Database Migration Service

MySQL 互換データベースを AWS DMSのターゲットとして使用し始める前に、次の前提条件を満たしていることを確認してください。

- MySQL AWS DMS 互換データベースへの読み取り/書き込み権限を持つユーザーアカウントを指定します。必要なアクセス権限を作成するには、以下のコマンドを実行します。

```
CREATE USER '<user acct>'@'%' IDENTIFIED BY '<user password>';
GRANT ALTER, CREATE, DROP, INDEX, INSERT, UPDATE, DELETE, SELECT ON <schema>.* TO
'<user acct>'@'%';
GRANT ALL PRIVILEGES ON awsdms_control.* TO '<user acct>'@'%';
```

- 全ロード移行フェーズ中、ターゲットテーブルで外部キーを無効にする必要があります。フルロード中に MySQL 互換データベースの外部キーチェックを無効にするには、 AWS DMS ターゲット

エンドポイントのコンソールの Extra connection attributes セクションに次のコマンドを追加します。

```
Initstmt=SET FOREIGN_KEY_CHECKS=0;
```

- データベースパラメータ `local_infile = 1` を設定して、 AWS DMS がターゲットデータベースにデータをロードできるようにします。

MySQL 互換データベースをターゲットとして使用する場合の制限事項 AWS Database Migration Service

MySQL データベースをターゲットとして使用する場合、 AWS DMS は以下をサポートしません。

- データ定義言語 (DDL) ステートメント: TRUNCATE PARTITION、DROP TABLE、RENAME TABLE。
- `ALTER TABLE table_name ADD COLUMN column_name` ステートメントを使用して、テーブルの先頭または中間に列を追加します。
- フルロードタスクで MySQL 互換のターゲットにデータをロードする場合、 AWS DMS タスクログの制約によるエラーは報告されません。これにより、重複キーエラーやレコード数との不一致が発生する可能性があります。これは、MySQL `LOAD DATA` によるコマンドでのローカルデータの処理方法によるものです。フルロードフェーズ時は、必ず次を実行します。
  - Constraint を無効にする
  - AWS DMS 検証を使用してデータに一貫性があることを確認します。
- 列の値を既存の値に更新すると、MySQL 互換データベースにより `0 rows affected` 警告が返されます。この動作は技術的にはエラーではありませんが、他のデータベースエンジンによって状況が処理される方法とは異なります。たとえば、Oracle は 1 行の更新を実行します。MySQL 互換データベースの場合、awsdms_apply_exceptions AWS DMS コントロールテーブルにエントリを生成し、次の警告を記録します。

```
Some changes from the source database had no impact when applied to
the target database. See awsdms_apply_exceptions table for details.
```

- MySQL バージョン 5.7 と互換性がある Amazon Aurora バージョン 2 のターゲットとして Aurora Serverless が利用可能です。(MySQL 5.7 と互換性がある Aurora Serverless を使用できるように

するには、Aurora MySQL バージョン 2.07.1 を選択します。) Aurora サーバーレスの詳細については、Amazon Aurora Aurora ユーザーガイドの「Aurora サーバーレス v2 の使用」を参照してください。

- AWS DMS インスタンスが書き込み可能モード、read_onlyinnodb_read_onlyつまりとパラメータがまたはに設定されている場合を除き、Aurora または Amazon RDS のリーダーエンドポイントの使用はサポートされていません。0 OFFAmazon RDS と Aurora をターゲットとして使用する方法の詳細については、次を参照してください。
  - 接続先の DB インスタンスの確認
  - MySQL でのリードレプリカの更新

MySQL 互換データベースをターゲットとして使用する場合のエンドポイント設定 AWS DMS

エンドポイントの設定を使用して、追加の接続属性の使用する場合と同様に、ターゲットの MySQL 互換データベースを設定できます。ターゲットエンドポイントを作成するときに、 AWS DMS コンソールを使用するか、JSON create-endpoint AWS CLI構文のコマンドを使用して設定を指定します。--my-sql-settings '{"*EndpointSetting*": *"value"*, *...*}'

次の表は、MySQL をターゲットとして使用できるエンドポイント設定を説明しています。

名前	説明
TargetDbType	ソーステーブルを移行するターゲット上の場所 (1 つのデータベースか複数のデータベースか) を指定します。指定する場合はSPECIFIC_DATABASE 、 AWS CLI または を使用するときにデータベース名を指定する必要があります AWS Management Console。 デフォルト値: MULTIPLE_DATABASES 有効な値: {SPECIFIC_DATABASE , MULTIPLE_DATABASES } 例: --my-sql-settings '{"TargetDbType": "MULTIPLE_DATABASES"}'
ParallelLoadThreads	データを MySQL 互換ターゲットデータベースにロードする際のパフォーマンスが向上します。データを MySQL

名前	説明
	互換ターゲットデータベースにロードする際に使用するスレッドの数を指定します。スレッドごとに別個の接続が必要になるため、スレッド数を大きく設定するとデータベースのパフォーマンスに悪影響を生じる場合があります。 デフォルト値： 1 有効な値: 1〜5 例: `--my-sql-settings '{"ParallelLoadThreads": 1}'`
`AfterConnectScript`	AWS DMS がエンドポイントに接続した直後に実行するスクリプトを指定します。 例えば、MySQL 互換のターゲットが受信するステートメントをデータベースのデフォルトのコンパイル済み文字セットである latin1 文字セットに変換するように指定できる。このパラメータでは通常、UTF8 クライアントからの変換時にパフォーマンスが向上します。 例: `--my-sql-settings '{"AfterConnectScript": "SET character_set_connection='latin1'"}'`
`MaxFileSize`	MySQL 互換データベースへのデータ転送に使用される .csv ファイルの最大サイズ (KB 単位) を指定します。 デフォルト値: 32768 KB (32 MB) 有効な値: 1 ~ 1,048,576 `--my-sql-settings '{"MaxFileSize": 512}'`

名前	説明
CleanSrcMetadataOnMismatch	不一致が発生すると、レプリケーションインスタンスのテーブルメタデータ情報をクリーンアップして再作成します。例: テーブルの ALTER DDL ステートメントを実行する場合、レプリケーションインスタンスにキャッシュされているテーブルに関する情報が変更される場合があります。Boolean。 デフォルト値: false 例: --my-sql-settings '{"CleanSrcMetadataOnMismatch": false}'

追加の接続属性を使用して、MySQL 互換のターゲットデータベースを設定することもできます。

次の表は、MySQL をターゲットとして使用できる追加の接続属性を説明しています。

名前	説明
Initstmt=SET FOREIGN_KEY_CHECKS=0;	外部キーチェックを無効にします。 例: --extra-connection-attributes "Initstmt=SET FOREIGN_KEY_CHECKS=0;"
Initstmt=SET time_zone	ターゲット MySQL 互換データベースのタイムゾーンを指定します。 デフォルト値: UTC 有効値:ターゲットの MySQL データベースで使用可能なタイムゾーン名。 例: --extra-connection-attributes "Initstmt=SET time_zone= *US/Pacific* ;"

別の方法として、--my-sql-settings コマンドの AfterConnectScript パラメータを使用して外部キーチェックを無効にし、データベースのタイムゾーンを指定することもできる。

MySQL のターゲットデータ型

次の表は、使用時にサポートされる MySQL AWS DMS データベースターゲットのデータ型と、AWS DMS データ型からのデフォルトのマッピングを示しています。

AWS DMS データ型に関する追加情報については、を参照してくださいAWS Database Migration Service のデータ型。

AWS DMS データタイプ	MySQL のデータ型
BOOLEAN	BOOLEAN
BYTES	長さが 1 ～ 65,535 の場合、VARBINARY (長さ) を使用します。 長さが 65,536 ～ 2,147,483,647 の場合、LONGLOB を使用します。
DATE	DATE
TIME	TIME
タイムスタンプ	"スケールが 0 以上、6 以下の場合: DATETIME (Scale) スケールが 7 以上、9 以下の場合: VARCHAR (37)"
INT1	TINYINT
INT2	SMALLINT
INT4	INTEGER
INT8	BIGINT
NUMERIC	DECIMAL (p,s)
REAL4	FLOAT
REAL8	DOUBLE PRECISION

AWS DMS データタイプ	MySQL のデータ型
STRING	長さが 1 ~ 21,845 の場合、VARCHAR (長さ) を使用します。 長さが 21,846 ~ 2,147,483,647 の場合、LONGTEXT を使用します。
UINT1	UNSIGNED TINYINT
UINT2	UNSIGNED SMALLINT
UINT4	UNSIGNED INTEGER
UINT8	UNSIGNED BIGINT
WSTRING	長さが 1 ~ 32,767 の場合、VARCHAR (長さ) を使用します。 長さが 32,768 ~ 2,147,483,647 の場合、LONGTEXT を使用します。
BLOB	長さが 1 ~ 65,535 の場合、BLOB を使用します。 長さが 65,536 ~ 2,147,483,647 の場合、LONGBLOB を使用します。 長さが 0 の場合、LONGBLOB (LOB を完全にサポート) を使用します。

AWS DMS データタイプ	MySQL のデータ型
NCLOB	長さが 1 ~ 65,535 の場合、TEXT を使用します。 長さが 65,536 ~ 2,147,483,647 の場合、CHARACTER SET が ucs2 の LONGTEXT を使用します。 長さが 0 の場合、ucs2 が CHARACTER SET の LONGTEXT (LOB を完全にサポート) を使用します。
CLOB	長さが 1 ~ 65,535 の場合、TEXT を使用します。 長さが 65,536 ~ 2147483647 の場合、LONGTEXT を使用します。 長さが 0 の場合、LONGTEXT (LOB を完全にサポート) を使用します。

AWS Database Migration Service のターゲットとしての Amazon Redshift データベースの使用

AWS Database Migration Service を使用して Amazon Redshift データベースにデータを移行できます。Amazon Redshift は、 クラウド内でのフルマネージド型、ペタバイト規模のデータウェアハウスサービスです。ターゲットとして Amazon Redshift データベースを使用すると、サポートされている他のすべてのソースデータベースからデータを移行できます。

AWS DMS のターゲットとして Amazon Redshift Serverless を使用できます。詳細については、「Amazon Redshift Serverless をターゲットする AWS DMS の使用」を参照してください。

Amazon Redshift クラスターは、レプリケーション インスタンスと同じ AWS アカウントと同じ AWS リージョンに存在している必要があります。

Amazon Redshift へのデータベース移行中、AWS DMS はまずデータを Amazon S3 バケットに移動します。ファイルが Amazon S3 バケットに移動されると、AWS DMS はそれらのファイルを Amazon Redshift データウェアハウス内の適切なテーブルに転送します。AWS DMS は S3 バケッ

トを Amazon Redshift データベースと同じ AWS リージョンに作成します。AWS DMS レプリケーション インスタンスはその同じ AWS リージョンに存在している必要があります。

AWS CLI または DMS API を使用してデータを Amazon Redshift に移行する場合、S3 アクセスを許可するように AWS Identity and Access Management (IAM) ロールを設定します。この IAM ロールの作成に関する詳細については、「AWS CLI および AWS DMS API で使用する IAM ロールの作成」をご参照ください。

Amazon Redshift エンドポイントは、以下の完全なオートメーションを行います。

- スキーマ生成およびデータ型マッピング
- ソースデータベーステーブルの全ロード
- ソーステーブルに加えられた変更の増分ロード
- ソーステーブルに加えられたスキーマ変更のデータ定義言語 (DDL) での適用
- 全ロードプロセスと変更データキャプチャ (CDC) プロセスの間の同期

AWS Database Migration Service では、全ロードオペレーションと変更処理オペレーションの両方がサポートされています。AWS DMS は、ソースデータベースからデータを読み取り、一連のカンマ区切り値 (.csv) ファイルを作成します。全ロードオペレーションの場合、AWS DMS はテーブルごとにファイルを作成し、次に、AWS DMS は各テーブルのテーブルファイルを Amazon S3 内の別個のフォルダにコピーします。ファイルが Amazon S3 にアップロードされると、AWS DMS がコピーコマンドを送信し、ファイル内のデータが Amazon Redshift にコピーされます。変更処理オペレーションの場合、AWS DMS は差分変更を .csv ファイルにコピーします。その後、AWS DMS は差分変更ファイルを Amazon S3 にアップロードし、データを Amazon Redshift にコピーします。

AWS DMS のターゲットとしての Amazon Redshift の使用の詳細については、以下のセクションをご参照ください：

トピック

- AWS Database Migration Service のターゲットとして Amazon Redshift データベースを使用する場合の前提条件
- ターゲットとして Redshift を使用するために必要な権限
- AWS Database Migration Service のターゲットとして Amazon Redshift を使用する場合の制限
- AWS Database Migration Service のターゲットとしての Amazon Redshift データベースの設定
- AWS Database Migration Service のターゲットとしての Amazon Redshift での VPC ルーティングの使用

- Amazon Redshift ターゲットデータを暗号化する AWS KMS キーの作成と使用
- AWS DMS のターゲットとして Amazon Redshift を使用する場合のエンドポイントの設定
- データ暗号化キーと中間ストレージとしての Amazon S3 バケットの使用
- Amazon Redshift のマルチスレッドタスク設定
- Amazon Redshift のターゲットデータ型
- Amazon Redshift Serverless をターゲットする AWS DMS の使用

AWS Database Migration Service のターゲットとして Amazon Redshift データベースを使用する場合の前提条件

以下のリストでは、データ移行のターゲットとして Amazon Redshift を使用する場合に必要な前提条件について説明します。

- Amazon Redshift クラスターを起動するには、AWS マネジメントコンソールを使用します。AWS アカウントと Amazon Redshift クラスターに関するパスワード、ユーザー名、データベース名など基本的な情報を書き留めます。これらの値は、Amazon Redshift ターゲット エンドポイントを作成するときに必要になります。
- Amazon Redshift クラスターは、レプリケーション インスタンスと同じ AWS アカウントと同じ AWS リージョンに存在している必要があります。
- AWS DMS レプリケーション インスタンスには、クラスターで使用される Amazon Redshift エンドポイント (ホスト名とポート) へのネットワーク接続が必要です。
- AWS DMS は Amazon S3 バケットを使用してデータを Amazon Redshift データベースに転送します。AWS DMS がバケットを作成できるようにするため、コンソールは IAM ロール `dms-access-for-endpoint` を使用します。AWS CLI または DMS API を使用して、ターゲットデータベースとして Amazon Redshiftを使用したデータベース移行を作成する場合、この IAM ロールを作成する必要があります。このロールの作成に関する詳細については、「AWS CLI および AWS DMS API で使用する IAM ロールの作成」をご参照ください。
- AWS DMS は、ターゲットの Amazon Redshift インスタンスで BLOB、および CLOB、NをVARCHAR に変換します。Amazon Redshift では、64 KB を超える VARCHAR のデータ型がサポートされていないため、従来の LOB を Amazon Redshift に保存することはできません。
- `true` にターゲット メタデータタスクの設定 BatchApplyEnabled を設定し、AWS DMS が CDC 中に Amazon Redshift ターゲットテーブルへの変更を処理できるようにします。ソーステーブルとターゲットテーブルの両方にプライマリキーが必要です。プライマリキーがない場合、変更はステートメントによって適用されます。また、ターゲットレイテンシーを引き起こし、クラスターコ

ミットキューに影響を与えるため、CDC 中のタスクのパフォーマンスに悪影響を及ぼす可能性があります。

ターゲットとして Redshift を使用するために必要な権限

GRANT コマンドを使用して、ユーザーまたはユーザー グループのアクセス権限を定義します。権限には、テーブルとビューのデータの読み取り、データの書き込み、テーブルの作成など、アクセスオプションが含まれます。Amazon Redshift での GRANT 使用についての詳細は、Amazon Redshift データベース デベロッパー ガイド の「GRANT」をご参照ください。

Amazon Redshift テーブルおよびビューに対するテーブルまたは、データベース、スキーマ、関数、プロシージャ、言語レベルの権限に対する特定の権限を付与する構文を次に示します。

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | REFERENCES } [,...] | ALL
 [ PRIVILEGES ] }
    ON { [ TABLE ] table_name [, ...] | ALL TABLES IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | GROUP group_name | PUBLIC } [, ...]

GRANT { { CREATE | TEMPORARY | TEMP } [,...] | ALL [ PRIVILEGES ] }
    ON DATABASE db_name [, ...]
    TO { username [ WITH GRANT OPTION ] | GROUP group_name | PUBLIC } [, ...]

GRANT { { CREATE | USAGE } [,...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schema_name [, ...]
    TO { username [ WITH GRANT OPTION ] | GROUP group_name | PUBLIC } [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { FUNCTION function_name ( [ [ argname ] argtype [, ...] ] ) [, ...] | ALL
 FUNCTIONS IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | GROUP group_name | PUBLIC } [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { PROCEDURE procedure_name ( [ [ argname ] argtype [, ...] ] ) [, ...] | ALL
 PROCEDURES IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | GROUP group_name | PUBLIC } [, ...]

GRANT USAGE
    ON LANGUAGE language_name [, ...]
    TO { username [ WITH GRANT OPTION ] | GROUP group_name | PUBLIC } [, ...]
```

Amazon Redshift テーブルとビューに対する列レベルの権限の構文を次に示します。

```
GRANT { { SELECT | UPDATE } ( column_name [, ...] ) [, ...] | ALL [ PRIVILEGES ]
 ( column_name [,...] ) }
     ON { [ TABLE ] table_name [, ...] }
     TO { username | GROUP group_name | PUBLIC } [, ...]
```

次に、指定されたロールを持つユーザーおよびグループに付与される ASSUMEROLE 権限の構文を示します。

```
GRANT ASSUMEROLE
    ON { 'iam_role' [, ...] | ALL }
    TO { username | GROUP group_name | PUBLIC } [, ...]
    FOR { ALL | COPY | UNLOAD } [, ...]
```

AWS Database Migration Service のターゲットとして Amazon Redshift を使用する場合の制限

Amazon Redshift データベースをターゲットとして使用する場合、次の制限が適用されます。

- Amazon Redshift ターゲットの中間ストレージとして使用する S3 バケットのバージョニングは有効にしないでください。S3 のバージョニングが必要な場合は、ライフサイクルポリシーを使用して古いバージョンを積極的に削除します。これを行わないと、S3 `list-object` コールのタイムアウトが原因でエンドポイント接続テストが失敗する可能性があります。S3 バケットのライフサイクルポリシーを作成するには、「ストレージのライフサイクルの管理」を参照してください。S3 オブジェクトのバージョンを削除するには、「バージョニングが有効なバケットからのオブジェクトバージョンの削除」を参照してください。
- 以下の DDL はサポートされていません。

```
ALTER TABLE table name MODIFY COLUMN column name data type;
```

- AWS DMS は、名前がアンダースコア (_) で始まるスキーマへの変更を移行またはレプリケートできません。名前がアンダースコアで始まるスキーマがある場合は、マッピング変換を使用してターゲットでスキーマの名前を変更してください。
- Amazon Redshift は 64 KB より大きい VARCHAR をサポートしていません。従来のデータベースからの LOB を Amazon Redshift に保存することはできません。

- 複数の列のプライマリキーを持つテーブルへの DELETE ステートメントの適用は、プライマリキーの列名で予約語が使用されている場合にはサポートされません。Amazon Redshift の予約語の一覧はここをご参照ください。
- ソースシステムがソーステーブルのプライマリ キーに対して UPDATE 操作を実行すると、パフォーマンスの問題が発生することがあります。これらのパフォーマンスの問題は、ターゲットに変更を適用するときに発生します。これは、UPDATE (および DELETE) オペレーションは、ターゲット行を識別するためにプライマリ キーの値に依存するためです。ソーステーブルのプライマリ キーを更新すると、タスクログに次のようなメッセージが表示されます。

```
Update on table 1 changes PK to a PK that was previously updated in the same bulk
 update.
```

- DMS は Redshift クラスターのエンドポイントを構成するときにカスタム DNS 名をサポートしないため、Amazon が提供する DNS 名を使用する必要があります。Amazon Redshift クラスターはレプリケーション インスタンスと同じ AWS アカウントとリージョンにある必要があるため、カスタム DNS エンドポイントを使用すると検証が失敗します。
- Amazon Redshift には、デフォルトで 4 時間のアイドルセッションタイムアウトがあります。DMS レプリケーションタスクでアクティビティがない場合、Redshift は 4 時間後にセッションを切断します。DMS が接続できないためにエラーが発生し、再起動が必要になる場合があります。回避策として、DMS レプリケーションユーザーのセッションタイムアウト制限を 4 時間以上に設定します。または、「Amazon Redshift データベースデベロッパーガイド」の ALTER USER の説明を参照してください。
- AWS DMS がプライマリキーや一意のキーを使用せずにソーステーブルデータをレプリケートすると、CDC レイテンシーが増大し、パフォーマンスが許容できないレベルとなる可能性があります。

AWS Database Migration Service のターゲットとしての Amazon Redshift データベースの設定

Amazon Redshift インスタンスを使用できるように AWS Database Migration Service を設定する必要があります。以下の表では、Amazon Redshift エンドポイントに使用できる設定プロパティについて説明します。

プロパティ	説明
server	使用している Amazon Redshift クラスターの名前。

プロパティ	説明
ポート	Amazon Redshift のポート番号。デフォルト値は 5439 です。
username	登録済みユーザーの Amazon Redshift ユーザー名。
password	username プロパティで指定されたユーザーのパスワード。
データベース	使用する Amazon Redshift データウェアハウス (サービス) の名前。

Amazon Redshift エンドポイントに追加の接続文字列属性を追加する場合、`maxFileSize` 属性と `fileTransferUploadStreams` 属性を指定できます。これらの属性の詳細については、「AWS DMS のターゲットとして Amazon Redshift を使用する場合のエンドポイントの設定」をご参照ください。

AWS Database Migration Service のターゲットとしての Amazon Redshift での VPC ルーティングの使用

Amazon Redshift ターゲットで拡張された VPC のルーティングを使用すると、Amazon Redshift クラスターとデータリポジトリ間のすべての COPY トラフィックは VPC を介します。拡張された VPC ルーティングは、他のリソースに Amazon Redshift がアクセスする方法に影響を与えるため、VPC を正しく設定していないと、COPY コマンドが失敗することがあります。

AWS DMS がこの動作の影響を受けることがあるのは、COPY コマンドを使用して S3 内のデータを Amazon Redshift クラスターに移動するためです。

以下に示しているのは、AWS DMS が Amazon Redshift ターゲットにデータをロードするステップです。

1. AWS DMS がソースからレプリケーションサーバー上の .csv ファイルにデータをコピーします。
2. AWS DMS が AWS SDK を使用してアカウントの S3 バケットに .csv ファイルをコピーします。
3. AWS DMS がその後、Amazon Redshift で COPY コマンドを使用して、S3 内の .csv ファイルから Amazon Redshift 内の該当するテーブルにデータをコピーします。

拡張された VPC ルーティングが有効でない場合、Amazon Redshift は AWS ネットワーク内のその他のサービスなどへのトラフィックを、インターネット経由でルーティングします。この機能が有効でない場合は、ネットワークパスを設定する必要はありません。この機能が有効な場合は、クラスターの VPC とデータリソースとの間のネットワークパスを別に作成する必要があります。必要な設

定の詳細については、Amazon Redshift のドキュメントの「拡張された VPC ルーティング」をご参照ください。

Amazon Redshift ターゲットデータを暗号化する AWS KMS キーの作成と使用

ターゲットデータが Amazon Redshift にコピーされる前に、Amazon S3 にプッシュされるターゲットデータを暗号化できます。このため、カスタム AWS KMS キーを作成して使用できます。Amazon Redshift ターゲット エンドポイント作成時に次のいずれかのメカニズムを使用して、ターゲットデータを暗号化するために作成したキーを使用できます。

- AWS CLI を使用して `create-endpoint` コマンドを実行するとき、次のオプションを使用します。

  ```
  --redshift-settings '{"EncryptionMode": "SSE_KMS", "ServerSideEncryptionKmsKeyId":
   "your-kms-key-ARN"}'
  ```

  ここで *`your-kms-key-ARN`* とは、KMS キーの Amazon リソースネーム (ARN) です。詳細については、「データ暗号化キーと中間ストレージとしての Amazon S3 バケットの使用」をご参照ください。
- 値 (`SSE_KMS`) に追加の接続属性 (`encryptionMode`) を設定し、KMS キーの ARN に追加の接続属性 (`serverSideEncryptionKmsKeyId`) を設定します。詳細については、「AWS DMS のターゲットとして Amazon Redshift を使用する場合のエンドポイントの設定」をご参照ください。

KMS キーを使用して Amazon Redshift ターゲットデータを暗号化するには、Amazon Redshift データにアクセスする権限がある AWS Identity and Access Management (IAM) ロールが必要です。次に、この IAM ロールは作成した暗号化キーに添付されるポリシー (キーポリシー) にアクセスします。これは、IAM コンソールで次を作成して実行します：

- AWS が管理するポリシーがある IAM ロールです。
- このロールを参照するキーポリシーがある KMS キーです。

以下の手順でこれを行う方法について説明します。

必要な AWS 管理ポリシーで 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. ナビゲーションペインで [Roles] (ロール) を選択します。[ロール] ページが開きます。
3. [Create role] (ロールの作成) を選択します。[Create role] (ロールの作成) ページが開きます。
4. 信頼されたエンティティとして選択された AWS サービス で、ロールを使用するサービスとして DMS を選択します。
5. [Next: Permissions] (次へ: アクセス許可) を選択します。[Attach permissions policies (アクセス権限ポリシーをアタッチする)] ページが表示されます。
6. `AmazonDMSRedshiftS3Role` ポリシーを見つけて選択します。
7. [Next:Tags](次へ: タグ) を選択します。タグの追加 ページが表示されます。ここでは、任意のタグを追加することができます。
8. [Next: Review (次の手順: 確認)] を選択し、結果を確認します。
9. 必要な設定が構成されている場合にはロールに名前 (`DMS-Redshift-endpoint-access-role` など) および追加の説明を入力し、[Create role] (ロールの作成) を選択します。[ロール] ページが開き、ロールが作成されたことを示すメッセージが表示されます。

`DMS-Redshift-endpoint-access-role` を使用したデータ移行のターゲットとして使用するように Amazon Redshiftデータベースを設定します。

この IAM ロールを参照するキーポリシーを持つ AWS KMS 暗号化キーを作成するには

> **Note**
>
> AWS DMS と AWS KMS 暗号化キーの連携について詳しくは、「暗号化キーの設定と AWS KMS アクセス許可の指定」をご参照ください。

1. AWS Management Console にサインインし、AWS Key Management Service (AWS KMS) コンソール (https://console.aws.amazon.com/kms) を開きます。
2. AWS リージョン を変更するには、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。
4. [Create key] (キーの作成) を選択します。[キーの設定] ページが開きます。
5. [キーの種類] で、[対称] を選択します。

> **Note**
>
> このキーを作成する場合、Amazon Redshift などすべての AWS サービスでは対称暗号化キーのみを使用できるため、作成できるのは対称キーのみです。

6. [アドバンスドオプション] を選択します。[キーマテリアルのオリジン] で、[KMS] が選択されていることを確認し、[次へ] を選択します。[ラベルの追加] ページが開きます。
7. [エイリアスと説明の作成] で、キーのエイリアス (`DMS-Redshift-endpoint-encryption-key` など) と追加の説明を入力します。
8. [タグ] で、キーを識別してその使用状況を追跡するために役立つ任意のタグを追加したら、[次へ] を選択します。[キー管理アクセス許可の定義] ページが開き、選択できるユーザーおよびロールの一覧が表示されます。
9. キーを管理するユーザーおよびロールを追加します。このユーザーとロールにキーを管理するために必要な権限があることを確認してください。
10. [キーの削除] で、キー管理者がそのキーを削除できるかどうかを選択したら、[次へ] を選択します。[キーの使用アクセス許可の定義] ページが開き、選択できる追加のユーザーおよびロールの一覧が表示されます。
11. [This account] (このアカウント) で、Amazon Redshift ターゲットに対して暗号化オペレーションを実行できるユーザーを選択します。また、[Roles] (ロール) で以前に作成したロールを選択して、`DMS-Redshift-endpoint-access-role` などの Amazon Redshift ターゲットオブジェクトを暗号化するためのアクセスを有効化します。
12. リストにない他のアカウントに同じアクセス権限を付与するには、[他の AWS アカウント] で [別の AWS アカウントを追加する]、[次へ] の順に選択します。[キーポリシーの表示と編集] ページが開き、既存の JSON に入力して表示および編集できるキーポリシーの JSON が表示されます。ここでは、前のステップで選択したロールおよびユーザー (例えば、`Admin` と `User1`) を参照するキーポリシーを表示できます。また、次の例に示すように、異なるプリンシパル (ユーザーとロール) に許可される別々のキーアクションも確認できます。

```
{
  "Id": "key-consolepolicy-3",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
```

```
        "AWS": [
          "arn:aws:iam::111122223333:root"
        ]
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow access for Key Administrators",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:role/Admin"
        ]
      },
      "Action": [
        "kms:Create*",
        "kms:Describe*",
        "kms:Enable*",
        "kms:List*",
        "kms:Put*",
        "kms:Update*",
        "kms:Revoke*",
        "kms:Disable*",
        "kms:Get*",
        "kms:Delete*",
        "kms:TagResource",
        "kms:UntagResource",
        "kms:ScheduleKeyDeletion",
        "kms:CancelKeyDeletion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:role/DMS-Redshift-endpoint-access-role",
          "arn:aws:iam::111122223333:role/Admin",
          "arn:aws:iam::111122223333:role/User1"
        ]
      },
      "Action": [
```

```
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:role/DMS-Redshift-endpoint-access-role",
          "arn:aws:iam::111122223333:role/Admin",
          "arn:aws:iam::111122223333:role/User1"
        ]
      },
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": "*",
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": true
        }
      }
    }
  ]
```

13. [終了] を選択します。[Encryption keys] (暗号化キー) ページが開き、AWS KMS key が作成されたことを示すメッセージが表示されます。

これで、指定したエイリアス (DMS-Redshift-endpoint-encryption-key など) を使用する新しい KMS キーが作成されました。このキーによって AWS DMS は Amazon Redshift ターゲットデータを暗号化できます。

AWS DMS のターゲットとして Amazon Redshift を使用する場合のエンドポイントの設定

追加の接続属性の使用と同様、エンドポイントの設定を使用して、ターゲットの Amazon Redshift データベースを設定できます。ターゲットエンドポイントを作成する際に、AWS DMS コンソールを使用するか、AWS CLI で `--redshift-settings '{"EndpointSetting": "value", ...}'` の JSON 構文を指定して `create-endpoint` コマンドを使用して設定を指定します。

次の表は、ターゲットとして Amazon Redshift を使用できるエンドポイント設定を説明しています。

名前	説明
`MaxFileSize`	Amazon Redshift へのデータ転送に使用される .csv ファイルの最大サイズ (KB 単位) を指定します。 デフォルト値: 32768 KB (32 MB) 有効な値: 1 ~ 1,048,576 例: `--redshift-settings '{"MaxFileSize": 512}'`
`FileTransferUploadStreams`	1 つのファイルをアップロードするのに使用されるスレッドの数を指定します。 デフォルト値: 10 有効な値: 1 ~ 64 例: `--redshift-settings '{"FileTransferUploadStreams": 20}'`
`Acceptanydate`	0000-00-00 などの無効な日付形式を含む、あらゆる日付形式を受け入れるかどうかを指定します。ブール値。 デフォルト値: false 有効な値: true | false

名前	説明
	例: `--redshift-settings '{"Accept anydate": true}'`
`Dateformat`	日付形式を指定します。これは、文字列入力であり、デフォルトでは空です。デフォルトの形式は、YYYY-MM-DD ですが、DD-MM-YYYY などに変更できます。日の値または時間の値で異なる形式が使用される場合、`Dateformat` パラメータとともに auto 引数を使用します。auto 引数は、`Dateformat` 文字列を使用する場合にサポートされない形式を認識します。auto キーワードでは大文字小文字を区別します。 デフォルト値: 空 有効値: "*dateformat_string* " または auto 例:`--redshift-settings '{"Dateformat": "auto"}'`
`Timeformat`	時間形式を指定します。これは、文字列入力であり、デフォルトでは空です。auto 引数は、`Timeformat` 文字列を使用する場合にサポートされない形式を認識します。日の値および時間の値でそれぞれ異なる形式が使用される場合、`Timeformat` パラメータとともに auto 引数を使用します。 デフォルト値: 10 有効値: "*Timeformat_string* " | "auto" | "epochsecs" | "epochmillisecs" 例:`--redshift-settings '{"Timeformat": "auto"}'`

名前	説明
`Emptyasnull`	AWS DMS が空の CHAR および VARCHAR フィールドを null として移行するかどうかを指定します。値が true の場合、空の CHAR および VARCHAR フィールドが null として設定されます。 デフォルト値: false 有効な値: true | false 例: `--redshift-settings '{"Emptyasnull": true}'`
`TruncateColumns`	列の仕様に合うよう、該当する文字数で列のデータを切り捨てます。データ型が VARCHAR または CHAR の列、およびサイズが 4 MB 以下の行にのみ適用されます。 デフォルト値: false 有効な値: true | false 例: `--redshift-settings '{"TruncateColumns": true}'`
`RemoveQuotes`	入力データの文字列を囲む引用符を削除します。区切り記号を含む引用符内のすべての文字は保持されます。Amazon Redshift ターゲットの引用符を削除する詳細については、「Amazon Redshift データベース デベロッパーガイド」をご参照ください。 デフォルト値: false 有効な値: true | false 例: `--redshift-settings '{"RemoveQuotes": true}'`

名前	説明
TrimBlanks	VARCHAR 文字列から末尾の空白文字を削除します。このパラメータは VARCHAR データ型の列にのみ適用されます。 デフォルト値: false 有効な値: true | false 例: `--redshift-settings '{"TrimBlanks": true}'`
EncryptionMode	データが Amazon Redshift にコピーされる前に S3 にデータをプッシュするために使用するサーバー側の暗号化モードを指定します。有効な値は、SSE_S3 (S3 サーバー側の暗号化) または SSE_KMS (KMS キーの暗号化) です。SSE_KMS を選択する場合、暗号化のために使用する KMS キーの Amazon リソースネーム (ARN) に ServerSideEncryptionKmsKeyId パラメータを設定します。 **Note** CLI modify-endpoint コマンドを使用して、既存のエンドポイントの EncryptionMode 設定値を SSE_KMS から SSE_S3 に変更することもできる。ただし、EncryptionMode の値を SSE_S3 から SSE_KMS に変えることはできない。 デフォルト値: SSE_S3 有効な値: SSE_S3 または SSE_KMS 例:`--redshift-settings '{"EncryptionMode": "SSE_S3"}'`

名前	説明
ServerSideEncryptionKmsKeyId	SSE_KMS に EncryptionMode を設定する場合、KMS キーの ARN にこのパラメータを設定します。アカウントに作成した AWS KMS キーのリスト内でキーエイリアスを選択すると、この ARN が見つかります。キーを作成するとき、特定のポリシーとロールをこのキーに関連付ける必要があります。詳細については、「Amazon Redshift ターゲットデータを暗号化する AWS KMS キーの作成と使用」を参照してください。 例: `--redshift-settings '{"ServerSideEncryptionKmsKeyId":"arn:aws:kms:us-east-1:111122223333:key/11a1a1a1-aaaa-9999-abab-2bbbbbb222a2"}'`
EnableParallelBatchInMemoryCSVFiles	EnableParallelBatchInMemoryCSVFiles 設定は、DMS がメモリではなくディスクに書き込むようにすることで、大規模なマルチスレッドのフルロードタスクのパフォーマンスを向上させる。デフォルト値は false です。
CompressCsvFiles	この属性を使用して、移行中に Amazon Redshift ターゲットに送信されるデータを圧縮する。デフォルト値は true で、圧縮はデフォルトで有効。

データ暗号化キーと中間ストレージとしての Amazon S3 バケットの使用

Amazon Redshift ターゲット エンドポイント設定を使用して、以下を設定できます：

- カスタム AWS KMS データ暗号化キー。その後、このキーを使用して、データが Amazon Redshift にコピーされる前に、Amazon S3 にプッシュされるデータを暗号化できます。
- Amazon Redshift に移行するデータ用の中間ストレージとしてのカスタム S3 バケット。
- ブール型を PostgreSQL ソースからのブール型としてマップします。ブール型はデフォルトで varchar (1) として移行されます。次の例のとおり、MapBooleanAsBoolean を指定すると、Redshift ターゲットでブール型をブール型として移行できるようになります。

```
--redshift-settings '{"MapBooleanAsBoolean": true}'
```

この設定を有効にするには、ソースエンドポイントとターゲットエンドポイントの両方でこの設定を設定する必要があることに注意します。

データ暗号化に対する KMS キー設定

次の例では、S3 にプッシュされるデータを暗号化するカスタム KMS キーの設定を示しています。開始するには、AWS CLI で次の `create-endpoint` 呼び出しを行う場合があります。

```
aws dms create-endpoint --endpoint-identifier redshift-target-endpoint --endpoint-type
 target
--engine-name redshift --username your-username --password your-password
--server-name your-server-name --port 5439 --database-name your-db-name
--redshift-settings '{"EncryptionMode": "SSE_KMS",
"ServerSideEncryptionKmsKeyId": "arn:aws:kms:us-east-1:111122223333:key/24c3c5a1-
f34a-4519-a85b-2debbef226d1"}'
```

ここでは、`--redshift-settings` オプションで指定される JSON オブジェクトは 2 つのパラメータを定義します。1 つは、値が `SSE_KMS` の `EncryptionMode` パラメータです。もう 1 つは、値が `arn:aws:kms:us-east-1:111122223333:key/24c3c5a1-f34a-4519-a85b-2debbef226d1` の `ServerSideEncryptionKmsKeyId` パラメータです。この値は、カスタム KMS キーの Amazon リソースネーム (ARN) です。

デフォルトでは、S3 データ暗号化は S3 サーバー側の暗号化を使用して行われます。前述の例の Amazon Redshift ターゲットの場合、次の例に示すように、これはその エンドポイント設定を指定することと同様です。

```
aws dms create-endpoint --endpoint-identifier redshift-target-endpoint --endpoint-type
 target
--engine-name redshift --username your-username --password your-password
--server-name your-server-name --port 5439 --database-name your-db-name
--redshift-settings '{"EncryptionMode": "SSE_S3"}'
```

S3 サーバー側の暗号化の詳細については、Amazon Simple Storage Service ユーザーガイド の「サーバー側の暗号化を使用したデータの保護」をご参照ください。

> **Note**
>
> CLI `modify-endpoint` コマンドを使用して、既存のエンドポイントの `EncryptionMode` のパラメータ値を `SSE_KMS` から `SSE_S3` に変更するすることもできます。しかし `EncryptionMode` の値を `SSE_S3` から `SSE_KMS` に変えることはできません。

Amazon S3 バケットのセットアップ

Amazon Redshift ターゲットエンドポイントにデータを移行する際、AWS DMS はデフォルトの Amazon S3 バケットを中間タスクストレージとして使用し、その後移行されたデータを Amazon Redshift にコピーします。例えば、AWS KMS データ暗号化キーを使用して Amazon Redshift ターゲットエンドポイントを作成する例では、このデフォルトの S3 バケットを使用します (データ暗号化に対する KMS キー設定 を参照)。

代わりに、この中間ストレージにカスタム S3 バケットを指定することもできます。これを行うには、AWS CLI `create-endpoint` コマンドで `--redshift-settings` オプションの値に次のパラメータを含めます。

- `BucketName` - S3 バケットストレージの名前として指定する文字列。サービスアクセスロールが `AmazonDMSRedshiftS3Role` ポリシーに基づいている場合、この値には `dms-` というプレフィックスが必要です (`dms-my-bucket-name` など)。
- `BucketFolder` - (オプション) 指定された S3 バケットのストレージフォルダー名として指定できる文字列。
- `ServiceAccessRoleArn` - S3 バケットへの管理アクセスを許可する IAM ロールの ARN。一般的に、 `AmazonDMSRedshiftS3Role` ポリシーに基づいてこのロールを作成します。例については、Amazon Redshift ターゲットデータを暗号化する AWS KMS キーの作成と使用 で、必要な AWS が管理するポリシーを持つ IAM ロールの作成手順 をご参照ください。

  > **Note**
  >
  > `create-endpoint` コマンドの `--service-access-role-arn` オプションを使用して別の IAM ロールの ARN を指定した場合、その IAM ロールオプションが優先されます。

以下の例では、これらのパラメータを使用して、AWS CLI を使用した以下の `create-endpoint` コールでカスタム Amazon S3 バケットを指定しています。

```
aws dms create-endpoint --endpoint-identifier redshift-target-endpoint --endpoint-type
 target
--engine-name redshift --username your-username --password your-password
--server-name your-server-name --port 5439 --database-name your-db-name
--redshift-settings '{"ServiceAccessRoleArn": "your-service-access-ARN",
"BucketName": "your-bucket-name", "BucketFolder": "your-bucket-folder-name"}'
```

Amazon Redshift のマルチスレッドタスク設定

マルチスレッドタスク設定を使用して Amazon Redshift ターゲット エンドポイントの全ロードと変更データ キャプチャ (CDC) タスクのパフォーマンスを向上させることができます。これを行うには、同時スレッドの数とバッファに保存するレコード数を指定できます。

Amazon Redshift でのマルチスレッド全ロードタスク設定

全負荷のパフォーマンスを促進するには、以下の `ParallelLoad*` タスク設定を使用します:

- `ParallelLoadThreads` – データレコードを Amazon Redshift ターゲットエンドポイントにプッシュするために全ロード中に DMS が使用する同時スレッドの数を指定します。デフォルト値は 0 で、最大値は 32 です。詳細については、「全ロードタスク設定」を参照してください。

  `ParallelLoadThreads` タスク設定を使用する際、`enableParallelBatchInMemoryCSVFiles` 属性を `false` に設定することができます。この属性は、DMS がメモリではなくディスクに書き込むことにより、大規模なマルチスレッドの全ロードタスクのパフォーマンスを向上させます。デフォルト値は `true` です。
- `ParallelLoadBufferSize` — Redshift ターゲットで並列ロード スレッドを使用しているときの最大データレコード要求を指定します。デフォルト値は 100 で、最大値は 1,000 です。このオプションは、ParallelLoadThreads > 1 (1 より大きい) 場合に使用することをお勧めします。

> **Note**
>
> フルロードから Amazon Redshift ターゲットエンドポイントへの `ParallelLoad*` タスク設定の使用のサポートは、AWS DMS バージョン 3.4.3 以降で利用できます。
> `ReplaceInvalidChars` Redshift エンドポイント設定は、変更データ キャプチャ (CDC) 中または並列ロードが有効な全ロード移行タスク中での使用には対応していません。並列ロードが有効でない場合、全ロード移行でサポートされます。詳細については、AWS Database Migration Service API リファレンス の [RedshiftSettings] (Redshift設定) をご参照ください。

Amazon Redshift のマルチスレッド CDC タスク設定

CDC のパフォーマンスを向上させるため、以下の `ParallelApply*` タスク設定を使用できます:

- `ParallelApplyThreads` – データレコードを Amazon Redshift ターゲット エンドポイントにプッシュするために CDC ロード中に AWS DMS が使用する同時スレッドの数を指定します。デフォルト値は 0 で、最大値は 32 です。推奨される最小値は、クラスター内のスライスの数と同数です。
- `ParallelApplyBufferSize` — Redshift ターゲットで並列適用スレッドを使用しているときの最大データレコード要求を指定します。デフォルト値は 100 で、最大値は 1,000 です。このオプションは、ParallelApplyThreads > 1 (1 より大きい) 場合に使用することをお勧めします。

  Redshift をターゲットとして最大のメリットを得るには、`ParallelApplyBufferSize` の値は `ParallelApplyThreads` の少なくとも2倍 (2倍以上) の数の数にしてください。

> **Note**
>
> CDC から Amazon Redshift ターゲットエンドポイントへの `ParallelApply*` タスク設定の使用のサポートは、AWS DMS バージョン 3.4.3 以降で利用できます。

適用される並列度のレベルは、データの転送に使用される [batch size] (バッチサイズ) と[maximum file size] (最大ファイルサイズ) 合計間の相関関係によって異なります。Redshift ターゲットでマルチスレッド CDC タスク設定を使用する場合、バッチサイズが最大ファイルサイズに対して大きい場合にメリットが得られます。例えば、エンドポイントとタスクの設定の次の組み合わせを使用して、最適なパフォーマンスをチューニングできます。

```
// Redshift endpoint setting

        MaxFileSize=250000;

// Task settings

        BatchApplyEnabled=true;
        BatchSplitSize =8000;
        BatchApplyTimeoutMax =1800;
        BatchApplyTimeoutMin =1800;
        ParallelApplyThreads=32;
        ParallelApplyBufferSize=100;
```

上記の例の設定を使用すると、重いトランザクションワークロードがある場合、最大ファイルサイズ 250 MB で 32 の並列スレッドを利用し、1,800 秒でフルになる 8,000 レコードのバッチバッファーを最大限に活用できます。

詳細については、「変更処理のチューニング設定」を参照してください。

> **Note**
>
> Redshift クラスターへの継続的レプリケーション中に実行される DMS クエリは、実行中の他のアプリケーション クエリと同じ WLM (ワークロード管理) キューを共有できます。そのため、Redshift ターゲットへの継続的なレプリケーション中にパフォーマンスに影響を与えるように WLM プロパティを適切に設定することを検討してください。例えば、他のパラレル ETL クエリが実行されている場合、DMS の実行が遅くなり、パフォーマンスは向上しなくなります。

Amazon Redshift のターゲットデータ型

AWS DMS の Amazon Redshift エンドポイントでは、Amazon Redshift のほとんどのデータ型がサポートされます。以下の表に、AWS DMS を使用する場合にサポートされる Amazon Redshift のターゲットデータ型と、AWS DMS のデータ型からのデフォルトマッピングを示します。

AWS DMS のデータ型の詳細については、「AWS Database Migration Service のデータ型」をご参照ください。

AWS DMS データ型	Amazon Redshift のデータ型
BOOLEAN	BOOL
BYTES	VARCHAR (長さ)
DATE	DATE
TIME	VARCHAR(20)
DATETIME	位取り => 0 かつ =< 6 の場合、Redshift ターゲット列のデータ型に応じて、次のいずれかになる。

AWS DMS データ型	Amazon Redshift のデータ型
	タイムスタンプ (s) TIMESTAMPTZ (s) — ソースタイムスタンプにゾーンオフセットが含まれている場合 (SQL Server や Oracle など)、insert または update 時に UTC に変換される。オフセットが含まれていない場合は、時刻は既に UTC で考慮される。 スケールが 7 以上、9 以下の場合 VARCHAR(37)
INT1	INT2
INT2	INT2
INT4	INT4
INT8	INT8
NUMERIC	スケールが 0 以上、37 以下の場合 NUMERIC (p,s) スケールが 38 以上、127 以下の場合 VARCHAR (長さ)
REAL4	FLOAT4
REAL8	FLOAT8
STRING	長さが 1 ~ 65,535 の場合、VARCHAR (バイト単位の長さ) を使用します 長さが 65,536 ~ 2,147,483,647 の場合、VARCHAR (65535) を使用します

AWS DMS データ型	Amazon Redshift のデータ型
UINT1	INT2
UINT2	INT2
UINT4	INT4
UINT8	NUMERIC (20,0)
WSTRING	長さが 1 ~ 65,535 の場合、NVARCHAR (バイト単位の長さ) を使用します 長さが 65,536 ~ 2,147,483,647 の場合、NVARCHAR (65535) を使用します
BLOB	VARCHAR (最大 LOB サイズ *2) 最大 LOB サイズが 31 KB を超えることはできません。Amazon Redshift は 64 KB より大きい VARCHAR をサポートしていません。
NCLOB	NVARCHAR (最大 LOB サイズ) 最大 LOB サイズが 63 KB を超えることはできません。Amazon Redshift は 64 KB より大きい VARCHAR をサポートしていません。
CLOB	VARCHAR (最大 LOB サイズ) 最大 LOB サイズが 63 KB を超えることはできません。Amazon Redshift は 64 KB より大きい VARCHAR をサポートしていません。

Amazon Redshift Serverless をターゲットする AWS DMS の使用

AWS DMS は、ターゲットエンドポイントとして Amazon Redshift Serverless の使用をサポートします。Amazon Redshift Serverless の使用の詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift Serverless」を参照してください。

このトピックでは、AWS DMS で Amazon Redshift Serverless のエンドポイントを使用する方法について説明します。

> **Note**
>
> Amazon Redshift Serverless エンドポイントを作成する場合、RedshiftSettings エンドポイント設定の [DatabaseName] フィールドに Amazon Redshift データウェアハウス名またはワークグループエンドポイント名を使用します。[ServerName] フィールドには、サーバーレスクラスターの [ワークグループ] ページに表示されるエンドポイントの値を使用します (`default-workgroup.093291321484.us-east-1.redshift-serverless.amazonaws.com` など)。エンドポイント作成の詳細については、「ソースおよびターゲットエンドポイントの作成」を参照してください。ワークグループエンドポイントの詳細については、「Amazon Redshift Serverless への接続」を参照してください。

Amazon Redshift Serverless をターゲットとする信頼ポリシー

Amazon Redshift Serverless をターゲットエンドポイントとして使用する場合、次の強調表示されたセクションを信頼ポリシーに追加する必要があります。このポリシーは、`dms-access-for-endpoint` ロールにアタッチされます。

```
{
    "PolicyVersion": {
        "CreateDate": "2016-05-23T16:29:57Z",
        "VersionId": "v3",
        "Document": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": [
                    "ec2:CreateNetworkInterface",
                    "ec2:DescribeAvailabilityZones",
                    "ec2:DescribeInternetGateways",
                    "ec2:DescribeSecurityGroups",
                    "ec2:DescribeSubnets",
                    "ec2:DescribeVpcs",
                    "ec2:DeleteNetworkInterface",
                    "ec2:ModifyNetworkInterfaceAttribute"
                ],
                "Resource": "arn:aws:service:region:account:resourcetype/id",
                "Effect": "Allow"
```

```
            },
            {
                "Sid": "",
                "Effect": "Allow",
                "Principal": {
                    "Service": "redshift-serverless.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    },
    "IsDefaultVersion": true
    }
}
```

AWS DMS での信頼ポリシーの使用方法の詳細については、「AWS CLI および AWS DMS API で使用する IAM ロールの作成」を参照してください。

Amazon Redshift Serverless をターゲットとして使用する場合の制限

Redshift Serverless をターゲットとして使用する場合、次の制限があります。

- AWS DMS は、Amazon Redshift Serverless をサポートするリージョンのエンドポイントとしてのみ Amazon Redshift Serverless をサポートします。Amazon Redshift Serverless をサポートしているリージョンについては、「AWS 全般のリファレンス」の「Amazon Redshift エンドポイントとクォータ」トピックの「Redshift Serverless API」を参照してください。
- 拡張 VPC ルーティングを使用する場合は、必ず Redshift Serverless クラスターまたは Redshift プロビジョンドクラスターと同じ VPC に Amazon S3 エンドポイントを作成します。詳細については、「AWS Database Migration Service のターゲットとしての Amazon Redshift での VPC ルーティングの使用」を参照してください。
- AWS DMS は Amazon Redshift Serverless をターゲットとしてはサポートしていません。

AWS Database Migration Service のターゲットとしての SAP ASE データベースの使用

AWS DMS を使用して、SAP Adaptive Server Enterprise (ASE) データベース (旧名 Sybase) にサポートされる任意のデータベースからデータを移行できます。

AWS DMS がターゲットとしてサポートする SAP ASE のバージョンについては、「のターゲット AWS DMS」を参照してください。

SAP ASE データベースを AWS Database Migration Service のターゲットとして使用するための前提条件

AWS DMS のターゲットとして SAP ASE データベースの使用を開始する前に、次の前提条件を満たしていることを確認します。

- AWS DMS ユーザーに SAP ASE アカウントのアクセス権限を付与します。このユーザーには、SAP ASE データベースでの読み取り/書き込み権限が必要です。
- 場合によっては、非ラテン文字 (中国語など) 向けに設定された、Microsoft Windows の Amazon EC2 インスタンスで、SAP ASE バージョン 15.7 にレプリケートすることがあります。このような場合、AWS DMS では、ターゲットの SAP ASE マシンに SAP ASE 15.7 SP121 をインストールする必要があります。

SAP ASE データベースを AWS DMS のターゲットとして使用するための制限

SAP ASE データベースを AWS DMS のターゲットとして使用する場合、以下の制限が適用されます。

- AWS DMS は、次のデータ型のフィールドがあるテーブルをサポートしません。このようなデータ型の列をレプリケートすると、NULL 列が生成されます。
  - ユーザー定義のデータ型 (UDT)

SAP ASE を AWS DMS のターゲットとして使用する場合のエンドポイントの設定

追加の接続属性の使用と同様、エンドポイントの設定を使用して、ターゲットの SAP ASE データベースを設定できます。ターゲットエンドポイントを作成する際に、AWS DMS コンソールを使用するか、AWS CLI で `--sybase-settings '{"EndpointSetting": "value", ...}'` の JSON 構文を指定して `create-endpoint` コマンドを使用して設定を指定します。

次の表は、ターゲットとして SAP ASE を使用できるエンドポイント設定を説明しています。

名前	説明
`Driver`	ASE 15.7 以降のバージョンで TLS を使用する場合は、この属性を設定する。 デフォルト値: `Adaptive Server Enterprise`

名前	説明
	例: `driver=Adaptive Server Enterprise 16.03.06;` 有効値: `Adaptive Server Enterprise 16.03.06`
`AdditionalConnectionProperties`	指定する任意の追加の ODBC 接続パラメータ

ターゲットの SAP ASE のデータ型

次の表は、AWS DMS を使用する場合にサポートされるターゲットの SAP ASE データベースのデータ型と、AWS DMS のデータ型からのデフォルトマッピングを説明しています。

AWS DMS のデータ型の詳細については、「AWS Database Migration Service のデータ型」を参照してください。

AWS DMS のデータ型	SAP ASE のデータ型
BOOLEAN	BIT
BYTES	VARBINARY (長さ)
DATE	DATE
TIME	TIME
TIMESTAMP	位取り => 0 かつ =< 6 の場合: BIGDATETIME 位取り => 7 かつ =< 9 の場合: VARCHAR (37)
INT1	TINYINT
INT2	SMALLINT
INT4	INTEGER
INT8	BIGINT

AWS DMS のデータ型	SAP ASE のデータ型
NUMERIC	NUMERIC (p,s)
REAL4	REAL
REAL8	DOUBLE PRECISION
STRING	VARCHAR (長さ)
UINT1	TINYINT
UINT2	UNSIGNED SMALLINT
UINT4	UNSIGNED INTEGER
UINT8	UNSIGNED BIGINT
WSTRING	VARCHAR (長さ)
BLOB	IMAGE
CLOB	UNITEXT
NCLOB	TEXT

AWS Database Migration Service のターゲットに Amazon S3 を使用する

サポートされるデータベースのソースから AWS DMS を使用することにより、データを Amazon S3 に移行できます。AWS DMS タスクのターゲットとして Amazon S3 を使用する場合、全ロードと変更データ キャプチャ (CDC) データの両方はデフォルトでカンマ区切り値 (.csv) 形式で書き込まれます。よりコンパクトなストレージと高速なクエリオプションにおいては、Apache Parquet (.parquet) 形式でデータを書き込むオプションもあります。

AWS DMS は、.csv ファイルについて LOAD00001.csv、LOAD00002...、LOAD00009、LOAD0000A など、増分の 16 進数カウンタを使用して全ロード時に作成されたファイルに名前を付けます。AWS DMS は、20141029-1134010000.csv などの、タイムスタンプを使用して CDC ファイルの名前を付けます。レコードを含むソーステーブルごとに、ソーステーブルが空出ない限り、AWS DMS は、指定されたターゲット フォルダの下にフォルダを作成します。AWS DMS は、指定された Amazon S3

バケットに、すべての全ロードおよび CDC ファイルを書き込みます。AWS DMS が作成するファイルのサイズは、MaxFileSize のエンドポイント設定を使用して制御できます。

パラメータ bucketFolder には、.csv ファイルまたは .parquet ファイルが S3 バケットにアップロードされる前に保存される場所が含まれます。.csv ファイルでは、テーブルデータは、全ロードファイルとともに表示されている S3 バケットに以下の形式で保存されます。

```
database_schema_name/table_name/LOAD00000001.csv
database_schema_name/table_name/LOAD00000002.csv
...
database_schema_name/table_name/LOAD00000009.csv
database_schema_name/table_name/LOAD0000000A.csv
database_schema_name/table_name/LOAD0000000B.csv
...database_schema_name/table_name/LOAD0000000F.csv
database_schema_name/table_name/LOAD00000010.csv
...
```

追加の接続属性を使用して、列の区切り文字、行の区切り文字、およびその他のパラメータを指定できます。追加の接続属性の詳細については、このセクションの最後にある「AWS DMS のターゲットとして Amazon S3を使用する場合のエンドポイントの設定」をご参照ください。

次のとおり、ExpectedBucketOwner Amazon S3 エンドポイント設定を使用してバケット所有者を指定して、スナイプ攻撃を防ぐことができます。その後、接続をテストしたり移行の実行をリクエストしたりすると、S3 は指定されたパラメータに対してバケット所有者のアカウント ID をチェックします。

```
--s3-settings='{"ExpectedBucketOwner": "AWS_Account_ID"}'
```

AWS DMS を使用してデータ変更レプリケーションを行う場合、.csv または .parquet 出力ファイルの最初の列に、次の .csv ファイルのように、行データがどのように変更されたかが示されます。

```
I,101,Smith,Bob,4-Jun-14,New York
U,101,Smith,Bob,8-Oct-15,Los Angeles
U,101,Smith,Bob,13-Mar-17,Dallas
D,101,Smith,Bob,13-Mar-17,Dallas
```

この例では、ソースデータベースに EMPLOYEE テーブルがあるとします。AWS DMS は以下のイベントに応答して、データを .csv または .parquet ファイルに書き込みます。

- 新しい従業員 (Bob Smith、従業員 ID 101) がニューヨークオフィスに 2014 年 6 月 4 日に採用されました。.csv または .parquet ファイルで、最初の列の I は、新しい行がソースデータベースの EMPLOYEE テーブルに INSERT されたことを示しています。
- 2015 年 10 月 8 日に、Bob はロサンゼルスオフィスに転勤になりました。.csv または .parquet ファイルで、U は、Bob の新しい勤務地を反映するため、EMPLOYEE テーブルの対応する行が UPDATE されたことを示しています。その他の行は、UPDATE の後に表示される、EMPLOYEE テーブルの行を反映しています。
- 2017 年 3 月 13 日に、Bob はダラスオフィスに再度転勤になりました。.csv または .parquet ファイルで、U はこの行が再度 UPDATE されたことを示しています。その他の行は、UPDATE の後に表示される、EMPLOYEE テーブルの行を反映しています。
- Bob は、しばらくダラスに勤務した後、退職しました。.csv または .parquet ファイルで、D は、行がソーステーブルで DELETE されたことを示しています。その他の行は、削除される前に行が EMPLOYEE テーブルにどのように表示されたかを反映しています。

CDC のデフォルトでは、AWS DMS がトランザクションの順序に関係なく、各データベーステーブルの行の変更を保存することを覚えておいてください。トランザクション順序に従って CDC ファイルに行の変更を保存する場合は、S3 エンドポイント設定を使用して、S3 ターゲットに CDC トランザクションファイルを保存するフォルダ パスを指定する必要があります。詳細については、「S3 ターゲットでのトランザクション順序を含むデータ変更 (CDC) のキャプチャ」をご参照ください。

データ レプリケーション タスク中に Amazon S3 ターゲットへの書き込みの頻度を制御するには、cdcMaxBatchInterval と cdcMinFileSize の追加接続属性を設定することができます。これにより、追加のオーバーヘッド オペレーションなしでデータを分析する際のパフォーマンスが向上します。詳細については、「AWS DMS のターゲットとして Amazon S3を使用する場合のエンドポイントの設定」をご参照ください。

トピック

- ターゲットとして Amazon S3 を使用するための前提条件
- ターゲットとしての Amazon S3 の使用における制限
- セキュリティ
- Apache Parquet を使用した Amazon S3 オブジェクトの保存
- Amazon S3 オブジェクトのタグ付け
- Amazon S3 ターゲットオブジェクトを暗号化する AWS KMS キーの作成
- 日付ベースのフォルダパーティション分割を使用する

- AWS DMS のターゲットとして Amazon S3 を使用する場合のパーティション分割されたソースの並列ロード
- AWS DMS のターゲットとして Amazon S3を使用する場合のエンドポイントの設定
- AWS DMS のための Amazon S3 ターゲットでの AWS Glue Data Catalog データカタログの使用
- Amazon S3 ターゲットでのデータ暗号化、parquet ファイル、CDC の使用
- 移行済み S3 データでのソース DB オペレーションの表示
- S3 Parquet のターゲットデータ型

ターゲットとして Amazon S3 を使用するための前提条件

ターゲットとして Amazon S3 を使用する前に、以下が true であることを確認します：

- ターゲットとして使用している S3 バケットは、データの移行に使用している DMS レプリケーション インスタンスと同じ AWS リージョンにあります。
- 移行に使用する AWS アカウントには、ターゲットとして使用する S3 バケットへの書き込みおよび削除アクセス許可が付与された IAM ロールがあります。
- このロールにはタグ付けのためのアクセスが許可されているため、ターゲットバケットに書き込まれる任意の S3 オブジェクトにタグ付けができます。
- IAM ロールには DMS (dms.amazonaws.com) が[trusted entity] (信頼されたエンティティ) として追加されています。

このアカウントアクセスを設定するには、移行タスクを作成するために使用するユーザーアカウントに割り当てられたロールに次の一連のアクセス許可があることを確認します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:PutObjectTagging"
            ],
            "Resource": [
                "arn:aws:s3:::buckettest2/*"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::buckettest2"
            ]
        }
    ]
}
```

S3 をターゲットとする検証を使用する際の前提条件については、「S3 ターゲット検証の前提条件」を参照してください。

ターゲットとしての Amazon S3 の使用における制限

ターゲットとして Amazon S3 を使用する場合、以下の制限が適用されます：

- S3 のバージョニングは有効にしません。S3 のバージョニングが必要な場合は、ライフサイクルポリシーを使用して古いバージョンを積極的に削除します。これを行わないと、S3 `list-object` コールのタイムアウトが原因でエンドポイント接続テストが失敗する可能性があります。S3 バケットのライフサイクルポリシーを作成するには、「ストレージのライフサイクルの管理」を参照してください。S3 オブジェクトのバージョンを削除するには、「バージョニングが有効なバケットからのオブジェクトバージョンの削除」を参照してください。
- VPC 対応 (ゲートウェイ VPC) S3 バケットはバージョン 3.4.7 以降でサポートされます。
- 変更データキャプチャ (CDC) では、次のデータ定義言語 (DDL) コマンドがサポートされています：[Truncate Table] (テーブルの切り捨て)、[Drop Table] (テーブルの削除)、[Create Table] (テーブルの作成)、[Rename Table] (テーブルの名前変更)、[Add Column] (列追加)、[Rename Column] (列名変更)、[Change Column Data Type] (列データ型の変更)。ソースデータベースで列が追加、ドロップ、または名前の変更が実行されても、ターゲットの S3 バケットには ALTER ステートメントは記録されず、AWS DMS は以前に作成されたレコードを新しい構造に一致するように変更することがないことに注意します。AWS DMS は変更後に新しいテーブル構造を使用して新しいレコードを作成します。

> **Note**
>
> DDL の切り捨てオペレーションは、S3 バケットからすべてのファイルおよび対応するテーブルフォルダを削除します。タスク設定を使用して、この動作を無効にし、変更デー

タ キャプチャ (CDC) 中に DMS が DDL 動作を処理する方法を設定できます。詳細については、「変更処理の DDL 処理のタスク設定」をご参照ください。

- 完全 LOB モードはサポートされていません。
- 全ロード時のソーステーブル構造に対する変更はサポートされていません。全ロード時のデータ変更はできます。
- 同じソーステーブルから同じターゲット S3 エンドポイントバケットにデータをレプリケートする複数のタスクを実行すると、それらのタスクが同じファイルに書き込みます。同じテーブルのデータソースを使用する場合、異なるターゲットエンドポイント (バケット) を指定することをお勧めします。
- `BatchApply` は S3 エンドポイントに対応していません。バッチ 適用の使用 (例えば、S3 ターゲットに対して `BatchApplyEnabled` ターゲットメタデータ (タスク設定) を実行すると、データ損失につながることがあります。
- `DatePartitionEnabled` または `addColumnName` は `PreserveTransactions` または `CdcPath` と組み合わせて使用できません。
- AWS DMS は、変換ルールを使用した複数のソーステーブル名の同じターゲットフォルダへの変更をサポートしていません。
- フルロードフェーズ中にソーステーブルへの書き込みが多い場合、DMS は S3 バケットまたはキャッシュされた変更に重複レコードを書き込むことがあります。
- `TargetTablePrepMode` の `DO_NOTHING` を使用してタスクに設定すると、フルロードフェーズ中にタスクが突然停止して再開した場合に、DMS は S3 バケットに重複レコードを書き込むことがあります。
- `PreserveTransactions` を `true` に設定してターゲットエンドポイントを設定すると、テーブルをリロードしても以前に生成した CDC ファイルはクリアされません。詳細については、「S3 ターゲットでのトランザクション順序を含むデータ変更 (CDC) のキャプチャ」を参照してください。

S3 をターゲットとする検証を使用する際の制限については、「ターゲットの S3 の検証を使用する場合の制限」を参照してください。

セキュリティ

ターゲットとして Amazon S3 を使用するには、移行のために使用されるアカウントに、ターゲットとして使用される Amazon S3 バケットに対する書き込みおよび削除アクセス権限が前提で

す。Amazon S3 にアクセスするために必要なアクセス許可がある、IAM ロールの Amazon リソースネーム (ARN) を指定します。

AWS DMS は、既定アクセスコントロールリスト (ACL) で周知の Amazon S3に対する一連の事前定義済みの許可をサポートしています。各既定 ACL には、Amazon S3 バケットのアクセス許可を設定するために使用できる一連の被付与者とアクセス許可があります。S3 ターゲットエンドポイントの接続文字列属性で `cannedAclForObjects` で使用して、既定 ACL を指定できます。追加接続属性 `cannedAclForObjects` の使用についての詳細は、「AWS DMS のターゲットとして Amazon S3 を使用する場合のエンドポイントの設定」をご参照ください。Amazon S3 の既定 ACL の詳細については、「既定 ACL」をご参照ください。

移行に使用する IAM ロールは、`s3:PutObjectAcl` API オペレーションを実行できる必要があります。

Apache Parquet を使用した Amazon S3 オブジェクトの保存

カンマ区切り値 (.csv) 形式は、Amazon S3 ターゲットオブジェクトのデフォルトのストレージ形式です。よりコンパクトなストレージと高速なクエリについては、代わりに Apache Parquet (.parquet) をストレージ形式として使用できます。

Apache Parquet は、Hadoop 向けに当初設計されたオープンソースのファイルストレージ形式です。Apache Parquet の詳細については、「https://parquet.apache.org/」を参照してください。

移行された S3 ターゲットオブジェクトに .parquet ストレージ形式を設定するには、以下のメカニズムを使用できます。

- AWS CLI あるいは AWS DMS の API を使用してエンドポイントを作成するときに、JSON オブジェクトのパラメータとして指定するエンドポイント設定です。詳細については、「Amazon S3 ターゲットでのデータ暗号化、parquet ファイル、CDC の使用」をご参照ください。
- エンドポイント作成時にセミコロンで区切られたリストとして指定する追加の接続属性です。詳細については、「AWS DMS のターゲットとして Amazon S3を使用する場合のエンドポイントの設定」をご参照ください。

Amazon S3 オブジェクトのタグ付け

タスクテーブル マッピングルールの一部として適切な JSON オブジェクトを指定することで、レプリケーション インスタンスが作成する Amazon S3 オブジェクトにタグを付けることができます。有効なタグ名を含む S3 オブジェクトのタグ付けに関する要件とオプションの詳細については、Amazon Simple Storage Service ユーザーガイドの「オブジェクトのタグ付け」をご参照くださ

い。JSON を使用したテーブルマッピングの詳細については、「JSON を使用するテーブル選択および変換を指定する」をご参照ください。

selection ルールタイプの 1 つ以上の JSON オブジェクトを使用して、指定するテーブルおよびスキーマで S3 オブジェクトにタグ付けします。次に、post-processing ルールタイプの 1 つ以上の JSON オブジェクトで add-tag アクションを使用して、selection オブジェクト (1 つ以上のオブジェクト) を行います。この後処理ルールは、タグ付けする S3 オブジェクトを識別し、これらの S3 オブジェクトに追加するタグに名前と値を指定します。

post-processing ルールタイプの JSON オブジェクトで指定するパラメータは、次のテーブルにあります。

パラメータ	使用できる値:	説明
rule-type	post-processing	生成されたターゲットオブジェクトへの後処理アクションに適用する値です。選択した S3 オブジェクトには、1 つ以上の後処理ルールを指定できます。
rule-id	数値。	ルールを識別する一意の数値。
rule-name	英数字値。	ルールを特定する一意な名前。
rule-action	add-tag	S3 オブジェクトに適用する後処理アクションです。add-tag アクションの単一の JSON 後処理オブジェクトを使用して、1 つ以上のタグを追加できます。
object-locator	schema-name - テーブルスキーマの名前。 table-name - テーブルの名前。	ルールが適用されるスキーマおよびテーブルごとの名前。各 object-locator パラメータの値のすべてあるいは一部で、「%」パーセント記号をワイルドカードとして使用できます。したがって、これらの項目には以下を対応させることができます。

パラメータ	使用できる値:	説明
		• 単一のスキーマの 1 つのテーブル • すべてあるいは一部のスキーマの 1 つのテーブル • 単一のスキーマの一部あるいはすべてのテーブル • 一部あるいはすべてのスキーマの一部あるいはすべてのテーブル

パラメータ	使用できる値:	説明
tag-set	key - 単一のタグの任意の有効な名前。 value - このタグに有効な任意の JSON 値。	特定の object-locator に一致する作成された S3 オブジェクトごとに設定する 1 つ以上のタグの名前と値です。単一の tag-set パラメータオブジェクトで 10 個までのキー値ペアを指定することができます。S3 オブジェクトタグの詳細については、Amazon Simple Storage Service ユーザーガイドの「オブジェクトのタグ付け」をご参照ください。 また、タグの key および value パラメータの両方で、${*dyn-value*} を使用して値のすべてあるいは一部に動的な値を指定することもできます。ここでは、${*dyn-value*} は ${schema-name} あるいは ${table-name} のいずれかにできます。したがって、現在選択されているスキーマあるいはテーブルにパラメータ値の一部あるいは全体として名前を挿入することができます。 **Note** **Important** key パラメータに動的な値を挿入する場合には、使用方法に応じて、S3 オブジェクトに名前が重複するタグを生成することができます。この

パラメータ	使用できる値:	説明
		場合、重複するタグの 1 つのみがオブジェクトに追加されます。

S3 オブジェクトの選択をタグ付けするために複数の post-processing ルールタイプを指定する場合、各 S3 オブジェクトは 1 つの後処理ルールをから 1 つの tag-set オブジェクトのみを使用してタグ付けされます。指定する S3 オブジェクトへのタグ付けに使用される特定のタグセットは、その S3 オブジェクトに最も一致するオブジェクトロケーターに関連する後処理ルールのうちの 1 つです。

たとえば、同じ S3 オブジェクトに 2 つの後処理ルールが識別されるとします。また、1 つのルールのオブジェクトロケーターはワイルドカードを使用し、別のルールのオブジェクトロケーターは S3 オブジェクトを識別するための完全な一致 (ワイルドカードなし) を使用するとします。この場合、完全に一致する後処理ルールに関連付けられたタグが S3 オブジェクトのタグ付けに使用されます。複数の後処理ルールが指定された S3 オブジェクトに同様に一致する場合、この後処理ルールに最初に関連図けられたタグセットがオブジェクトのタグ付けに使用されます。

Example 単一のテーブルとスキーマに作成された S3 オブジェクトへの静的なタグの追加

次の選択と後処理ルールは 3 つのタグ (tag_1、tag_2、tag_3) と該当する静的値 (value_1、value_2、value_3) を追加して S3 オブジェクトを作成します。この S3 オブジェクトは、aat2 という名前のスキーマがある STOCK という名前のソース内の単一のテーブルに該当します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "5",
            "rule-name": "5",
            "object-locator": {
                "schema-name": "aat2",
                "table-name": "STOCK"
            },
            "rule-action": "include"
```

```
        },
        {
            "rule-type": "post-processing",
            "rule-id": "41",
            "rule-name": "41",
            "rule-action": "add-tag",
            "object-locator": {
                "schema-name": "aat2",
                "table-name": "STOCK"
            },
            "tag-set": [
              {
                "key": "tag_1",
                "value": "value_1"
              },
              {
                "key": "tag_2",
                "value": "value_2"
              },
              {
                "key": "tag_3",
                "value": "value_3"
              }
           ]
        }
    ]
}
```

Example 複数のテーブルとスキーマに作成された S3 オブジェクトへの静的および動的タグの追加

次の例には 1 つの選択と 2 つの後処理ルールがあり、ここでは、ソースからの入力にすべてのテーブルとすべてのスキーマが含まれています。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
```

```
            "rule-action": "include"
        },
        {
            "rule-type": "post-processing",
            "rule-id": "21",
            "rule-name": "21",
            "rule-action": "add-tag",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%",
            },
            "tag-set": [
              {
                "key": "dw-schema-name",
                "value":"${schema-name}"
              },
              {
                "key": "dw-schema-table",
                "value": "my_prefix_${table-name}"
              }
            ]
        },
        {
            "rule-type": "post-processing",
            "rule-id": "41",
            "rule-name": "41",
            "rule-action": "add-tag",
            "object-locator": {
                "schema-name": "aat",
                "table-name": "ITEM",
            },
            "tag-set": [
              {
                "key": "tag_1",
                "value": "value_1"
              },
              {
                "key": "tag_2",
                "value": "value_2"
              }           ]
        }
    ]
}
```

最初の後処理ルールは、該当する動的値 (${schema-name} と my_prefix_${table-name}) を使用した 2 つのタグ (dw-schema-name と dw-schema-table) をターゲットに作成されたほぼすべての S3 オブジェクトに追加します。例外は、2 番目の後処理ルールで色別されてタグ付けされる S3 オブジェクトです。したがって、ワイルドカードオブジェクトロケーターによって識別された各ターゲット S3 オブジェクトは、ソースで該当するスキーマおよびテーブルを識別するタグを使用して作成されます。

2 番目の後処理ルールは、完全に一致するオブジェクトロケーターによって識別される S3 オブジェクトに、該当する静的値 (value_1 と value_2) を使用して tag_1 および tag_2 を追加します。作成されたこの S3 オブジェクトはしたがって、aat という名前のスキーマがある ITEM という名前のソース内の単一のテーブルに該当します。完全一致のため、前述のタグは、最初の後処理ルール (ワイルドカードのみによって S3 に一致) でオブジェクトに追加されたタグを上書きします。

Example S3 オブジェクトに動的タグの名前と値の両方を追加する

次の例には 2 つの選択ルールと 1 つの後処理ルールがあります。ここでは、ソースの入力には retail あるいは wholesale スキーマに ITEM テーブルのみが含まれています。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "retail",
                "table-name": "ITEM"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "wholesale",
                "table-name": "ITEM"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "post-processing",
```

```
            "rule-id": "21",
            "rule-name": "21",
            "rule-action": "add-tag",
            "object-locator": {
                "schema-name": "%",
                "table-name": "ITEM",
            },
            "tag-set": [
              {
                "key": "dw-schema-name",
                "value":"${schema-name}"
              },
              {
                "key": "dw-schema-table",
                "value": "my_prefix_ITEM"
              },
              {
                "key": "${schema-name}_ITEM_tag_1",
                "value": "value_1"
              },
              {
                "key": "${schema-name}_ITEM_tag_2",
                "value": "value_2"
              }
            ]
    ]
}
```

後処理ルールのタグセットは、ターゲットの ITEM テーブルに作成されたすべての S3 オブジェクトに 2 つのタグ (dw-schema-name と dw-schema-table) を追加します。最初のタグには動的値 "${schema-name}" があり、2 つ目のタグには静的値 "my_prefix_ITEM" があります。したがって、各ターゲット S3 オブジェクトは、ソースで該当するスキーマおよびテーブルを識別するタグを使用して作成されます。

さらに、このタグセットは 2 つの追加タグを動的な名前 (${schema-name}_ITEM_tag_1 と "${schema-name}_ITEM_tag_2") で追加します。これには該当する静的値 (value_1 と value_2) があります。したがって、これらのタグはそれぞれ現在のスキーマ (retail あるいは wholesale) に対して命名されます。各オブジェクトは単一の一意のスキーマ名で作成されたため、このオブジェクトで動的な名前を重複することはできません。スキーマ名はそれ以外の一意のタグ名を作成するために使用されます。

Amazon S3 ターゲットオブジェクトを暗号化する AWS KMS キーの作成

Amazon S3 ターゲットオブジェクトを暗号化するカスタム AWS KMS キーを作成して使用できます。KMS キーを作成したら、S3 ターゲットエンドポイントを作成するときに次のいずれかの方法を使用して、KMS キーを使用してオブジェクトを暗号化できます。

- AWS CLI を使用して `create-endpoint` コマンドを実行するときに、S3 ターゲットオブジェクトに対して次のオプションを (デフォルトの .csv ファイルストレージ形式で) 使用します。

```
--s3-settings '{"ServiceAccessRoleArn": "your-service-access-ARN",
"CsvRowDelimiter": "\n", "CsvDelimiter": ",", "BucketFolder": "your-bucket-folder",
"BucketName": "your-bucket-name", "EncryptionMode": "SSE_KMS",
"ServerSideEncryptionKmsKeyId": "your-KMS-key-ARN"}'
```

  ここで *your-KMS-key-ARN* とは、KMS キーの Amazon リソースネーム (ARN) です。詳細については、「Amazon S3 ターゲットでのデータ暗号化、parquet ファイル、CDC の使用」をご参照ください。

- 値 (SSE_KMS) に追加の接続属性 (encryptionMode) を設定し、KMS キーの ARN に追加の接続属性 (serverSideEncryptionKmsKeyId) を設定します。詳細については、「AWS DMS のターゲットとして Amazon S3を使用する場合のエンドポイントの設定」をご参照ください。

KMS キーを使用して Amazon S3 ターゲットオブジェクトを暗号化するには、Amazon S3 バケットにアクセスする許可がある IAM ロールが必要です。次に、この IAM ロールは作成した暗号化キーに添付されるポリシー (キーポリシー) にアクセスします。これは、IAM コンソールで次を作成して実行します：

- Amazon S3 バケットにアクセスする権限があるポリシーです。
- このポリシーがある IAM ロールです。
- このロールを参照するキーポリシーを持つ KMS 暗号化キー

以下の手順でこれを行う方法について説明します。

Amazon S3 バケットへのアクセス許可を持つ IAM ポリシーを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで、ナビゲーションペインの [ポリシー] を選択します。[ポリシー] ページが開きます。

3. [Create policy] (ポリシーを作成) を選択します。[Create policy (ポリシーの作成)] ページが開きます。
4. [サービス]、[S3] の順に選択します。アクションのアクセス権限の一覧が表示されます。
5. [すべて展開] を選択して一覧を展開し、少なくとも以下のアクセス権限を選択します。

   - ListBucket
   - PutObject
   - DeleteObject

   必要に応じて他のアクセス権限を選択したら、[すべて折りたたむ] を選択して一覧を折りたたみます。
6. [リソース] を選択してアクセスするリソースを指定します。少なくとも [All resources] (すべてのリソース) を選択して、全般的な Amazon S3 リソースへのアクセスを提供します。
7. 必要に応じて他の条件やアクセス許可を追加したら、[ポリシーの確認] を選択します。[ポリシーの確認] ページで結果を確認します。
8. 設定が必要に応じている場合には、ポリシーの名前 (`DMS-S3-endpoint-access` など) と他の説明を入力し、[ポリシーの作成] を選択します。[ポリシー] ページが開き、ポリシーが作成されたことを示すメッセージが表示されます。
9. [ポリシー] のリストからポリシー名を検索して選択します。[概要] ページが表示され、次のようなポリシーの JSON を示します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": "*"
        }
    ]
}
```

これで、暗号化のために Amazon S3 リソースにアクセスする新しいポリシーが指定した名前 (DMS-S3-endpoint-access など) で作成されました。

このポリシーを使用して IAM ロールを作成するには

1. IAM コンソールで、ナビゲーションペインの [Roles] (ロール) を選択します。[ロール] の詳細ページが開きます。
2. [Create role] (ロールの作成) を選択します。[Create role] (ロールの作成) ページが開きます。
3. AWS のサービスが信頼されるエンティティとして選択されたら、この IAM ロールを使用するサービスとして DMS を選択します。
4. [Next: Permissions] (次へ: アクセス許可) を選択します。[Create role] (ロールの作成) ページに [Attach permissions policies] (アクセス権限ポリシーの添付) ビューが表示されます。
5. 前の手順で作成した IAM ロールの IAM ポリシーを見つけて選択します (DMS-S3-endpoint-access)。
6. [Next:Tags](次へ: タグ) を選択します。[Create role] (ロールの作成) ページに[Add tags] (タグを追加) ビューが表示されます ここでは、任意のタグを追加することができます。
7. [Next: Review] (次のステップ: レビュー) を選択します。[Create role] (ロールの作成) ページに[Review] (確認) ビューが表示されます。ここで、結果を確認できます。
8. 設定が必要に応じている場合には、ロールの名前 (必須、DMS-S3-endpoint-access-role など) と追加の説明を入力して、[Create role] (ロールの作成) を選択します。[ロール] の詳細ページが開き、ロールが作成されたことを示すメッセージが表示されます。

これで、暗号化のために Amazon S3 リソースにアクセスする新しいロールが指定した名前 (DMS-S3-endpoint-access-role など) で作成されました。

この IAM ロールを参照するキーポリシーを持つ KMS 暗号化キーを作成するには

> **Note**
>
> AWS DMS と AWS KMS 暗号化キーの連携について詳しくは、「暗号化キーの設定と AWS KMS アクセス許可の指定」をご参照ください。

1. AWS Management Console にサインインし、AWS Key Management Service (AWS KMS) コンソール (https://console.aws.amazon.com/kms) を開きます。
2. AWS リージョン を変更するには、ページの右上隅にあるリージョンセレクターを使用します。

3. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。
4. [Create key] (キーの作成) を選択します。[キーの設定] ページが開きます。
5. [キーの種類] で、[対称] を選択します。

> **Note**
>
> このキーを作成する場合、Amazon S3 などすべての AWS サービスでは対称暗号化キーしか使用できないため、作成できるのは対称キーのみです。

6. [アドバンスドオプション] を選択します。[キーマテリアルのオリジン] で、[KMS] が選択されていることを確認し、[次へ] を選択します。[ラベルの追加] ページが開きます。
7. [エイリアスと説明の作成] で、キーのエイリアス (`DMS-S3-endpoint-encryption-key` など) と追加の説明を入力します。
8. [タグ] で、キーを識別してその使用状況を追跡するために役立つ任意のタグを追加したら、[次へ] を選択します。[キー管理アクセス許可の定義] ページが開き、選択できるユーザーおよびロールの一覧が表示されます。
9. キーを管理するユーザーおよびロールを追加します。このユーザーとロールにキーを管理するために必要な権限があることを確認してください。
10. [キーの削除] で、キー管理者がそのキーを削除できるかどうかを選択したら、[次へ] を選択します。[キーの使用アクセス許可の定義] ページが開き、選択できる追加のユーザーおよびロールの一覧が表示されます。
11. [This account](このアカウント) で、Amazon S3 ターゲットに対して暗号化オペレーションを実行できるユーザーを選択します。また、[Roles] (ロール) で以前に作成したロールを選択して、Amazon S3 ターゲットオブジェクト (`DMS-S3-endpoint-access-role` など)を暗号化するためのアクセスを有効化します。
12. リストにない他のアカウントに同じアクセス権限を付与するには、[他の AWS アカウント] で [別の AWS アカウントを追加する]、[次へ] の順に選択します。[キーポリシーの表示と編集] ページが開き、既存の JSON に入力して表示および編集できるキーポリシーの JSON が表示されます。ここでは、前のステップで選択したロールおよびユーザー (例えば、`Admin` と `User1`) を参照するキーポリシーを表示できます。また、次の例に示すように、異なるプリンシパル (ユーザーとロール) に許可される別々のキーアクションも確認できます。

```
{
  "Id": "key-consolepolicy-3",
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:root"
        ]
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow access for Key Administrators",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:role/Admin"
        ]
      },
      "Action": [
        "kms:Create*",
        "kms:Describe*",
        "kms:Enable*",
        "kms:List*",
        "kms:Put*",
        "kms:Update*",
        "kms:Revoke*",
        "kms:Disable*",
        "kms:Get*",
        "kms:Delete*",
        "kms:TagResource",
        "kms:UntagResource",
        "kms:ScheduleKeyDeletion",
        "kms:CancelKeyDeletion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:role/DMS-S3-endpoint-access-role",
          "arn:aws:iam::111122223333:role/Admin",
```

```
          "arn:aws:iam::111122223333:role/User1"
        ]
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:role/DMS-S3-endpoint-access-role",
          "arn:aws:iam::111122223333:role/Admin",
          "arn:aws:iam::111122223333:role/User1"
        ]
      },
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": "*",
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": true
        }
      }
    }
  ]
```

13. [終了] を選択します。[Encryption keys] (暗号化キー) ページが開き、KMS キーが作成されたことを示すメッセージが表示されます。

これで、指定したエイリアス (`DMS-S3-endpoint-encryption-key` など) を使用する新しい KMS キーが作成されました。このキーによって AWS DMS は Amazon S3 ターゲットオブジェクトを暗号化できます。

日付ベースのフォルダパーティション分割を使用する

AWS DMS は、ターゲット エンドポイントとして Amazon S3 を使用する場合のトランザクション コミット日に基づく S3 フォルダパーティションをサポートします。日付ベースのフォルダパーティション分割を使用すると、1 つのソーステーブルから S3 バケット内の時間階層フォルダ構造にデータを書き込むことができます。S3 ターゲット エンドポイントを作成するときにフォルダをパーティション分割することで、次のことが可能になります：

- S3 オブジェクトの管理が向上します
- 各 S3 フォルダのサイズを制限する
- データレイクのクエリやその他の後続オペレーションを最適化する

S3 ターゲット エンドポイントを作成するときに、日付ベースのフォルダのパーティション分割を有効にできます。このモードは、既存のデータを移行して進行中の変更レプリケーションを行うか (全ロード + CDC)、またはデータ変更のみレプリケーションする (CDCのみ) とき有効にできます。以下のターゲット エンドポイント設定を使用します：

- `DatePartitionEnabled` — 日付に基づいてパーティション分割を指定します。このブールオプションを `true` に設定し、トランザクションのコミット日に基づいて S3 バケット フォルダをパーティション分割します。

  この設定を `PreserveTransactions` や `CdcPath` と併用することはできません。

  デフォルト値は `false` です。
- `DatePartitionSequence` - フォルダのパーティション分割中に使用する日付形式のシーケンスを識別します。この ENUM オプションを `YYYYMMDD` または `YYYYMMDDHH`、`YYYYMM`、`MMYYYYDD`、`DDMMYYYY` に設定します。デフォルト値は `YYYYMMDD` です。`DatePartitionEnabled` が `true`. に設定されている場合は、この設定を使用します。
- `DatePartitionDelimiter` - フォルダのパーティション分割時に使用する日付区切り記号を指定します。この ENUM オプションを `SLASH` または `DASH`、`UNDERSCORE`、`NONE` に設定します。デフォルト値は `SLASH` です。`DatePartitionEnabled` が `true` に設定されている場合は、この設定を使用します。

次の例は、データパーティションシーケンスと区切り文字のデフォルト値を使用して、日付ベースのフォルダパーティション分割を有効にする方法を示します。これは、AWS CLI.`create-endpoint` コマンド のオプション `--s3-settings '{`*`json-settings`*`}'` を使用します。

```
  --s3-settings '{"DatePartitionEnabled": true,"DatePartitionSequence":
"YYYYMMDD","DatePartitionDelimiter": "SLASH"}'
```

AWS DMS のターゲットとして Amazon S3 を使用する場合のパーティション分割されたソースの並列ロード

パーティション化されたデータソースの Amazon S3 ターゲットへの並列フルロードを設定できます。この方法を使用すると、サポートされているソースデータベースエンジンからパーティション化されたデータを S3 ターゲットに移行するため、ロード時間を短縮できます。パーティション化されたソースデータのロード時間を短縮するには、ソースデータベースのすべてのテーブルのパーティションにマップされた S3 ターゲットサブフォルダを作成します。このようなパーティションにバインドされたサブフォルダを使用すると、AWS DMS は並列プロセスを実行してターゲット上の各サブフォルダにデータを入力できます。

テーブルマッピングの `table-settings` ルールについて、S3 では S3 ターゲットの並列フルロードを設定するための次の 3 つの `parallel-load` ロードルールタイプをサポートしています。

- `partitions-auto`
- `partitions-list`
- `ranges`

上記ルールタイプの詳細については、「テーブルとコレクション設定のルールとオペレーション」を参照してください。

`partitions-auto` ルールタイプと `partitions-list` ルールタイプの場合、AWS DMS は、次のとおりソースエンドポイントの各パーティション名を使用してターゲットのサブフォルダ構造を特定します。

```
bucket_name/bucket_folder/database_schema_name/table_name/partition_name/
LOADseq_num.csv
```

この場合、データを移行して S3 ターゲットに保存するサブフォルダパスには、同じ名前のソースパーティションに対応する追加の *partition_name* サブフォルダがあります。この *partition_name* サブフォルダには、指定されたソースパーティションから移行されたデータを含む単一または複数の LOAD*seq_num*.csv ファイルが保存されます。この *seq_num* は、.csv ファ

イルの末尾に付けられるシーケンス番号の接尾辞です。例えば、LOAD00000001.csv という .csv ファイル名の場合は、00000001 の部分です。

MongoDB や DocumentDB など、データベースエンジンによっては、パーティション分割の概念がない場合があります。このようなデータベースエンジンの場合、AWS DMS は、次のとおり実行中のソースセグメントのインデックスをプレフィックスとしてターゲットの .csv ファイル名に追加します。

```
.../database_schema_name/table_name/SEGMENT1_LOAD00000001.csv
.../database_schema_name/table_name/SEGMENT1_LOAD00000002.csv
...
.../database_schema_name/table_name/SEGMENT2_LOAD00000009.csv
.../database_schema_name/table_name/SEGMENT3_LOAD0000000A.csv
```

この例では、SEGMENT1_LOAD00000001.csv ファイルと SEGMENT1_LOAD00000002.csv ファイルには同じ実行中のソースセグメントインデックスのプレフィックス SEGMENT1 が名前に追加されています。この 2 つの .csv ファイルに移行されるソースデータは同じ実行中のソースセグメントインデックスに関連付けられているため、このような命名になります。一方、ターゲットの SEGMENT2_LOAD00000009.csv ファイルと SEGMENT3_LOAD0000000A.csv ファイルに保存されている移行データは、実行中の別のソースセグメントインデックスに関連付けられます。各ファイルのファイル名には、実行中のセグメントインデックス名の SEGMENT2 と SEGMENT3 がプレフィックスとして付けられます。

ranges 並列ロードタイプの場合、table-settings ルールの columns と boundaries の設定を使用して列名と列値を定義します。このようなルールを使用すると、次のとおりセグメント名に対応するパーティションを指定できます。

```
"parallel-load": {
    "type": "ranges",
    "columns": [
         "region",
         "sale"
    ],
    "boundaries": [
          [
               "NORTH",
               "1000"
          ],
          [
               "WEST",
               "3000"
```

```
            ]
    ],
    "segment-names": [
            "custom_segment1",
            "custom_segment2",
            "custom_segment3"
    ]
}
```

この場合、segment-names の設定は、S3 ターゲットでデータを並行して移行するための 3 つのパーティション名を定義しています。移行データは並行してロードされ、次のとおりパーティションサブフォルダの下の .csv ファイルに順番に保存されます。

```
.../database_schema_name/table_name/custom_segment1/LOAD[00000001...].csv
.../database_schema_name/table_name/custom_segment2/LOAD[00000001...].csv
.../database_schema_name/table_name/custom_segment3/LOAD[00000001...].csv
```

ここで、AWS DMS は、3 つの各パーティションサブフォルダに .csv ファイルのセットを保存します。各パーティションのサブフォルダ内の一連の .csv ファイルには、すべてのデータが移行されるまで LOAD00000001.csv から始まるインクリメントでファイル名が付けられます。

場合によっては、segment-names 設定を使用して ranges 並列ロードタイプのパーティションサブフォルダに明示的に名前を付けないことがあります。このような場合、AWS DMS は、*table_name* サブフォルダの下に各 .csv ファイルのシリーズを作成するというデフォルトの方法を適用します。この場合、AWS DMS は、次のとおり実行中のソースセグメントインデックス名を各 .csv ファイルのシリーズのファイル名にプレフィックスとして付けます。

```
.../database_schema_name/table_name/SEGMENT1_LOAD[00000001...].csv
.../database_schema_name/table_name/SEGMENT2_LOAD[00000001...].csv
.../database_schema_name/table_name/SEGMENT3_LOAD[00000001...].csv
...
.../database_schema_name/table_name/SEGMENTZ_LOAD[00000001...].csv
```

AWS DMS のターゲットとして Amazon S3を使用する場合のエンドポイントの設定

追加の接続属性の使用と同様、エンドポイントの設定を使用して、ターゲットの Amazon S3 データベースを設定できます。ターゲットエンドポイントを作成する際に、AWS DMS コンソールを使用するか、AWS CLI で --s3-settings '{"*EndpointSetting*": *"value"*, *...*}' の JSON 構文を指定して create-endpoint コマンドを使用して設定を指定します。

次の表は、ターゲットとして Amazon S3 を使用できるエンドポイント設定を説明しています。

オプション	説明
CsvNullValue	AWS DMS の null 値処理方法を指定するオプションのパラメータです。NULL 値の処理中に、このパラメータを使用して、ターゲットに書き込むときにユーザー定義の文字列を NULL として渡すことができます。例えば、ターゲット列が NULL が許容しない場合、このオプションを使用して空の文字列値と null 値を区別できる。このパラメータ値を空の文字列 ("" または '') と設定すると、AWS DMS は空の文字列を NULL ではなく null 値として処理する。 デフォルト値: NULL 有効な値:任意の有効な文字列 例: `--s3-settings '{"CsvNullValue": " "}'`
AddColumnName	true または y に設定された場合に .csv 出力ファイルに列名情報を追加するために使用できるオプションのパラメータ。 このパラメータを PreserveTransactions や CdcPath と併用することはできない。 デフォルト値: false 有効な値: true、false、y、n 例: `--s3-settings '{"AddColumnName": true}'`
AddTrailingPaddingCharacter	S3 ターゲットエンドポイント設定 AddTrailingPaddingCharacter を使用して文字列データにパディングを追加する。デフォルト値は false です。 型: ブール 例: `--s3-settings '{"AddTrailingPaddingCharacter": true}'`
BucketFolder	S3バケット内のフォルダ名を設定するオプションのパラメータ。このパラメータを指定した場合、ターゲットオブジェクトは .csv または .parquet ファイルとしてパス *BucketFolder* /*schema_na*

オプション	説明
	me /*table_name* / に作成されます。このパラメータを指定しない場合、使用されるパスは *schema_name* /*table_name* / となります。 例: `--s3-settings '{"BucketFolder": "testFolder"}'`
`BucketName`	S3 ターゲットオブジェクトが .csv または .parquet ファイルとして作成される S3 バケットの名前。 例: `--s3-settings '{"BucketName": "buckettest"}'`
`CannedAcl ForObjects`	S3 バケットに csv または .parquet ファイルとして作成されたオブジェクトに対して、定義済みの (既定の) アクセスコントロールリストを AWS DMS が指定できるようにする値。Amazon S3 の既定 ACL の詳細については、Amazon S3 デベロッパーガイドの「既定 ACL」をご参照ください。 デフォルト値: なし この属性の有効な値は、NONE、PRIVATE、PUBLIC_READ、PUBLIC_READ_WRITE、AUTHENTICATED_READ、AWS_EXEC_READ、BUCKET_OWNER_READ、BUCKET_OWNER_FULL_CONTROL です。 例: `--s3-settings '{"CannedAclForObjects": "PUBLIC_READ"}'`

オプション	説明
CdcInsertsOnly	変更データキャプチャ (CDC) ロード時のオプションのパラメータ。INSERT オペレーションのみをカンマ区切り値 (.csv) または列ストレージ (.parquet) の出力ファイルに書き込みます。デフォルトでは (false 設定)、.csv または.parquet レコードの最初のフィールドに、I (INSERT)、U (UPDATE)、または D (DELETE) という文字が含まれます。この文字は、ターゲットへの CDC ロードのためにソースデータベースで行が挿入、更新、または削除されたかどうかを示します。cdcInsertsOnly が true または y に設定されている場合、ソースデータベースからの INSERT のみが .csv または .parquet ファイルに移行されます。 .csv 形式の場合にのみ、これらの INSERTS の記録方法は IncludeOpForFullLoad の値によって異なります。IncludeOpForFullLoad が true に設定されている場合、各 CDC レコードの最初のフィールドは、ソースでの INSERT オペレーションを示す I に設定されます。IncludeOpForFullLoad が false に設定されている場合、各 CDC レコードは、ソースでの INSERT オペレーションを示す最初のフィールドなしで書き込まれます。これらのパラメータがどのように連動するかの詳細については、「移行済み S3 データでのソース DB オペレーションの表示」をご参照ください。 デフォルト値: false 有効な値: true、false、y、n 例: --s3-settings '{"CdcInsertsOnly": true}'

オプション	説明
CdcInsertsAndUpdates	変更データキャプチャ (CDC) ロードを有効化し、INSERT および UPDATE オペレーションを .csv または .parquet (列指向ストレージ) 出力ファイルに書き込みます。デフォルト設定は false ですが、cdcInsertsAndUpdates が true または y に設定されている場合、ソースデータベースからの INSERT および UPDATE が .csv または .parquet ファイルに移行されます。 .csv 形式の場合、これらの INSERTS および UPDATE の記録方法は includeOpForFullLoad パラメータの値によって異なります。includeOpForFullLoad が true に設定されている場合、各 CDC レコードの最初のフィールドは、ソースでの INSERT および UPDATE オペレーションを示す I または U に設定されます。しかし、includeOpForFullLoad が false に設定されている場合、CDC レコードはソースでの INSERT または UPDATE オペレーションを示すことなく書き込まれます。 これらのパラメータがどのように連動するかの詳細については、「移行済み S3 データでのソース DB オペレーションの表示」をご参照ください。 **Note** CdcInsertsOnly および cdcInsertsAndUpdates の両方を、同じエンドポイントで true に設定することはできません。同じエンドポイントで cdcInsertsOnly と cdcInsertsAndUpdates のどちらかを true に設定できますが、両方を設定することはできません。 デフォルト値: false 有効な値: true、false、y、n 例: --s3-settings '{"CdcInsertsAndUpdates": true}'

オプション	説明
CdcPath	CDC ファイルのフォルダパスを指定します。S3 ソースについては、タスクで変更データをキャプチャする場合、この属性が必須ですが、それ以外の場合はオプションです。CdcPath が設定された場合、DMS はこのパスから CDC ファイルを読み取り、データ変更をターゲットエンドポイントにレプリケーションします。S3 ターゲットについて PreserveTransactions を true に設定した場合、DMS は、DMS が CDC ロードのトランザクション順序を保存できる S3 ターゲット上のフォルダパスにこのパラメータが設定されていることを確認します。DMS は、S3 ターゲットの作業ディレクトリまたは BucketFolder と BucketName で指定された S3 ターゲットの場所のいずれかにこの CDC フォルダパスを作成します。 このパラメータを DatePartitionEnabled や AddColumnName と併用することはできない。 型: 文字列 例えば、CdcPath を MyChangedData と指定し,BucketName を MyTargetBucket と指定し、BucketFolder は指定しない場合、DMS は次の CDC フォルダパス MyTargetBucket/MyChangedData を作成します。 同じ CdcPath を指定し、BucketName を MyTargetBucket と指定し、BucketFolder を MyTargetData と指定すると、DMS は次の CDC フォルダパス MyTargetBucket/MyTargetData/MyChangedData を作成します。 **Note** この設定は、AWS DMS バージョン 3.4.2 以降でサポートされる。 トランザクション順序でデータ変更をキャプチャする場合、DMS は、ターゲットの DataFormat S3 設定の値に関係なく、常に行の変更を.csv ファイルに保存します。DMS は、.parquet

オプション	説明
	ファイルを使用してデータ変更をトランザクション順序に保存しません。
CdcMaxBatchInterval	ファイルを Amazon S3 に出力するための最大インターバル長条件 (秒単位)です。 デフォルト値は 60 秒です。 CdcMaxBatchInterval と CdcMinFileSize が指定されている場合、ファイルの書き込みは、最初に満たされるパラメータ条件によってトリガーされる。
CdcMinFileSize	Amazon S3 にファイルを出力するための KB 単位による最小ファイルサイズ条件です。 デフォルト値: 32000 KB) CdcMinFileSize と CdcMaxBatchInterval が指定されている場合、ファイルの書き込みは、最初に満たされるパラメータ条件によってトリガーされる。

オプション	説明
PreserveTransactions	true に設定されている場合、DMS は、CdcPath によって指定された Amazon S3 ターゲットに変更データ キャプチャ (CDC) のトランザクション順序を保存します。 このパラメータを DatePartitionEnabled や AddColumnName と併用することはできない。 型: ブール トランザクション順序でデータ変更をキャプチャする場合、DMS は、ターゲットの DataFormat S3 設定の値に関係なく、常に行の変更を.csv ファイルに保存します。DMS は、.parquet ファイルを使用してデータ変更をトランザクション順序に保存しません。 **Note** この設定は、AWS DMS バージョン 3.4.2 以降でサポートされる。

オプション	説明
IncludeOpForFullLoad	全ロード時のオプションのパラメータ。INSERT オペレーションのみをカンマ区切り値 (.csv) 出力ファイルに書き込みます。 全ロードの場合、レコードの挿入のみ可能です。デフォルト (false 設定) では、全ロードの場合、ソースデータベースで行が挿入されたことを示す情報はこれらの出力ファイルに記録されません。IncludeOpForFullLoad が true または y に設定されている場合、INSERT は .csv ファイルの最初のフィールドに I 注釈として記録されます。 **Note** このパラメータは、.csv ファイルへの出力の場合にのみ、CdcInsertsOnly または CdcInsertsAndUpdates と連動します。これらのパラメータがどのように連動するかの詳細については、「移行済み S3 データでのソース DB オペレーションの表示」をご参照ください。 デフォルト値: false 有効な値: true、false、y、n 例: --s3-settings '{"IncludeOpForFullLoad": true}'
CompressionType	GZIP に設定された場合にターゲットの .csv または .parquet ファイルを圧縮するために GZIP を使用するオプションのパラメータ。このパラメータがデフォルトに設定されている場合、ファイルは圧縮されないままになります。 デフォルト値: NONE 有効な値: GZIP または NONE 例: --s3-settings '{"CompressionType": "GZIP"}'

オプション	説明
`CsvDelimiter`	.csv ソースファイル内の列を分離するために使用される区切り文字。デフォルトはカンマ (,) です。 例: `--s3-settings '{"CsvDelimiter": ","}'`
`CsvRowDelimiter`	.csv ソースファイル内の行を分離するために使用される区切り文字。デフォルトでは、改行 (\n) です。 例: `--s3-settings '{"CsvRowDelimiter": "\n"}'`
`MaxFileSize`	全ロードで S3 ターゲットに移行中に作成される .csv ファイルの最大サイズ (KB 単位) を指定する値です。 デフォルト値: 1,048,576 KB (1 GB) 有効な値: 1 ~ 1,048,576 例: `--s3-settings '{"MaxFileSize": 512}'`
`Rfc4180`	.csv ファイル形式のみを使用して Amazon S3 に移行するデータに対して、RFC に準拠する動作を設定するために使用するオプションのパラメータです。Amazon S3 をターゲットとして使用してこの値を `true` または `y` に設定する場合、データに引用符、カンマ、または改行文字が含まれていると、AWS DMS は列全体を追加の二重引用符 (") で囲む。データ内のすべての引用符が 2 回が繰り返されます。このフォーマットは、RFC 4180 に準拠しています。 デフォルト値: `true` 有効な値: `true`、`false`、`y`、`n` 例: `--s3-settings '{"Rfc4180": false}'`

オプション	説明
EncryptionMode	S3 にコピーした .csv または .parquet オブジェクトファイルを暗号化するサーバー側の暗号化モードです。有効な値は、SSE_S3 (S3 サーバー側の暗号化) または SSE_KMS (KMS キーの暗号化) です。SSE_KMS を選択する場合、暗号化のために使用する KMS キーの Amazon リソースネーム (ARN) に ServerSideEncryptionKmsKeyId パラメータを設定します。 **Note** CLI modify-endpoint コマンドを使用して、既存のエンドポイントの EncryptionMode のパラメータ値を SSE_KMS から SSE_S3 に変更するすることもできます。しかし EncryptionMode の値を SSE_S3 から SSE_KMS に変えることはできません。 デフォルト値: SSE_S3 有効な値: SSE_S3 または SSE_KMS 例: --s3-settings '{"EncryptionMode": SSE_S3}'
ServerSideEncryptionKmsKeyId	EncryptionMode を SSE_KMS に設定した場合は、このパラメータを KMS キーの Amazon リソースネーム (ARN) に設定します。アカウントに作成した AWS KMS キーのリスト内でキーエイリアスを選択すると、この ARN が見つかります。キーを作成する場合、特定のポリシーと関連付けられるロールをこの KMS キーに関連付ける必要があります。詳細については、「Amazon S3 ターゲットオブジェクトを暗号化する AWS KMS キーの作成」を参照してください。 例: --s3-settings '{"ServerSideEncryptionKmsKeyId":"arn:aws:kms:us-east-1:111122223333:key/11a1a1a1-aaaa-9999-abab-2bbbbbb222a2"}'

オプション	説明
DataFormat	S3 オブジェクトを作成するために AWS DMS が使用するファイルの出力形式。Amazon S3 ターゲットに対して AWS DMS は、.csv または .parquet ファイルをサポートしています。.parquet ファイルには、十分な圧縮オプションおよびより高速なクエリパフォーマンスのバイナリ列指向ストレージがあります。.parquet ファイルについての詳細は、「https://parquet.apache.org/」を参照してください。 デフォルト値: csv 有効な値: csv または parquet 例: --s3-settings '{"DataFormat": "parquet"}'
EncodingType	Parquet エンコードタイプです。このエンコードタイプオプションには、次が含まれます。 • rle-dictionary - このディクショナリエンコードは、より効率的な繰り返し値を保存するためにビットパッキングとランレングスエンコードを組み合わせて使用します。 • plain - エンコードなし。 • plain-dictionary - このディクショナリエンコードは、特定の列で発生する値のディクショナリを構築します。ディクショナリは各列チャンクのディクショナリページに保存されます。 デフォルト値: rle-dictionary 有効な値: rle-dictionary 、plain、または plain-dictionary 例: --s3-settings '{"EncodingType": "plain-dictionary"}'

オプション	説明
DictPageSizeLimit	.parquet ファイルのディクショナリページで許容される最大サイズ (バイト単位)。ディクショナリページがこの値を超えると、このページではプレーンエンコードが使用されます。 デフォルト値: 1,024,000 (1 MB) 有効な値: 任意の有効な整数値 例: `--s3-settings '{"DictPageSizeLimit": 2,048,000 }'`
RowGroupLength	.parquet ファイルの 1 つの行グループの行数です。 デフォルト値: 10,024 (10 KB) 有効値: 任意の有効な整数 例: `--s3-settings '{"RowGroupLength": 20,048}'`
DataPageSize	.parquet ファイルのデータページで許容される最大サイズ (バイト単位)。 デフォルト値: 1,024,000 (1 MB) 有効値: 任意の有効な整数 例: `--s3-settings '{"DataPageSize": 2,048,000}'`
ParquetVersion	.parquet ファイル形式のバージョン。 デフォルト値: PARQUET_1_0 有効な値: PARQUET_1_0 または PARQUET_2_0 例: `--s3-settings '{"ParquetVersion": "PARQUET_ 2_0"}'`

オプション	説明
EnableStatistics	true または y に設定された場合に、.parquet ファイルと行グループに関する統計を有効化します。 デフォルト値: true 有効な値: true、false、y、n 例: --s3-settings '{"EnableStatistics": false}'
TimestampColumnName	S3 ターゲットエンドポイントデータにタイムスタンプ列を含めるためのオプションのパラメータ。 TimestampColumnName を空白以外の値に設定すると、AWS DMS に移行済みデータの .csv ファイルまたは .parquet オブジェクトファイルに STRING 列が追加される。 全ロードの場合、このタイムスタンプ列の各行には、データが DMS によってソースからターゲットに転送されたときのタイムスタンプが含まれます。 CDC ロードの場合、タイムスタンプ列の各行には、ソースデータベースでのその行のコミットのタイムスタンプが含まれます。 タイムスタンプ列の値の文字列形式は yyyy-MM-dd HH:mm:ss.SSSSSS です。デフォルトでは、この値の精度はマイクロ秒です。CDC 負荷の場合、精度の丸めについては、ソースデータベースに対して DMS でサポートされているコミットのタイムスタンプによって異なります。 AddColumnName パラメータが true に設定されている場合、DMS は TimestampColumnName の空白以外の値として設定したタイムスタンプ列の名前も含める。 例: --s3-settings '{"TimestampColumnName": "TIMESTAMP"}'

オプション	説明
UseTaskStartTimeForFullLoadTimestamp	、このパラメータは true に設定した場合、時間データがターゲットに書き込まれるのではなく、タスクのスタート時刻をタイムスタンプ列の値として使用します。全ロードの場合、UseTaskStartTimeForFullLoadTimestamp が true に設定される場合、タイムスタンプ列の各行には、タスクのスタート時刻が格納されます。CDC ロードの場合、タイムスタンプ列の各行には、トランザクションのコミット時刻が含まれます。 UseTaskStartTimeForFullLoadTimestamp が false に設定される場合、タイムスタンプカラムの全ロードタイムスタンプは、データがターゲットに到着した時間により増分となります。 デフォルト値: false 有効な値: true、false 例: --s3-settings '{"UseTaskStartTimeForFullLoadTimestamp": true}' UseTaskStartTimeForFullLoadTimestamp: true は、CDC ロード用に TimestampColumnName で並べ替えることが可能な全ロードに対して S3 ターゲット TimestampColumnName の作成を助けます。

オプション	説明
ParquetTimestampInMillisecond	.parquet 形式で S3 オブジェクトに書き込まれるすべての TIMESTAMP 列値の精度を指定するオプションのパラメータ。 この属性を true または y に設定すると、AWS DMS はすべての TIMESTAMP 列をミリ秒の精度で .parquet 形式のファイルに書き込みます。それ以外の場合、DMS はそれらをマイクロ秒の精度で書き込みます。 現在、Amazon Athena と AWS Glue は、TIMESTAMP 値をミリ秒の精度でのみ処理できます。データを Athena または AWS Glue でクエリまたは処理する場合にのみ、.parquet 形式の S3 エンドポイント オブジェクトファイルに対してこの属性を true に設定します。 Note • AWS DMS は、マイクロ秒の精度で .csv 形式で S3 ファイルに書き込まれたすべての TIMESTAMP 列値を書き込みます。 • この属性の設定は、TimestampColumnName 属性を設定することで挿入されたタイムスタンプ列値の文字列の形式には影響しません。 デフォルト値: false 有効な値: true、false、y、n 例: --s3-settings '{"ParquetTimestampInMillisecond": true}'

オプション	説明
GlueCatalogGeneration	AWS Glue Data Catalog を生成するには、このエンドポイント設定を true に設定する。 デフォルト値: false 有効な値: true、false。 例: --s3-settings '{"GlueCatalogGeneration": true}' 注:GlueCatalogGeneration は PreserveTransactions と CdcPath と併用しない。

AWS DMS のための Amazon S3 ターゲットでの AWS Glue Data Catalog データカタログの使用

AWS Glue は、データのシンプルな分類方法を提供するサービスであり、AWS Glue Data Catalog と呼ばれるメタデータリポジトリで構成されています。AWS Glue Data Catalog を Amazon S3 ターゲットエンドポイントと統合し、Amazon Athena などのその他の AWS サービスを介して Amazon S3 データをクエリできます。Amazon Redshift AWS Glue と連携しますが、AWS DMS は事前構築オプションとしてサポートしていません。

データカタログを生成するには、次の AWS CLI 例のとおり、GlueCatalogGeneration エンドポイント設定を true に設定します。

```
aws dms create-endpoint --endpoint-identifier s3-target-endpoint
            --engine-name s3 --endpoint-type target--s3-settings
 '{"ServiceAccessRoleArn":
            "your-service-access-ARN", "BucketFolder": "your-bucket-folder",
 "BucketName":
            "your-bucket-name", "DataFormat": "parquet", "GlueCatalogGeneration":
 true}'
```

csv タイプのデータを含むフルロードレプリケーションタスクの場合は、IncludeOpForFullLoad を true に設定します。

GlueCatalogGeneration は PreserveTransactions と CdcPath と併用しないでください。AWS Glue クローラは、指定された CdcPath に保存されているファイルのさまざまなスキーマを調整できません。

Amazon Athena で Amazon S3 データのインデックスを作成して、Amazon Athena を介して標準 SQL クエリでデータをクエリするには、エンドポイントにアタッチされた IAM ロールに次のポリシーが必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::bucket123",
                "arn:aws:s3:::bucket123/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:GetDatabase",
                "glue:CreateTable",
                "glue:DeleteTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition"
            ],
            "Resource": [
```

```
                "arn:aws:glue:*:111122223333:catalog",
                "arn:aws:glue:*:111122223333:database/*",
                "arn:aws:glue:*:111122223333:table/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "athena:StartQueryExecution",
                "athena:GetQueryExecution",
                "athena:CreateWorkGroup"
            ],
            "Resource": "arn:aws:athena:*:111122223333:workgroup/
glue_catalog_generation_for_task_*"
        }
    ]
}
```

リファレンス

- AWS Glue の詳細については、AWS Glueデベロッパーガイドの「 の概念」を参照してください。
- AWS Glue Data Catalog の詳細については、「AWS Glue デベロッパーガイド」の「コンポーネント」を参照してください。

Amazon S3 ターゲットでのデータ暗号化、parquet ファイル、CDC の使用

S3 ターゲットエンドポイント設定を使用して、以下を構成できます。

- S3 ターゲットオブジェクトを暗号化するカスタム KMS キーです。
- S3 ターゲットオブジェクトのストレージ形式としての Parquet ファイルです。
- S3 ターゲットでのトランザクション順序を含むデータ キャプチャ (CDC) を変更します。
- AWS Glue Data Catalog データカタログを Amazon S3 ターゲットエンドポイントと統合し、Amazon Athena などのその他のサービスを介して Amazon S3 データをクエリする

AWS KMS データ暗号化に対する キー設定

次の例では、S3 ターゲットオブジェクトを暗号化するカスタム KMS キーの設定を示しています。開始するには、次の `create-endpoint` CLI コマンドを実行します。

```
aws dms create-endpoint --endpoint-identifier s3-target-endpoint --engine-name s3 --
endpoint-type target
--s3-settings '{"ServiceAccessRoleArn": "your-service-access-ARN", "CsvRowDelimiter":
 "\n",
"CsvDelimiter": ",", "BucketFolder": "your-bucket-folder",
"BucketName": "your-bucket-name",
"EncryptionMode": "SSE_KMS",
"ServerSideEncryptionKmsKeyId": "arn:aws:kms:us-
east-1:111122223333:key/72abb6fb-1e49-4ac1-9aed-c803dfcc0480"}'
```

ここでは、--s3-settings オプションで指定される JSON オブジェクトは 2 つのパラメータを定義します。1 つは、値が SSE_KMS の EncryptionMode パラメータです。もう 1 つは、arn:aws:kms:us-east-1:111122223333:key/72abb6fb-1e49-4ac1-9aed-c803dfcc0480 の値がある ServerSideEncryptionKmsKeyId パラメータです。この値は、カスタム KMS キーの Amazon リソースネーム (ARN) です。S3 ターゲットの場合、追加の設定も指定します。これによって、ロールへのサーバーアクセスの識別、デフォルトの CSV オブジェクトストレージ形式の区切り記号の提供、S3 ターゲットオブジェクトを保存するバケットの場所と名前の指定が行われます。

デフォルトでは、S3 データ暗号化は S3 サーバー側の暗号化を使用して行われます。前述の例の S3 ターゲットの場合、次の例に示すように、これはそのエンドポイント設定を指定することと同様です。

```
aws dms create-endpoint --endpoint-identifier s3-target-endpoint --engine-name s3 --
endpoint-type target
--s3-settings '{"ServiceAccessRoleArn": "your-service-access-ARN", "CsvRowDelimiter":
 "\n",
"CsvDelimiter": ",", "BucketFolder": "your-bucket-folder",
"BucketName": "your-bucket-name",
"EncryptionMode": "SSE_S3"}'
```

S3 サーバー側暗号化の使用の詳細については、「サーバー側の暗号化を使用したデータの保護」をご参照ください。

> **Note**
>
> CLI modify-endpoint コマンドを使用して、既存のエンドポイントの EncryptionMode のパラメータ値を SSE_KMS から SSE_S3 に変更するすることもできます。しかし EncryptionMode の値を SSE_S3 から SSE_KMS に変えることはできません。

S3 ターゲットオブジェクトを保存するために .parquet ファイルを使用する設定

S3 ターゲットオブジェクトを作成するデフォルトの形式は .csv ファイルです。次の例では、S3 ターゲットオブジェクトを作成する形式として .parquet ファイルを指定するいくつかのエンドポイント設定を示しています。次の例に示すように、すべてのデフォルトで .parquet ファイル形式を指定できます。

```
aws dms create-endpoint --endpoint-identifier s3-target-endpoint --engine-name s3 --
endpoint-type target
--s3-settings '{"ServiceAccessRoleArn": "your-service-access-ARN", "DataFormat":
 "parquet"}'
```

ここでは、DataFormat パラメータが parquet に設定されて、すべての S3 デフォルトでこの形式を有効化しています。これらのデフォルトには、繰り返し値を効率的に保存するためにビットパッキングおよびランレングスエンコードの組み合わせを使用するディクショナリエンコード ("EncodingType: "rle-dictionary") が含まれます。

次の例に示すように、デフォルト以外のオプションに追加の設定をさら加えることができます。

```
aws dms create-endpoint --endpoint-identifier s3-target-endpoint --engine-name s3 --
endpoint-type target
--s3-settings '{"ServiceAccessRoleArn": "your-service-access-ARN", "BucketFolder":
 "your-bucket-folder",
"BucketName": "your-bucket-name", "CompressionType": "GZIP", "DataFormat": "parquet",
 "EncodingType: "plain-dictionary", "DictPageSizeLimit": 3,072,000,
"EnableStatistics": false }'
```

ここでは、複数の標準 S3 バケットオプションおよび DataFormat パラメータに加えて、次のような追加の .parquet ファイルパラメータが設定されています。

- EncodingType - ディクショナリページの列チャンクごとで各列に発生する値を保存するディクショナリエンコード (plain-dictionary) を設定します。
- DictPageSizeLimit - ディクショナリページの最大のサイズを 3 MB に設定します。
- EnableStatistics - Parquet ファイルページおよび行グループに関する統計のコレクションを有効化するデフォルトを無効にします。

S3 ターゲットでのトランザクション順序を含むデータ変更 (CDC) のキャプチャ

デフォルトでは、AWS DMS がCDC タスクを実行する際は、ソースデータベース (またはデータベース) にログされたすべての行の変更を、各テーブルの 1 つ以上のファイルに保存します。同じテーブルに対する変更を含む各ファイル集合は、そのテーブルに関連付けられた単一のターゲットディレクトリに常駐します。AWS DMS は、Amazon S3 ターゲット エンドポイントに移行されたデータベーステーブルと同じ数のターゲットディレクトリを作成します。ファイルは、トランザクションの順序に関係なく、これらのディレクトリの S3 ターゲットに保存されます。ファイル命名規約、データコンテンツ、形式の詳細については、「AWS Database Migration Service のターゲットに Amazon S3 を使用する」をご参照ください。

トランザクション順序もキャプチャする方法でソースデータベースの変更をキャプチャするには、AWS DMS にすべてのデータベーステーブル行の変更をトランザクションサイズに応じて作成される 1 つ以上の .csv ファイルに保存するよう指示する S3 エンドポイント設定を指定できます。これらの.csv トランザクションファイルには各トランザクションに関連するすべてのテーブルについて、すべての行の変更がトランザクション順にリストされます。これらのトランザクションファイルは S3 ターゲットでも指定suru 1 つの[transaction directory] (トランザクションディレクトリ) に一括で常駐します。各トランザクション ファイルには、トランザクションオペレーションと、行変更ごとのデータベースとソーステーブルのアイデンティティが、次のとおり行データの一部として保存されます。

```
operation,table_name,database_schema_name,field_value,...
```

ここで、*operation* は、変更された行のトランザクションオペレーションで、*table_name* は、行が変更されるデータベーステーブルの名前、*database_schema_name* は、テーブルが常駐するデータベーススキーマの名前、*field_value* は、行のデータを指定する 1 つ以上のフィールド値の先頭です。

次のトランザクションファイルの例は、2 つのテーブルを含む 1 つ以上のトランザクションの変更された行を示しています。

```
I,Names_03cdcad11a,rdsTempsdb,13,Daniel
U,Names_03cdcad11a,rdsTempsdb,23,Kathy
D,Names_03cdcad11a,rdsTempsdb,13,Cathy
I,Names_6d152ce62d,rdsTempsdb,15,Jane
I,Names_6d152ce62d,rdsTempsdb,24,Chris
I,Names_03cdcad11a,rdsTempsdb,16,Mike
```

ここでは、各行のトランザクションオペレーションは I (挿入)、U (更新)、または D (削除) を最初の列に表示します。テーブル名は 2 番目のカラムの値です (例えば、Names_03cdcad11a)。データベーススキーマの名前は、3 列目の値です (例えば、rdsTempsdb)。残りの列には独自の行データが代入されます (例えば、13,Daniel)。

加えて、AWS DMS は、次の命名規則に従って、タイムスタンプを使用して Amazon S3 ターゲットで作成するトランザクションファイルに名前を付けます。

```
CDC_TXN-timestamp.csv
```

ここで、*timestamp* は、次の例のように、トランザクションファイルが作成された時刻です。

```
CDC_TXN-20201117153046033.csv
```

ファイル名にこのタイムスタンプを使用すると、トランザクションファイルがトランザクションディレクトリにリストされたときに、トランザクションファイルが作成され、トランザクション順にリストされます。

> **Note**
>
> トランザクションの順序でデータ変更を取り込む場合、AWS DMS は、ターゲットの DataFormat S3 設定値に関係なく、常に.csv ファイルに行の変更を保存します。AWS DMS は、.parquet ファイルを使用してトランザクション順序でデータの変更を保存しません。

データレプリケーションタスク中の Amazon S3 ターゲットへの書き込み頻度を制御するには、CdcMaxBatchInterval と CdcMinFileSize を設定します。これにより、追加のオーバーヘッド オペレーションなしでデータを分析する際のパフォーマンスが向上します。詳細については、「AWS DMS のターゲットとして Amazon S3を使用する場合のエンドポイントの設定」をご参照ください。

AWS DMS にすべての行の変更をトランザクション順序に保存するように指令するには

1. ターゲットの PreserveTransactions S3 設定を true に設定します。
2. ターゲットの CdcPath S3 設定を AWS DMS を保存したい目的の場所の相対フォルダパスに設定し、.csv トランザクションファイルを保存します。

AWS DMS は、このパスをデフォルトの S3 ターゲットバケットと作業ディレクトリ、またはターゲットの BucketName と BucketFolder S3 設定を使用して指定したバケットとバケットフォルダの下に作成します。

移行済み S3 データでのソース DB オペレーションの表示

AWS DMS は S3 ターゲットにレコードを移行するとき、移行済みレコードのそれぞれに追加のフィールドを作成できます。この追加のフィールドは、ソースデータベースでレコードに適用されたオペレーションを示します。AWS DMS がこの最初のフィールドを作成と設定する方法は、移行タスクのタイプとincludeOpForFullLoad、cdcInsertsOnly、cdcInsertsAndUpdates の設定によって異なります。

includeOpForFullLoad が true に指定されたフルロードの場合、AWS DMS は常に各 .csv レコードに追加の最初のフィールドを作成します。このフィールドには、ソースデータベースで行が挿入されたことを示す文字 I (INSERT) が含まれます。cdcInsertsOnly が false (デフォルト) に指定された CDC ロードの場合、AWS DMS は常に各 .csv レコードまたは .parquet レコードに追加の最初のフィールドを作成します。このフィールドには、ソースデータベースで行が挿入されたか、更新されたか、削除されたかを示す文字 I (INSERT)、U (UPDATE)、または D (DELETE) が含まれます。

次の表では、includeOpForFullLoad 属性と cdcInsertsOnly 属性の設定がどのように連動して、移行されたレコードの設定に影響を与えるかを示します。

使用するパラメータ設定		DMS が .csv 出力と .parquet 出力で設定するターゲットレコード	
includeOpForFullLoad	cdcInsertsOnly	全ロードの場合	CDC ロードの場合
true	true	最初のフィールド値が追加されて I に設定	最初のフィールド値が追加されて I に設定
false	false	追加のフィールドなし	最初のフィールド値が追加されて I、U、または D に設定

使用するパラメータ設定		DMS が .csv 出力と .parquet 出力で設定するターゲットレコード	
includeOpForFullLoad	cdcInsertsOnly	全ロードの場合	CDC ロードの場合
false	true	追加のフィールドなし	追加のフィールドなし
true	false	最初のフィールド値が追加されて I に設定	最初のフィールド値が追加されて I、U、または D に設定

includeOpForFullLoad と cdcInsertsOnly が同じ値に設定されている場合、ターゲットレコードは現在の移行タイプのレコード設定を制御する属性に従って設定されます。その属性は、全ロードの場合は includeOpForFullLoad、CDC ロードの場合は cdcInsertsOnly です。

includeOpForFullLoad と cdcInsertsOnly が異なる値に設定されている場合、AWS DMS によってターゲットレコードの設定は CDC と全ロードの両方に対して一貫したものになります。これは、CDC ロードのレコード設定を、includeOpForFullLoad で指定された前の全ロードのレコード設定と一致させることで行われます。

つまり、挿入されたレコードを示す最初のフィールドを追加するように全ロードが設定されているとします。この場合、続く CDC ロードは、挿入、更新、または削除されたレコードを示す最初のフィールドをソースで必要に応じて追加するように設定されます。逆に、挿入されたレコードを示す最初のフィールドを追加しないように全ロードが設定されているとします。この場合、CDC ロードも、ソースでの対応するレコードオペレーションに関係なく、各レコードに最初のフィールドを追加しないように設定されます。

同様に、DMS が追加の最初のフィールドを追加して設定する方法は、includeOpForFullLoad および cdcInsertsAndUpdates の設定によって異なります。以下の表では、includeOpForFullLoad および cdcInsertsAndUpdates 属性の設定がどのように連動して、この形式の移行済みレコードの設定に影響するかを確認できます。

使用するパラメータ設定		DMS が .csv 出力用に設定するターゲットレコード	
includeOpForFullLoad	cdcInsertsAndUpdates	全ロードの場合	CDC ロードの場合
true	true	最初のフィールド値が追加されて I に設定	最初のフィールド値が追加されて I または U に設定
false	false	追加のフィールドなし	最初のフィールド値が追加されて I、U、または D に設定
false	true	追加のフィールドなし	最初のフィールド値が追加されて I または U に設定
true	false	最初のフィールド値が追加されて I に設定	最初のフィールド値が追加されて I、U、または D に設定

S3 Parquet のターゲットデータ型

以下の表に、AWS DMS と AWS DMS データ型からのデフォルトマッピングをを使用する場合にサポートされる Parquet ターゲット データ型を示します。

AWS DMS のデータ型の詳細については、「AWS Database Migration Service のデータ型」をご参照ください。

AWS DMS データ型	S3 parquet データ型
BYTES	BINARY
DATE	DATE32
TIME	TIME32

AWS DMS データ型	S3 parquet データ型
DATETIME	TIMESTAMP
INT1	INT8
INT2	INT16
INT4	INT32
INT8	INT64
NUMERIC	DECIMAL
REAL4	FLOAT
REAL8	DOUBLE
STRING	STRING
UINT1	UINT8
UINT2	UINT16
UINT4	UINT32
UINT8	UINT64
WSTRING	STRING
BLOB	BINARY
NCLOB	STRING
CLOB	STRING
BOOLEAN	BOOL

AWS Database Migration Service のターゲットとしての Amazon DynamoDB データベースの使用

Amazon DynamoDB のテーブルにデータを移行するために、AWS DMS を使用できます。Amazon DynamoDB はシームレスなスケーラビリティが可能な高速で予測可能なパフォーマンスを発揮するフルマネージド NoSQL データベースサービスです。AWS DMS はソースとしてリレーショナルデータベースや MongoDB を使用して対応します。

DynamoDB では、テーブル、項目、および属性が、操作するコアコンポーネントです。[table] (テーブル) とは項目の集合であり、各 [item] (項目) は属性の集合です。DynamoDB は、テーブルの各項目を一意に識別するために、パーティションキーと呼ばれるプライマリキーを使用します。より柔軟なクエリを提供するために、キーおよびセカンダリインデックスを使用することもできます。

ソースのデータベースから、ターゲット DynamoDB テーブルにデータを移行するために、オブジェクトのマッピングを使用します。オブジェクトのマッピングを使用すると、ソースデータがターゲットのどこにあるか判定できます。

AWS DMS が DynamoDB ターゲット エンドポイントにテーブルを作成する際、ソースデータベースのエンドポイントと同じ数のテーブルを作成します。また、AWS DMS はいくつかの DynamoDB パラメータ値も設定します。テーブル作成のコストは、データの量および移行するテーブルの数によって異なります。

> **Note**
>
> AWS DMS コンソールまたは API の [SSL モード] オプションは、Kinesis や DynamoDB などの一部のデータ ストリーミングや NoSQL サービスには適用されません。これらはデフォルトで安全なため、AWS DMS では SSL モードの設定が none と等しい ([SSL Mode=None]) と表示されます。SSL を使用するために、エンドポイントに追加の設定は必要ありません。例えば、DynamoDB をターゲットエンドポイントとして使用する場合、デフォルトで保護されます。DynamoDB への API コールはすべて SSL を使用するため、AWS DMS エンドポイントに追加の SSL オプションは不要です。AWS DMS が DynamoDB データベースへの接続時にデフォルトで使用する HTTPS プロトコルを使用して、SSL エンドポイント経由で安全にデータを保存して取得できます。

転送の速度を高めるために、AWS DMS は DynamoDB ターゲットインスタンスへのマルチスレッド全ロードに対応しています。DMS では、このマルチスレッドでのタスク設定を、以下でサポートします。

- `MaxFullLoadSubTasks` - 並列ロードするソーステーブルの最大数を指定するにはこのオプションを使用します。DMS は、専用のサブタスクを使用して、対応する DynamoDB ターゲットテーブルに各テーブルをロードします。デフォルト値は 8 です。最大値は 49 です。
- `ParallelLoadThreads` - AWS DMS が各テーブルを DynamoDB ターゲットテーブルにロードするために使用するスレッド数を指定するには、このオプションを使用します。デフォルト値は 0 (シングルスレッド) です。最大値は 200 です。この上限を増やすよう依頼できます。

> **Note**
>
> DMS は、テーブルの各セグメントを独自のスレッドに割り当てロードします。したがって、`ParallelLoadThreads` をソースのテーブルに指定するセグメントの最大数に設定します。

- `ParallelLoadBufferSize` – DynamoDB ターゲットにデータをロードするために並列ロードスレッドが使用するバッファ内に保存するレコードの最大数を指定するには、このオプションを使用します。デフォルト値は 50 です。最大値は 1000 です。この設定は `ParallelLoadThreads` で使用します。`ParallelLoadBufferSize` は、複数のスレッドがある場合にのみ有効です。
- 個々のテーブルのテーブルマッピング設定 - `table-settings` ルールを使用して、並列ロードするソースから個々のテーブルを識別します。また、これらのルールを使用して、各テーブルの行をマルチスレッドロード用にセグメント化する方法を指定します。詳細については、「テーブルとコレクション設定のルールとオペレーション」をご参照ください。

> **Note**
>
> AWS DMS が移行タスクの DynamoDB パラメータ値を設定する場合、デフォルトの読み取り容量単位 (RCU) パラメータ値が 200 に設定されます。
> 書き込みキャパティティーユニット (WCU) のパラメータ値も設定されますが、その値は他のいくつかの設定によって異なります。
>
> - WCU パラメータのデフォルト値は 200 です。
> - `ParallelLoadThreads` タスク設定が 1 より大きい値に設定された場合 (デフォルトは 0)、WCU パラメータは `ParallelLoadThreads` 値の 200 倍に設定されます。
> - スタンダードの AWS DMS 使用料は使用するリソースに適用されます。

リレーショナルデータベースから DynamoDB テーブルへの移行

AWS DMS は、DynamoDB のスカラーデータ型へのデータの移行をサポートしています。Oracle や MySQL などのリレーショナルデータベースから DynamoDB に移行する場合は、データを格納する方法を再編成が必要となる場合があります。

現在 AWS DMS は、単一テーブルから単一テーブルに移行する場合の、DynamoDB のスカラー型属性への再構成をサポートしています。リレーショナルデータベースのテーブルから DynamoDB にデータを移行する場合、テーブルからデータを取得し、DynamoDB のスカラーデータ型属性に形式を変更します。これらの属性は複数の列からデータを受け入れることができるため、列を属性に直接マッピングすることができます。

AWS DMS は以下の DynamoDB のスカラーデータ型をサポートしています。

- 文字列
- 数
- ブール値

> **Note**
>
> ソースからの NULL データは、ターゲットで無視されます。

AWS Database Migration Service のターゲットとして DynamoDB を使用する場合の前提条件

AWS DMS のターゲットとして DynamoDBデータベースの使用を開始する前に、IAM ロールを必ず作成します。この IAM ロールにより、AWS DMS は移行先の DynamoDB テーブルへのアクセスを引き受け、許可できるようになります。アクセス許可の最小設定は、次の IAM ポリシーで示します。

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

```
            "Service": "dms.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
      }
   ]
}
```

DynamoDB に移行する際使用するロールには、次のアクセス許可が必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
        "dynamodb:PutItem",
        "dynamodb:CreateTable",
        "dynamodb:DescribeTable",
        "dynamodb:DeleteTable",
        "dynamodb:DeleteItem",
        "dynamodb:UpdateItem"
    ],
    "Resource": [
    "arn:aws:dynamodb:us-west-2:account-id:table/name1",
    "arn:aws:dynamodb:us-west-2:account-id:table/OtherName*",
    "arn:aws:dynamodb:us-west-2:account-id:table/awsdms_apply_exceptions",
    "arn:aws:dynamodb:us-west-2:account-id:table/awsdms_full_load_exceptions"
    ]
    },
    {
        "Effect": "Allow",
        "Action": [
        "dynamodb:ListTables"
        ],
        "Resource": "*"
    }
    ]
}
```

AWS Database Migration Service のターゲットとして DynamoDB を使用する場合の制限

ターゲットとして DynamoDB を使用する場合、以下の制限が適用されます：

- DynamoDB は、数値データ型の最大精度を 38 に制限します。文字列として高い精度のすべてのデータ型を保存します。オブジェクトマッピング機能を使用して、これを明示的に指定する必要があります。
- DynamoDB に Date データ型がないため、Date データ型を使用しているデータは、文字列に変換されます。
- DynamoDB はプライマリ キー属性の更新を許可しません。この制限は、ターゲットで不要なデータが発生する可能性があるため、変更データキャプチャ (CDC) で継続的なレプリケーションを使用する場合に重要です。オブジェクトのマッピング方法に応じて、プライマリキーを更新する CDC オペレーションは 2 つのうちのいずれかを実行できます。これは、更新されたプライマリキーおよび不完全なデータがある新しい項目を挿入できるか、あるいは失敗するかのいずれかです。
- AWS DMS は、非複合プライマリキーを含むテーブルのレプリケーションのみをサポートします。例外は、カスタムパーティションキーまたはソートキー、あるいはその両方があるターゲットテーブルにオブジェクトマッピングを指定する場合です。
- AWS DMS は、CLOB でない限り LOB データには対応しません。AWS DMS は、データの移行時に CLOB データを DynamoDB 文字列に変換します。
- ターゲットとして DynamoDB を使用する場合、例外適用制御テーブル (`dmslogs.awsdms_apply_exceptions`) のみがサポートされます。統制テーブルの詳細については、「制御テーブルタスク設定」をご参照ください。
- AWS DMS は、ターゲットとしての DynamoDB `TargetTablePrepMode=TRUNCATE_BEFORE_LOAD` のタスク設定をサポートしていません。
- AWS DMS は、ターゲットとしての DynamoDB `TaskRecoveryTableEnabled` のタスク設定をサポートしていません。

DynamoDB にデータを移行するためのオブジェクトマッピングの使用

AWS DMS は、ソースからターゲット DynamoDB テーブルにデータをマッピングするためのテーブルマッピングルールを使用します。DynamoDB ターゲットにデータをマッピングするために、object-mapping と呼ばれるテーブルマッピングルールのタイプを使用します。オブジェクトマッピングにより、移行するデータの属性名とデータを定義できます。オブジェクトマッピングを使用するときは選択ルールが必要です。

DynamoDB には、パーティションのキーとオプションのソートキー以外に、プリセット構造はありません。非複合プライマリキーが存在する場合は、AWS DMS はそのキーを使用します。複合プラ

イマリキーがある場合、またはソートキーを使用する必要がある場合は、ターゲット DynamoDB テーブルでそれらのキーと他の属性を定義します。

オブジェクトマッピングルールを作成するには、rule-type を object-mapping として指定します。このルールが、使用したいオブジェクトマッピングのタイプを指定します。

ルールの構造は次のとおりです。

```
{ "rules": [
    {
      "rule-type": "object-mapping",
      "rule-id": "<id>",
      "rule-name": "<name>",
      "rule-action": "<valid object-mapping rule action>",
      "object-locator": {
      "schema-name": "<case-sensitive schema name>",
      "table-name": ""
      },
      "target-table-name": "<table_name>"
    }
  ]
}
```

AWS DMS では現在、rule-action パラメータに対する有効な値として map-record-to-record および map-record-to-document のみがサポートされています。これらの値は、exclude-columns 属性リストの一部として除外されてないものとして AWS DMS がデフォルトで記録するものを指定します。これらの値は、どのような方法でも属性マッピングに影響しません。

- リレーショナルデータベースから DynamoDB に移行するときに map-record-to-record を使用できます。DynamoDB のパーティションキーとしてリレーショナルデータベースからプライマリ キーを使用し、ソースデータベース内の各列の属性を作成します。map-record-to-record を使用する場合、exclude-columns 属性リストに示されていないソーステーブル内の任意の列について、AWS DMS はターゲット DynamoDB インスタンスに対応する属性を作成します。これは、そのソース列が属性マッピングで使用されているかどうかにかかわらず作成されます。
- map-record-to-document を使用して、ターゲットの 1 つのフラット DynamoDB マップに "_doc." 属性名でソース列を配置します。map-record-to-document を使用する場合、AWS DMS はソースの 1 つのフラットな DynamoDB マップ属性にデータを配置します。この属性は "_doc." と呼ばれます。この配置は、exclude-columns 属性リストに含まれていないソーステーブル内のすべての列に適用されます。

rule-action パラメータの map-record-to-record と map-record-to-document の違いを理解する 1 つの方法は、この 2 つのパラメータの動作を確認することです。この例では、次の構造とデータを含むリレーショナルデータベースのテーブルから始めると想定してください。

FirstName	LastName	NickName	WorkAddress	WorkPhone	HomeAddress	HomePhone	income
Daniel	Sheridan	Dan	101 Main St Cambridge, MA	800-867-5309	100 Secret St, Unknownville, MA	123-456-7890	12345678

この情報を DynamoDB に移行するには、データを DynamoDB テーブルにマッピングするルールを作成します。パラメータにリストされている exclude-columns 列を書き留めてください。これらの列はターゲットに直接マッピングされていません。その代わりに、データを組み合わせて新しい項目を作成するために、属性マッピングが使用されています。たとえば、FirstName と LastName を組み合わせると DynamoDB ターゲット上の CustomerName になります。NickName と income は除外されていません。

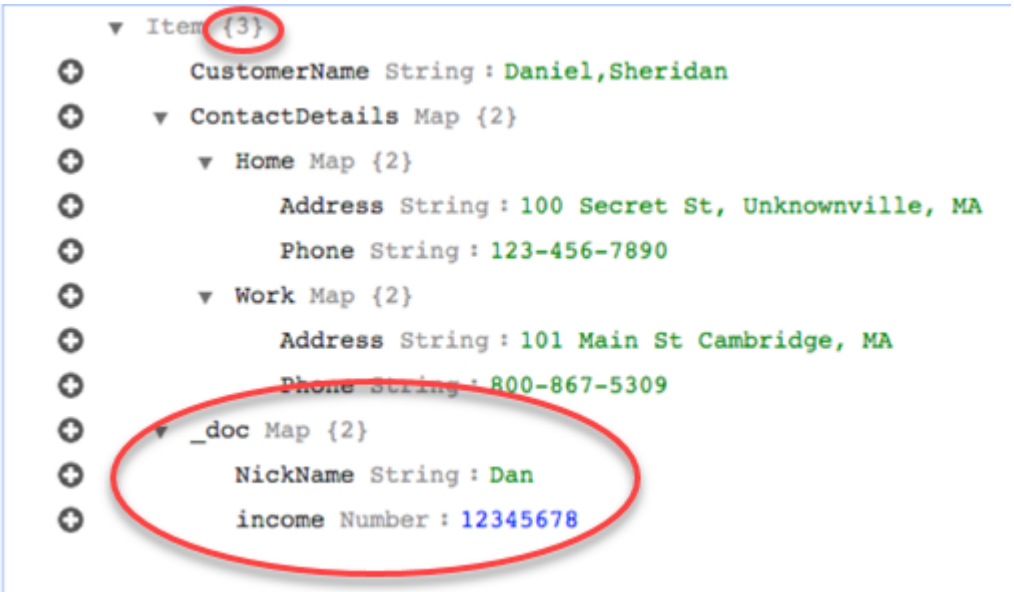
```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",
            "rule-name": "TransformToDDB",
            "rule-action": "map-record-to-record",
            "object-locator": {
                "schema-name": "test",
                "table-name": "customer"
            },
            "target-table-name": "customer_t",
            "mapping-parameters": {
                "partition-key-name": "CustomerName",
                "exclude-columns": [
                    "FirstName",
                    "LastName",
                    "HomeAddress",
```

```
                    "HomePhone",
                    "WorkAddress",
                    "WorkPhone"
                ],
                "attribute-mappings": [
                    {
                        "target-attribute-name": "CustomerName",
                        "attribute-type": "scalar",
                        "attribute-sub-type": "string",
                        "value": "${FirstName},${LastName}"
                    },
                    {
                        "target-attribute-name": "ContactDetails",
                        "attribute-type": "document",
                        "attribute-sub-type": "dynamodb-map",
                        "value": {
                            "M": {
                                "Home": {
                                    "M": {
                                        "Address": {
                                            "S": "${HomeAddress}"
                                        },
                                        "Phone": {
                                            "S": "${HomePhone}"
                                        }
                                    }
                                },
                                "Work": {
                                    "M": {
                                        "Address": {
                                            "S": "${WorkAddress}"
                                        },
                                        "Phone": {
                                            "S": "${WorkPhone}"
                                        }
                                    }
                                }
                            }
                        }
                    }
                ]
            }
        }
    ]
```

```
}
```

`rule-action` パラメータ map-record-to-recordを使用することで、NickName および [income] (所得) のデータは、DynamoDB ターゲットで同じ名前の項目にマッピングされます。

ただし、同じルールを使用しますが `rule-action` パラメータを map-record-to-document に変更するとします。この場合、`exclude-columns` パラメータ NickName および [income] (所得) にリストされない列は、_doc 項目にマッピングされます。



オブジェクトマッピングでのカスタム条件式の使用

DynamoDB テーブルに書き込み中のデータを操作するための条件式と呼ばれる DynamoDB の機能を使用することができます。DynamoDB の条件式の詳細については、「条件式」をご参照ください。

条件式のメンバーは次から構成されます。

- 式 (必須)
- 式の属性値 (オプション) 属性値の DynamoDB json 構造を指定します
- 式の属性名 (オプション)
- 条件式をいつ使用するかを選ぶオプション (オプション) デフォルトは apply-during-cdc = false および apply-during-full-load = true

ルールの構造は次のとおりです。

```
"target-table-name": "customer_t",
      "mapping-parameters": {
        "partition-key-name": "CustomerName",
        "condition-expression": {
          "expression":"<conditional expression>",
          "expression-attribute-values": [
              {
                "name":"<attribute name>",
                "value":<attribute value>
              }
          ],
          "apply-during-cdc":<optional Boolean value>,
          "apply-during-full-load": <optional Boolean value>
        }
```

次のサンプルは、条件式に使用されるセクションを主に示しています。


```
{
  "rules": [
    {
      "rule-type": "object-mapping",
      "rule-id": "1",
      "rule-name": "TransformToDDB",
      "rule-action": "map-record-to-record",
      "object-locator": {
        "schema-name": "test",
        "table-name": "customer",

      },
      "target-table-name": "customer_t",
      "mapping-parameters": {
        "partition-key-name": "CustomerName",
        "condition-expression": {
          "expression":"attribute_not_exists(version) or version <= :record_version",
          "expression-attribute-values": [
              {
                "name":":record_version",
                "value":{"N":"${version}"}
              }
          ],
          "apply-during-cdc":true,
          "apply-during-full-load": true
        }
        "attribute-mappings": [
          {
            "target-attribute-name": "CustomerName",
            "attribute-type": "scalar",
            "attribute-sub-type": "string",
            "value": "${FirstName},${LastName}"
          }
        ]
      }
    }
  ]
}
```

オブジェクトマッピングで属性マッピングを使用する

属性マッピングでは、ターゲット上のデータを再編成するために、ソース列名を使用してテンプレート文字列を指定することができます。ユーザーがテンプレートで指定する場合を除き、書式設定は行われません。

次の例は、ソースデータベースの構造と、DynamoDB ターゲットの必要とされる構造を示します。最初に示すのは、ソースの構造で、この場合は Oracle データベースです。次に DynamoDB 内のデータの必要とされる構造を示します。この例では最後に、必要なターゲット構造を作成するのに使用される JSON を示します。

Oracle データの構造は次のとおりです。

First	Las	St	Home/ ss	HomeF	WorkAddre ss	Work	DateOfBirth
プライマリキー				該当なし			
Ran	Mar	5	221B Baker Street	123456 0	31 Spooner Street, Quahog	9876 0	02/29/1988

DynamoDB データの構造は次のとおりです。

Custor ame	StoreId	ContactDetails	DateOfBirth
パーティションキー	ソートキー	該当なし	
Randy sh	5	{ "Name": "Randy", "Home": { "Address": "221B Baker Street", "Phone": 1234567890 }, "Work": { "Address": "31 Spooner Street, Quahog", "Phone": 9876541230 } }	02/29/1988

次の JSON は、DynamoDB 構造を達成するために使用されるオブジェクトマッピングと列マッピングを示します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",
            "rule-name": "TransformToDDB",
            "rule-action": "map-record-to-record",
            "object-locator": {
                "schema-name": "test",
                "table-name": "customer"
            },
            "target-table-name": "customer_t",
            "mapping-parameters": {
                "partition-key-name": "CustomerName",
                "sort-key-name": "StoreId",
                "exclude-columns": [
                    "FirstName",
                    "LastName",
                    "HomeAddress",
                    "HomePhone",
                    "WorkAddress",
                    "WorkPhone"
                ],
                "attribute-mappings": [
                    {
                        "target-attribute-name": "CustomerName",
                        "attribute-type": "scalar",
                        "attribute-sub-type": "string",
                        "value": "${FirstName},${LastName}"
```

```
                    },
                    {
                        "target-attribute-name": "StoreId",
                        "attribute-type": "scalar",
                        "attribute-sub-type": "string",
                        "value": "${StoreId}"
                    },
                    {
                        "target-attribute-name": "ContactDetails",
                        "attribute-type": "scalar",
                        "attribute-sub-type": "string",
                        "value": "{\"Name\":\"${FirstName}\",\"Home\":{\"Address
\":\"${HomeAddress}\",\"Phone\":\"${HomePhone}\"}, \"Work\":{\"Address\":
\"${WorkAddress}\",\"Phone\":\"${WorkPhone}\"}}"
                    }
                ]
            }
        }
    ]
}
```

列マッピングを使用するもう 1 つの方法は、ドキュメントタイプとして DynamoDB 形式を使用することです。次のコード例では、attribute-sub-type として dynamodb map を属性マッピングに使用します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",
            "rule-name": "TransformToDDB",
```

```
            "rule-action": "map-record-to-record",
            "object-locator": {
                "schema-name": "test",
                "table-name": "customer"
            },
            "target-table-name": "customer_t",
            "mapping-parameters": {
                "partition-key-name": "CustomerName",
                "sort-key-name": "StoreId",
                "exclude-columns": [
                    "FirstName",
                    "LastName",
                    "HomeAddress",
                    "HomePhone",
                    "WorkAddress",
                    "WorkPhone"
                ],
                "attribute-mappings": [
                    {
                        "target-attribute-name": "CustomerName",
                        "attribute-type": "scalar",
                        "attribute-sub-type": "string",
                        "value": "${FirstName},${LastName}"
                    },
                    {
                        "target-attribute-name": "StoreId",
                        "attribute-type": "scalar",
                        "attribute-sub-type": "string",
                        "value": "${StoreId}"
                    },
                    {
                        "target-attribute-name": "ContactDetails",
                        "attribute-type": "document",
                        "attribute-sub-type": "dynamodb-map",
                        "value": {
                          "M": {
                            "Name": {
                              "S": "${FirstName}"
                            },
                            "Home": {
                                    "M": {
                                        "Address": {
                                            "S": "${HomeAddress}"
                                        },
```

```
                                    "Phone": {
                                        "S": "${HomePhone}"
                                    }
                                }
                            },
                            "Work": {
                                "M": {
                                    "Address": {
                                        "S": "${WorkAddress}"
                                    },
                                    "Phone": {
                                        "S": "${WorkPhone}"
                                    }
                                }
                            }
                        }
                    }
                ]
            }
        }
    ]
}
```

次の例のように、dynamodb-map の代わりに dynamodb-list を属性マッピングの属性サブタイプとして使用することもできます。

```
{
"target-attribute-name": "ContactDetailsList",
"attribute-type": "document",
"attribute-sub-type": "dynamodb-list",
"value": {
    "L": [
            {
                "N": "${FirstName}"
            },
            {
                "N": "${HomeAddress}"
            },
            {
                "N": "${HomePhone}"
```

```
            },
            {
                "N": "${WorkAddress}"
            },
            {
                "N": "${WorkPhone}"
            }
        ]
    }
}
```

例1: オブジェクトマッピングで属性マッピングを使用する

次の例は、2 つの MySQL データベーステーブル nfl_data と sport_team のデータを、NFLTeams と SportTeams という 2 つの DynamoDB テーブルへ移行します。テーブルの構造、および MySQL データベーステーブルから DynamoDB テーブルへデータをマップするために使用される JSON は、次のとおりです。

MySQL データベース nfl_data の構造は次のとおりです。

```
mysql> desc nfl_data;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| Position      | varchar(5)  | YES  |     | NULL    |       |
| player_number | smallint(6) | YES  |     | NULL    |       |
| Name          | varchar(40) | YES  |     | NULL    |       |
| status        | varchar(10) | YES  |     | NULL    |       |
| stat1         | varchar(10) | YES  |     | NULL    |       |
| stat1_val     | varchar(10) | YES  |     | NULL    |       |
| stat2         | varchar(10) | YES  |     | NULL    |       |
| stat2_val     | varchar(10) | YES  |     | NULL    |       |
| stat3         | varchar(10) | YES  |     | NULL    |       |
| stat3_val     | varchar(10) | YES  |     | NULL    |       |
| stat4         | varchar(10) | YES  |     | NULL    |       |
| stat4_val     | varchar(10) | YES  |     | NULL    |       |
| team          | varchar(10) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

MySQL データベーステーブルsport_team の構造は次のとおりです。

```
mysql> desc sport_team;
+--------------------------+-------------+------+-----+---------+----------------+
| Field                    | Type        | Null | Key | Default | Extra          |
+--------------------------+-------------+------+-----+---------+----------------+
| id                       | mediumint(9) | NO   | PRI | NULL    | auto_increment |
| name                     | varchar(30)  | NO   |     | NULL    |                |
| abbreviated_name         | varchar(10)  | YES  |     | NULL    |                |
| home_field_id            | smallint(6)  | YES  | MUL | NULL    |                |
| sport_type_name          | varchar(15)  | NO   | MUL | NULL    |                |
| sport_league_short_name  | varchar(10)  | NO   |     | NULL    |                |
| sport_division_short_name | varchar(10)  | YES  |     | NULL    |                |
```

2 つのテーブルを 2 つの DynamoDB テーブルにマッピングするために使用されるテーブルマッピングルールは次のとおりです。

```
{
  "rules":[
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "dms_sample",
        "table-name": "nfl_data"
      },
      "rule-action": "include"
    },
    {
      "rule-type": "selection",
      "rule-id": "2",
      "rule-name": "2",
      "object-locator": {
        "schema-name": "dms_sample",
        "table-name": "sport_team"
      },
      "rule-action": "include"
    },
    {
      "rule-type":"object-mapping",
      "rule-id":"3",
```

```
      "rule-name":"MapNFLData",
      "rule-action":"map-record-to-record",
      "object-locator":{
        "schema-name":"dms_sample",
        "table-name":"nfl_data"
      },
      "target-table-name":"NFLTeams",
      "mapping-parameters":{
        "partition-key-name":"Team",
        "sort-key-name":"PlayerName",
        "exclude-columns": [
          "player_number", "team", "name"
        ],
        "attribute-mappings":[
          {
            "target-attribute-name":"Team",
            "attribute-type":"scalar",
            "attribute-sub-type":"string",
            "value":"${team}"
          },
          {
            "target-attribute-name":"PlayerName",
            "attribute-type":"scalar",
            "attribute-sub-type":"string",
            "value":"${name}"
          },
          {
            "target-attribute-name":"PlayerInfo",
            "attribute-type":"scalar",
            "attribute-sub-type":"string",
            "value":"{\"Number\": \"${player_number}\",\"Position\": \"${Position}\",
\"Status\": \"${status}\",\"Stats\": {\"Stat1\": \"${stat1}:${stat1_val}\",\"Stat2\":
 \"${stat2}:${stat2_val}\",\"Stat3\": \"${stat3}:${
stat3_val}\",\"Stat4\": \"${stat4}:${stat4_val}\"}"
          }
        ]
      }
    },
    {
      "rule-type":"object-mapping",
      "rule-id":"4",
      "rule-name":"MapSportTeam",
      "rule-action":"map-record-to-record",
      "object-locator":{
```

```
        "schema-name":"dms_sample",
        "table-name":"sport_team"
      },
      "target-table-name":"SportTeams",
      "mapping-parameters":{
        "partition-key-name":"TeamName",
        "exclude-columns": [
          "name", "id"
        ],
        "attribute-mappings":[
          {
            "target-attribute-name":"TeamName",
            "attribute-type":"scalar",
            "attribute-sub-type":"string",
            "value":"${name}"
          },
          {
            "target-attribute-name":"TeamInfo",
            "attribute-type":"scalar",
            "attribute-sub-type":"string",
            "value":"{\"League\": \"${sport_league_short_name}\",\"Division\":
 \"${sport_division_short_name}\"}"
          }
        ]
      }
    }
  ]
}
```

NFLTeams DynamoDB テーブルの出力例は次のとおりです。

```
 "PlayerInfo": "{\"Number\": \"6\",\"Position\": \"P\",\"Status\": \"ACT\",\"Stats\":
{\"Stat1\": \"PUNTS:73\",\"Stat2\": \"AVG:46\",\"Stat3\": \"LNG:67\",\"Stat4\": \"IN
20:31\"}",
 "PlayerName": "Allen, Ryan",
 "Position": "P",
 "stat1": "PUNTS",
 "stat1_val": "73",
 "stat2": "AVG",
 "stat2_val": "46",
 "stat3": "LNG",
```

```
  "stat3_val": "67",
  "stat4": "IN 20",
  "stat4_val": "31",
  "status": "ACT",
  "Team": "NE"
}
```

SportsTeams DynamoDB テーブルの出力例は次のとおりです。

```
{
  "abbreviated_name": "IND",
  "home_field_id": 53,
  "sport_division_short_name": "AFC South",
  "sport_league_short_name": "NFL",
  "sport_type_name": "football",
  "TeamInfo": "{\"League\": \"NFL\",\"Division\": \"AFC South\"}",
  "TeamName": "Indianapolis Colts"
}
```

DynamoDB のターゲットデータ型

AWS DMS の DynamoDB エンドポイントでは、DynamoDB のほとんどのデータ型がサポートされます。以下の表に、AWS DMS を使用する場合にサポートされる Amazon AWS DMS のターゲットデータ型と、AWS DMS のデータ型からのデフォルトマッピングを示します。

AWS DMS のデータ型の詳細については、「AWS Database Migration Service のデータ型」をご参照ください。

AWS DMS が異なるデータベースからデータを移行するときは、ソースデータベースからのデータ型を、AWS DMS データ型という中間のデータ型にマッピングします。その後、中間データ型をターゲットデータ型にマッピングします。以下の表は、各 AWS DMS データ型と DynamoDB でのマッピング先のデータ型を示しています。

AWS DMS データ型	DynamoDB データ型
文字列	文字列
WString	文字列

AWS DMS データ型	DynamoDB データ型
ブール値	ブール値
日付	文字列
DateTime	文字列
INT1	数
INT2	数
INT4	数
INT8	数
数値	数
Real4	数
Real8	数
UINT1	数
UINT2	数
UINT4	数
UINT8	数
CLOB	文字列

のターゲットとしての Amazon Kinesis Data Streams の使用 AWS Database Migration Service

を使用して AWS DMS 、Amazon Kinesis データストリームにデータを移行できます。Amazon Kinesis Data Streams サービスは、Amazon Kinesis Data Streams サービスの一部です。データレコードの大量のストリームをリアルタイムで収集し、処理するには、Kinesis データストリームを使用します。

Kinesis データストリームはシャードで構成されています。シャードは、ストリーム内のデータレコードの一意に識別されたシーケンスです。Amazon Kinesis Data Streams におけるシャードの詳細については、Amazon Kinesis Data Streams デベロッパーガイドの「シャード」をご参照ください。

AWS Database Migration Service は、JSON を使用してレコードを Kinesis データストリームに発行します。変換時に、 AWS DMS はソースデータベースから各レコードを JSON 形式または JSON_UNFORMATTED メッセージ形式の属性と値のペアにシリアル化します。JSON_UNFORMATTED メッセージ形式は、改行区切り文字を含む単一行の JSON 文字列です。これにより、Amazon Data Firehose は Kinesis データを Amazon S3 の送信先に配信し、Amazon Athena を含むさまざまなクエリエンジンを使用してクエリを実行できます。

サポートされている任意のデータソースから、ターゲットストリームにデータを移行するために、オブジェクトのマッピングを使用します。オブジェクトマッピングを使用して、ストリームにデータレコードを構築する方法を決定します。データをそのシャードにグループ化するために Kinesis Data Streams で使用する、各テーブルのパーティション キーも定義します。

AWS DMS が Kinesis Data Streams ターゲットエンドポイントにテーブルを作成する場合、ソースデータベースエンドポイントと同じ数のテーブルを作成します。 AWS DMS また、 は複数の Kinesis Data Streams パラメータ値も設定します。テーブル作成のコストは、データの量および移行するテーブルの数によって異なります。

> **Note**
>
> AWS DMS コンソールまたは API の SSL モードオプションは、一部のデータストリーミングや、Kinesis や DynamoDB などの NoSQL サービスには適用されません。これらはデフォルトで安全であるため、SSL モードの設定が none (SSL Mode=None) と等しい AWS DMS ことを示しています。SSL を使用するために、エンドポイントに追加の設定は必要ありません。例えば、Kinesis をターゲット エンドポイントとして使用する場合、デフォルトでセキュリティで保護されます。Kinesis へのすべての API コールは SSL を使用するため、 AWS DMS エンドポイントに追加の SSL オプションは必要ありません。 AWS DMS が Kinesis Data Stream への接続時にデフォルトで使用する HTTPS プロトコルを使用して、SSL エンドポイント経由で安全にデータを保存して取得できます。

Kinesis Data Streams エンドポイント設定

Kinesis Data Streams ターゲットエンドポイントを使用すると、 AWS DMS API の `KinesisSettings`オプションを使用してトランザクションとコントロールの詳細を取得できます。

接続は、次のいずれかの方法で設定できます。

- AWS DMS コンソールで、エンドポイント設定を使用します。
- CLI では、 CreateEndpoint コマンドの kinesis-settingsオプションを使用します。

CLI で、次の kinesis-settings オプションのリクエストパラメータを使用します。

> **Note**
>
> AWS DMS バージョン 3.4.1 以降では、IncludeNullAndEmpty エンドポイント設定に対応しています。ただし、Kinesis Data Streams ターゲットの他のエンドポイント設定のサポートは で利用できます AWS DMS。

- MessageFormat - エンドポイントで作成されたレコードの出力形式。メッセージ形式は JSON (デフォルト) または JSON_UNFORMATTED (タブなし 1 行) です。
- IncludeControlDetails - Kinesis メッセージ出力に、テーブル定義、列定義、テーブル、列の変更の詳細な制御情報を表示します。デフォルトは false です。
- IncludeNullAndEmpty — ターゲットに NULL 列と空の列を含めます。デフォルトは false です。
- IncludePartitionValue – パーティションタイプが schema-table-type でない限り、Kinesis メッセージ出力内のパーティション値を表示します。デフォルトは false です。
- IncludeTableAlterOperations – rename-table、drop-table、add-column、drop-column、rename-column など、制御データのテーブルを変更するデータ定義言語 (DDL) オペレーションが含まれます。デフォルトは false です。
- IncludeTransactionDetails - ソースデータベースからの詳細のトランザクション情報を提供します。この情報には、コミットタイムスタンプ、ログの位置、transaction_id、previous_transaction_id、および transaction_record_id (トランザクション内のレコードオフセット) の値が含まれます。デフォルトは false です。
- PartitionIncludeSchemaTable パーティションタイプが primary-key-type の場合、スキーマとテーブル名をパーティション値にプレフィックスします。これにより、Kinesis シャード間のデータ分散が増加します。例えば、SysBench スキーマに数千のテーブルがあり、各テーブルのプライマリ キーの範囲が制限されているとします。この場合、同じプライマリキーが数千のテーブルから同じシャードに送信され、スロットリングが発生します。デフォルトは false です。

次の例で、 AWS CLI を使用して発行された例 create-endpointコマンドを使用中の kinesis-settings オプションを示します。

```
aws dms create-endpoint --endpoint-identifier=$target_name --engine-name kinesis --
endpoint-type target
--region us-east-1 --kinesis-settings
 ServiceAccessRoleArn=arn:aws:iam::333333333333:role/dms-kinesis-role,
StreamArn=arn:aws:kinesis:us-east-1:333333333333:stream/dms-kinesis-target-
doc,MessageFormat=json-unformatted,
IncludeControlDetails=true,IncludeTransactionDetails=true,IncludePartitionValue=true,PartitionI
IncludeTableAlterOperations=true
```

マルチスレッド全ロードタスク設定

転送の速度を上げるために、 は Kinesis Data Streams ターゲットインスタンスへのマルチスレッド全ロード AWS DMS をサポートします。DMS では、このマルチスレッドでのタスク設定を、以下でサポートします。

- MaxFullLoadSubTasks - 並列ロードするソーステーブルの最大数を指定するにはこのオプションを使用します。DMS は、専用のサブタスクを使用して、対応する Kinesis ターゲットテーブルに各テーブルをロードします。デフォルトは 8、最大値は 49 です。
- ParallelLoadThreads – このオプションを使用して、 AWS DMS が各テーブルを Kinesis ターゲットテーブルにロードするために使用するスレッドの数を指定します。Kinesis Data Streams ターゲットの最大値は 32 です。この上限を増やすよう依頼できます。
- ParallelLoadBufferSize – Kinesis ターゲットにデータをロードするために並列ロードスレッドが使用する、バッファ内に保存するレコードの最大数を指定するには、このオプションを使用します。デフォルト値は 50 です。最大値は 1000 です。この設定は ParallelLoadThreads で使用します。ParallelLoadBufferSize は、複数のスレッドがある場合にのみ有効です。
- ParallelLoadQueuesPerThread - このオプションを使用して、各同時スレッドがキューからデータレコードを取り出し、ターゲットのバッチロードを生成するためにアクセスするキューの数を指定します。デフォルト は 1 です。ただし、さまざまなペイロードサイズの Kinesis ターゲットの場合、有効な範囲は 1 スレッドあたり 5 ~ 512 キューです。

マルチスレッド CDC ロードタスクの設定

タスク設定を使用して PutRecords API コールの動作を変更するなど、Kinesis などのリアルタイム データストリーミング ターゲット エンドポイントの変更データキャプチャ (CDC) のパフォーマンスを向上させることができます。これを行うには、ParallelApply* タスク設定を使用して、同

時スレッドの数、スレッドあたりのキュー数、バッファに格納するレコード数を指定します。例えば、CDC ロードを実行し、128 本のスレッドを並列に適用するとします。また、スレッドあたり 64 個のキューにアクセスして、バッファあたり 50 個のレコードを保存する必要があります。

CDC パフォーマンスを向上させるために、 は次のタスク設定 AWS DMS をサポートします。

- `ParallelApplyThreads` – CDC ロード中に AWS DMS がデータレコードを Kinesis ターゲットエンドポイントにプッシュするために使用する同時スレッドの数を指定します。デフォルト値は 0 で、最大値は 32 です。
- `ParallelApplyBufferSize` – CDC ロード中に同時スレッドが Kinesis ターゲット エンドポイントにプッシュする場合に、各バッファキューに保存するレコードの最大数を指定します。デフォルト値は 100 で、最大値は 1,000 です。このオプションは、`ParallelApplyThreads` で複数のスレッドを指定する場合に使用します。
- `ParallelApplyQueuesPerThread` – 各スレッドがキューからデータレコードを取り出し、CDC 中に Kinesis エンドポイントのバッチロードを生成するためにアクセスするキューの数を指定します。デフォルト値は 1、最大値は 512 です。

`ParallelApply*` タスク設定を使用する場合、`partition-key-type` のデフォルトは `schema-name.table-name` ではなくテーブルの `primary-key` です。

前イメージを使用した Kinesis データストリームの CDC 行の元の値のターゲットとしての表示

Kinesis のようなデータストリーミングターゲットに CDC 更新を書き込む場合、更新によって変更される前に、ソースデータベースの行の元の値を表示できます。これを可能にするために、 はソースデータベースエンジンによって提供されるデータに基づいて、更新イベントの前のイメージ AWS DMS を設定します。

ソースデータベースエンジンによって、前イメージに対してさまざまな量の情報が提供されます。

- Oracle では、列が変更された場合にのみ列の更新が提供されます。
- PostgreSQL は、プライマリキーの一部である列のデータ (変更されたかどうか) のみを提供します。すべての列に (変更の有無を問わず) データを提供するには、`REPLICA_IDENTITY` を `DEFAULT`でなく `FULL` に設定する必要があります。各テーブルを `REPLICA_IDENTITY` に設定する際は、慎重に行うべきであることに注意します。`REPLICA_IDENTITY` を `FULL` に設定すると、すべての列値がログ先行書き込み (WAL) に継続的に書き込まれます。これにより、頻繁に更新されるテーブルではパフォーマンスやリソースの問題が発生する可能性があります。

- MySQL は通常、BLOB および CLOB データ型を除くすべての列のデータを (変更の有無を問わず) 提供します。

前イメージを有効にして、ソースデータベースから元の値を AWS DMS 出力に追加するには、BeforeImageSettings タスク設定または add-before-image-columns パラメータを使用します。このパラメータは、列変換ルールを適用します。

BeforeImageSettings は、次に示すように、ソースデータベースシステムから収集された値を使用して、すべての更新オペレーションに新しい JSON 属性を追加します。

```
"BeforeImageSettings": {
    "EnableBeforeImage": boolean,
    "FieldName": string,
    "ColumnFilter": pk-only (default) / non-lob / all (but only one)
}
```

> **Note**
>
> フルロードと CDC BeforeImageSettings AWS DMS タスク (既存のデータを移行して進行中の変更をレプリケートする) などの CDC コンポーネントを含むタスク、または CDC のみのタスク (データ変更のみをレプリケートする) にのみ適用されます。全ロードのタスクには BeforeImageSettings を適用しないでください。

BeforeImageSettings オプションには、次の項目が適用されます。

- EnableBeforeImage オプションを true に設定して、前イメージを有効にします。デフォルトは false です。
- FieldName オプションを使用して、新しい JSON 属性に名前を割り当てます。EnableBeforeImage が true の場合、FieldName は必須であり、空にすることはできません。
- ColumnFilter オプションは、前イメージを使用して追加する列を指定します。テーブルのプライマリキーの一部である列だけを追加するには、デフォルト値 pk-only を使用します。前イメージ値を持つ列を追加するには、all を使用します。変更前イメージには、CLOB や BLOB などの LOB データ型の列が含まれていないことに注意します。

```
"BeforeImageSettings": {
    "EnableBeforeImage": true,
    "FieldName": "before-image",
    "ColumnFilter": "pk-only"
  }
```

> **Note**
>
> Amazon S3 ターゲットは BeforeImageSettings をサポートしていません。S3 ターゲットの場合、CDC 中に前イメージを実行するには add-before-image-columns 変換ルールのみを使用します。

前イメージ変換ルールの使用

タスク設定の代わりに、列変換ルールを適用する add-before-image-columns パラメータを使用できます。このパラメータを使用すると、Kinesis のようなデータストリーミングターゲットで CDC 中に前イメージを有効にできます。

変換ルールで add-before-image-columns を使用すると、前イメージの結果のよりきめ細かい制御を適用することができます。変換ルールを使用すると、オブジェクトロケーターを使用し、ルールに選択したテーブルを制御できます。また、変換ルールを連結することもできます。これにより、テーブルごとに異なるルールを適用できます。その後、他のルールを使用して生成された列を操作できます。

> **Note**
>
> 同じタスク内で、add-before-image-columns パラメータと同時に BeforeImageSettings タスク設定を使用しないでください。代わりに、1 つのタスクにこのパラメータとこの設定のいずれかを使用し、両方を使用しないでください。

列の add-before-image-columns パラメータを持つ transformation ルールタイプは、before-image-def セクションを提供する必要があります。例を以下に示します。

```
    {
      "rule-type": "transformation",
```

```
      …
      "rule-target": "column",
      "rule-action": "add-before-image-columns",
      "before-image-def":{
        "column-filter": one-of  (pk-only / non-lob / all),
        "column-prefix": string,
        "column-suffix": string,
      }
    }
```

column-prefix の値は列名の前に付加され、column-prefix のデフォルト値は BI_ です。column-suffix の値は列名に追加され、デフォルトは空です。column-prefix と column-suffix の両方を空の文字列に設定しないでください。

column-filter の値を 1 つ選択します。テーブルのプライマリキーの一部である列だけを追加するには、pk-only を選択します。LOB タイプではない列のみを追加するように non-lob を選択します。または、前イメージの値を持つ任意の列を追加するように all を選択します。

前イメージ変換前ルールの例

次の例の変換ルールは、ターゲットに BI_emp_no という新しい列を追加します。したがって、UPDATE employees SET emp_no = 3 WHERE emp_no = 1; のようなステートメントは、BI_emp_no フィールドに 1 を設定します。CDC 更新を Amazon S3 ターゲットに書き込むと、更新された元の行は BI_emp_no 列からわかります。

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "%",
        "table-name": "%"
      },
      "rule-action": "include"
    },
    {
      "rule-type": "transformation",
      "rule-id": "2",
      "rule-name": "2",
      "rule-target": "column",
```

```
      "object-locator": {
        "schema-name": "%",
        "table-name": "employees"
      },
      "rule-action": "add-before-image-columns",
      "before-image-def": {
        "column-prefix": "BI_",
        "column-suffix": "",
        "column-filter": "pk-only"
      }
    }
  ]
}
```

add-before-image-columns ルールアクションの使用方法については、「変換ルールおよび変換アクション」をご参照ください。

のターゲットとして Kinesis データストリームを使用するための前提条件 AWS Database Migration Service

のターゲットとして Kinesis データストリームを使用するための IAM ロール AWS Database Migration Service

のターゲットとして Kinesis データストリームを設定する前に AWS DMS、必ず IAM ロールを作成してください。このロールは AWS DMS 、 が移行先の Kinesis データストリームを引き受けてアクセスを許可することを許可する必要があります。アクセス許可の最小設定は、次の IAM ポリシーで示します。

```
{
   "Version": "2012-10-17",
   "Statement": [
   {
     "Sid": "1",
     "Effect": "Allow",
     "Principal": {
        "Service": "dms.amazonaws.com"
     },
   "Action": "sts:AssumeRole"
   }
]
}
```

Kinesis データストリームに移行する際に使用するロールには、次のアクセス許可が必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": "arn:aws:kinesis:region:accountID:stream/streamName"
        }
    ]
}
```

のターゲットとしての Kinesis データストリームへのアクセス AWS Database Migration Service

AWS DMS バージョン 3.4.7 以降では、Kinesis エンドポイントに接続するには、次のいずれかを実行する必要があります。

- VPC エンドポイントを使用するように DMS を設定します。VPC エンドポイントを使用するように DMS を設定する方法については、「AWS DMS ソースエンドポイントとターゲットエンドポイントとしての VPC エンドポイントの設定」を参照してください。
- パブリックルートを使用するように DMS を設定します。つまり、レプリケーションインスタンスをパブリックにします。パブリックレプリケーションインスタンスの詳細については、「パブリックおよびプライベートレプリケーション インスタンス」を参照してください。

のターゲットとして Kinesis Data Streams を使用する場合の制限 AWS Database Migration Service

ターゲットとして Kinesis Data Streams を使用する場合、以下の制限が適用されます。

- AWS DMS は、トランザクションに関係なく、特定の Kinesis データストリーム内の 1 つのデータレコードとして、ソースデータベース内の 1 つのレコードに各更新を発行します。ただ

し、KinesisSettings API の関連パラメータを使用して、各データレコードのトランザクションの詳細を含めることができます。

- 完全 LOB モードはサポートされていません。
- サポートされる最大 LOB サイズは 1 MB です。
- Kinesis Data Streamsは、重複排除をサポートしていません。ストリームからデータを消費するアプリケーションは、重複したレコードを処理する必要があります。詳細については、Amazon Kinesis Data Streams デベロッパーガイドの「重複レコードの処理」をご参照ください。
- AWS DMS は、パーティションキーに対して次の 2 つの形式をサポートしています。
  - SchemaName.TableName: スキーマとテーブル名の組み合わせ。
  - ${AttributeName}: JSON のいずれかのフィールドの値、またはソースデータベースのテーブルのプライマリキー。
- Kinesis Data Streams 内の保管中のデータの暗号化の詳細については、AWS Key Management Service デベロッパーガイドの「Kinesis Data Streams でのデータ保護」をご参照ください。
- BatchApplyは Kinesis エンドポイントではサポートされていません。Kinesis ターゲットにバッチ 適用を使用 (例えば、BatchApplyEnabled ターゲットメタデータ タスク設定) すると、データが失われる可能性があります。
- Kinesis ターゲットは、レプリケーションインスタンスと同じ AWS リージョン AWS アカウントおよび同じ の Kinesis データストリームでのみサポートされます。
- MySQL ソースから移行する場合、 BeforeImage データには CLOB および BLOB データ型は含まれません。詳細については、「前イメージを使用した Kinesis データストリームの CDC 行の元の値のターゲットとしての表示」を参照してください。
- AWS DMS は、16 桁を超えるBigIntデータ型の値の移行をサポートしていません。この制限を回避するには、次の変換ルールを使用して BigInt 列を文字列に変換できます。変換ルールの詳細については、「 変換ルールおよび変換アクション」を参照してください。

```
{
    "rule-type": "transformation",
    "rule-id": "id",
    "rule-name": "name",
    "rule-target": "column",
    "object-locator": {
        "schema-name": "valid object-mapping rule action",
        "table-name": "",
        "column-name": ""
    },
    "rule-action": "change-data-type",
```

```
    "data-type": {
        "type": "string",
        "length": 20
    }
}
```

Kinesis データストリームにデータを移行するためのオブジェクトマッピングの使用

AWS DMS は、テーブルマッピングルールを使用して、ソースからターゲット Kinesis データストリームにデータをマッピングします。ターゲットストリームにデータをマッピングするために、オブジェクトマッピングと呼ばれるテーブルマッピングルールのタイプを使用します。オブジェクトマッピングを使用して、ソースのデータレコードがどのように Kinesis データストリームに発行されたデータレコードにマッピングされるかを定義します。

Kinesis データストリームには、パーティション　キー以外にプリセット構造はありません。オブジェクトマッピングルールでは、データレコードの `partition-key-type` に指定できる値は、`schema-table`、`transaction-id`、`primary-key`、`constant`、`attribute-name` です。

オブジェクトマッピングルールを作成するには、`object-mapping` として `rule-type` を指定します。このルールが、使用したいオブジェクトマッピングのタイプを指定します。

ルールの構造は次のとおりです。

```
{
    "rules": [
        {
            "rule-type": "object-mapping",
            "rule-id": "id",
            "rule-name": "name",
            "rule-action": "valid object-mapping rule action",
            "object-locator": {
                "schema-name": "case-sensitive schema name",
                "table-name": ""
            }
        }
    ]
}
```

AWS DMS は現在、 `rule-action`パラメータの唯一の有効な値`map-record-to-document`として `map-record-to-record`と をサポートしています。これらの設定は、`exclude-columns`

属性リストの一部として除外されない値に影響します。map-record-to-record および map-record-to-documentの値は、 がデフォルトでこれらのレコード AWS DMS を処理する方法を指定します。これらの値は、どのような方法でも属性マッピングに影響しません。

リレーショナルデータベースから Kinesis データストリームに移行する際に map-record-to-record を使用します。このルールタイプでは、Kinesis データストリームのパーティション キーとしてリレーショナルデータベースから taskResourceId.schemaName.tableName 値を使用し、ソースデータベース内の各列の属性を作成します。

map-record-to-record を使用する場合は、次の点に注意します。

- この設定は、exclude-columns リストで除外されている列にのみ影響します。
- このような列ごとに、 はターゲットトピックに対応する属性 AWS DMS を作成します。
- AWS DMS は、ソース列が属性マッピングで使用されているかどうかにかかわらず、この対応する属性を作成します。

map-record-to-document を使用して、属性名「_doc」を使用しソース列を適切なターゲットストリーム内の単一のフラットドキュメントに配置します。 AWS DMS 説「_doc」という名前のソースで 1 つのフラットマップにデータを配置します。この配置は、exclude-columns 属性リストに含まれていないソーステーブル内のすべての列に適用されます。

map-record-to-record を理解するための 1 つの方法は、実際の動作を確認することです。この例では、次の構造とデータを含むリレーショナルデータベースのテーブルの行から始めると想定してください。

FirstName	LastName	StoreId	HomeAddress	HomePhone	WorkAddress	WorkPhone	DateofBirth
Randy	Marsh	5	221B Baker Street	1234567890	31 Spooner Street, Quahog	9876543210	02/29/1988

この情報を Test という名前のスキーマから Kinesis データストリームに移行するには、データをターゲットストリームにマッピングするルールを作成します。以下のルールはマッピングを示しています。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "rule-action": "include",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            }
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",
            "rule-name": "DefaultMapToKinesis",
            "rule-action": "map-record-to-record",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Customers"
            }
        }
    ]
}
```

次は、Kinesis データストリームの結果のレコード形式を示しています：

- StreamName: TAK
- PartitionKey: Test.Customers //schmaName.tableName
- データ: //次の JSON メッセージ

```
{
   "FirstName": "Randy",
   "LastName": "Marsh",
   "StoreId":  "5",
   "HomeAddress": "221B Baker Street",
   "HomePhone": "1234567890",
   "WorkAddress": "31 Spooner Street, Quahog",
   "WorkPhone": "9876543210",
   "DateOfBirth": "02/29/1988"
}
```

ただし、同じルールを使用しますが、rule-action パラメータを map-record-to-document に変更して特定の列を除外します。以下のルールはマッピングを示しています。

```
{
 "rules": [
    {
   "rule-type": "selection",
   "rule-id": "1",
   "rule-name": "1",
   "rule-action": "include",
   "object-locator": {
    "schema-name": "Test",
    "table-name": "%"
   }
  },
  {
   "rule-type": "object-mapping",
   "rule-id": "2",
   "rule-name": "DefaultMapToKinesis",
   "rule-action": "map-record-to-document",
   "object-locator": {
    "schema-name": "Test",
    "table-name": "Customers"
   },
   "mapping-parameters": {
    "exclude-columns": [
     "homeaddress",
     "homephone",
     "workaddress",
     "workphone"
    ]
   }
  }
 ]
}
```

この場合、exclude-columns パラメータ、FirstName、LastName、StoreId、DateOfBirth は _doc にマッピングされます。次は、この結果のレコード形式を示しています。

```
      {
           "data":{
               "_doc":{
                   "FirstName": "Randy",
                   "LastName": "Marsh",
                   "StoreId":  "5",
                   "DateOfBirth": "02/29/1988"
               }
           }
       }
```

属性マッピングを使用したデータの再構築

属性マップを使用してデータを Kinesis データストリームに移行している間にデータを再構築できます。例えば、ソース内の複数のフィールドを結合してターゲット内に 1 つのフィールドを構成することもできます。以下の属性マップはデータを再構築する方法を示しています。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "rule-action": "include",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            }
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",
            "rule-name": "TransformToKinesis",
            "rule-action": "map-record-to-record",
            "target-table-name": "CustomerData",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Customers"
            },
            "mapping-parameters": {
                "partition-key-type": "attribute-name",
```

```
                "partition-key-name": "CustomerName",
                "exclude-columns": [
                    "firstname",
                    "lastname",
                    "homeaddress",
                    "homephone",
                    "workaddress",
                    "workphone"
                ],
                "attribute-mappings": [
                    {
                        "target-attribute-name": "CustomerName",
                        "attribute-type": "scalar",
                        "attribute-sub-type": "string",
                        "value": "${lastname}, ${firstname}"
                    },
                    {
                        "target-attribute-name": "ContactDetails",
                        "attribute-type": "document",
                        "attribute-sub-type": "json",
                        "value": {
                            "Home": {
                                "Address": "${homeaddress}",
                                "Phone": "${homephone}"
                            },
                            "Work": {
                                "Address": "${workaddress}",
                                "Phone": "${workphone}"
                            }
                        }
                    }
                ]
            }
        }
    ]
}
```

partition-key の定数値を設定するには、partition-key 値を指定します。たとえば、すべてのデータを 1 つのシャードに強制的に格納するためにこれを行うことができます。以下のマッピングはこの方法を示しています。

```
{
    "rules": [
```

```
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "1",
            "rule-name": "TransformToKinesis",
            "rule-action": "map-record-to-document",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Customer"
            },
            "mapping-parameters": {
                "partition-key": {
                    "value": "ConstantPartitionKey"
                },
                "exclude-columns": [
                    "FirstName",
                    "LastName",
                    "HomeAddress",
                    "HomePhone",
                    "WorkAddress",
                    "WorkPhone"
                ],
                "attribute-mappings": [
                    {
                        "attribute-name": "CustomerName",
                        "value": "${FirstName},${LastName}"
                    },
                    {
                        "attribute-name": "ContactDetails",
                        "value": {
                            "Home": {
                                "Address": "${HomeAddress}",
                                "Phone": "${HomePhone}"
                            },
                            "Work": {
```

```
                            "Address": "${WorkAddress}",
                            "Phone": "${WorkPhone}"
                        }
                    }
                },
                {
                    "attribute-name": "DateOfBirth",
                    "value": "${DateOfBirth}"
                }
            ]
        }
    }
]
}
```

> **Note**
>
> 特定のテーブル用のコントロールレコードの partition-key 値は、TaskId.SchemaName.TableName です。特定のタスク用のコントロールレコードの partition-key 値は、そのレコードの TaskId です。オブジェクトマッピングの partition-key 値を指定することは、コントロールレコードの partition-key には影響しません。

Kinesis Data Streams のメッセージ形式

JSON 出力は、単にキーと値のペアのリストです。JSON_UNFORMATTED メッセージ形式は、改行区切り文字を含む単一行の JSON 文字列です。

AWS DMS には、Kinesis Data Streams からのデータを簡単に使用できるように、次の予約フィールドが用意されています。

RecordType

レコードタイプはデータまたはコントロールのいずれかです。データレコードは、ソースの実際の行を表します。コントロールレコードは、タスクの再起動など、ストリーム内の重要なイベント用です。

操作

データレコードの場合、オペレーションは load、insert、update、または delete です。

コントロールレコードの場合、オペレーションは `create-table`、`rename-table`、`drop-table`、`change-columns`、`add-column`、`drop-column`、`rename-column`、`column-type-change` です。

SchemaName

レコードのソーススキーマ。コントロールレコードの場合、このフィールドは空です。

TableName

レコードのソーステーブル。コントロールレコードの場合、このフィールドは空です。

タイムスタンプ

JSON メッセージが構築された時刻のタイムスタンプ。このフィールドは ISO 8601 形式でフォーマットされます。

のターゲットとしての Apache Kafka の使用 AWS Database Migration Service

を使用して AWS DMS 、Apache Kafka クラスターにデータを移行できます。Apache Kafka は分散ストリーミング プラットフォームです。Apache Kafka を使用すると、ストリーミング　データをリアルタイムで取り込み、処理できます。

AWS は、 AWS DMS ターゲットとして使用する Amazon Managed Streaming for Apache Kafka (Amazon MSK) も提供します。Amazon MSK は、Apache Kafka インスタンスの実装と管理を簡素化する、フルマネージド型 Apache Kafka ストリーミング サービスです。オープンソースの Apache Kafka バージョンで動作し、他の Apache Kafka インスタンスとまったく同じ AWS DMS ようにターゲットとして Amazon MSK インスタンスにアクセスします。詳細については、Amazon Managed Streaming for Apache Kafka デベロッパーガイドの「Amazon MSK」をご参照ください。

Kafka クラスターは、パーティションに分割されたトピックと呼ばれるカテゴリにレコードのストリームを保存します。パーティションは、トピック内のデータレコード (メッセージ) の一意に識別されたシーケンスです。パーティションは、トピックレコードの並列処理を可能にするために、クラスター内の複数のブローカーに分散することができます。トピックとパーティション、および Apache Kafka での分散の詳細については、「トピックとログ」と「分散」をご参照ください。

Kafka クラスターは、Amazon MSK インスタンス、Amazon EC2 インスタンス上で実行されるクラスター、またはオンプレミスのクラスターのいずれかです。Amazon MSK インスタンスまたは Amazon EC2 インスタンス上のクラスターは、同じ VPC 内にも、別の VPC 内にも配置できます。クラスターがオンプレミスの場合は、レプリケーションインスタンスに独自のオンプレミスのネーム

サーバーを使用して、クラスターのホスト名を解決できます。レプリケーションインスタンスのネームサーバーのセットアップについては、「独自のオンプレミスネームサーバーの使用」を参照してください。ネットワークの設定の詳細については、「レプリケーション インスタンスのためのネットワークのセットアップ」を参照してください。

Amazon MSK クラスターを使用する場合、セキュリティグループがレプリケーションインスタンスからのアクセスを許可していることを確認します。Amazon MSK クラスターのセキュリティグループの変更については、「Amazon MSK クラスターのセキュリティグループの変更」を参照してください。

AWS Database Migration Service は、JSON を使用して Kafka トピックにレコードを発行します。変換時、 AWS DMS はソースデータベースからの各レコードを JSON フォーマットの属性と値のペアにシリアル化します。

サポートされている任意のデータソースから、ターゲット Kafka クラスターにデータを移行するために、オブジェクトのマッピングを使用します。オブジェクトマッピングを使用して、ターゲットトピックにデータ レコードを構築する方法を決定します。データをそのパーティションにグループ化するために Apache Kafka で使用する、各テーブルのパーティションキーも定義します。

現在、 はタスクごとに 1 つのトピック AWS DMS をサポートしています。単一のタスクに複数のテーブルがある場合、すべてのメッセージが単一のトピックに送信されます。各メッセージには、ターゲットスキーマと table. AWS DMS versions 3.4.6 以降を識別するメタデータセクションが含まれており、オブジェクトマッピングを使用したマルチトピックレプリケーションをサポートしています。詳細については、「オブジェクトマッピングを使用したマルチトピックレプリケーション」を参照してください。

Apache Kafka のエンドポイント設定

接続の詳細は、 AWS DMS コンソールのエンドポイント設定、または CLI の `--kafka-settings`オプションを使用して指定できます。各設定の要件は次のとおりです。

- `Broker` — Kafka クラスター内の 1 つ以上のブローカーの場所を、*`broker-hostname`*`:`*`port`* のカンマ区切りリストの形式で指定します。例: `"ec2-12-345-678-901.compute-1.amazonaws.com:2345,ec2-10-987-654-321.compute-1.a`
  この設定では、クラスター内の任意またはすべてのブローカーのロケーションを指定できます。クラスターブローカーはすべて、トピックに移行されたデータレコードのパーティション化を処理するために通信します。
- `Topic` - (オプション) 最大 255 文字および記号のトピック名を指定します。ピリオド (.)、アンダースコア (_)、マイナス (-) を使用できます。ピリオド (.) またはアンダースコア (_) があるト

ピック名は、内部データ構造内で衝突する可能性があります。トピック名には、どちらか一方を使用し、両方とも使用することは避けてください。トピック名を指定しない場合、 は移行トピック"kafka-default-topic"として AWS DMS を使用します。

> **Note**
>
> 指定した移行トピックまたはデフォルトトピックのいずれかを で AWS DMS 作成するには、Kafka クラスター設定auto.create.topics.enable = trueの一部として を設定します。詳細については、「のターゲットとして Apache Kafka を使用する場合の制限事項 AWS Database Migration Service」をご参照ください。

- MessageFormat - エンドポイントで作成されたレコードの出力形式。メッセージ形式は JSON (デフォルト) または JSON_UNFORMATTED (タブなし 1 行) です。
- MessageMaxBytes — エンドポイントで作成されたレコードの最大サイズ (バイト単位)。デフォルトは 1,000,000 です。

> **Note**
>
> AWS CLI/SDK を使用してのみ、デフォルト以外の値MessageMaxBytesに変更できます。例えば、既存の Kafka エンドポイントを変更して MessageMaxBytes を変更するには、以下のコマンドを使用します。
>
> ```
> aws dms modify-endpoint --endpoint-arn your-endpoint
> --kafka-settings Broker="broker1-server:broker1-port,broker2-server:broker2-
> port,...",
> Topic=topic-name,MessageMaxBytes=integer-of-max-message-size-in-bytes
> ```

- IncludeTransactionDetails - ソースデータベースからの詳細のトランザクション情報を提供します。この情報には、コミットタイムスタンプ、ログの位置、transaction_id、previous_transaction_id、および transaction_record_id (トランザクション内のレコードオフセット) の値が含まれます。デフォルトは false です。
- IncludePartitionValue パーティションタイプが でない限り、Kafka メッセージ出力内のパーティション値を表示します。schema-table-typeデフォルトは false です。
- PartitionIncludeSchemaTable パーティションタイプが primary-key-type の場合、スキーマとテーブル名をパーティション値にプレフィックスします。これにより、Kafka パーティション間のデータ分散が増加します。例えば、SysBench スキーマに数千のテーブルがあり、各テーブルのプライマリ キーの範囲が制限されているとします。この場合、同じプライマリキーが

数千のテーブルから同じパーティションに送信され、スロットリングが発生します。デフォルトは false です。

- IncludeTableAlterOperations – rename-table、drop-table、add-column、drop-column、rename-column など、制御データのテーブルを変更するデータ定義言語 (DDL) オペレーションが含まれます。デフォルトは false です。
- IncludeControlDetails - Kafka メッセージ出力に、テーブル定義、列定義、テーブルおよび列の変更の詳細な制御情報を表示します。デフォルトは false です。
- IncludeNullAndEmpty — ターゲットに NULL 列と空の列を含めます。デフォルトは false です。
- SecurityProtocol — Transport Layer Security (TLS) を使用して Kafka ターゲット エンドポイントへの安全な接続を設定します。オプションには ssl-authentication、ssl-encryption、および sasl-ssl があります。sasl-ssl を使用して SaslUsername と SaslPassword を要求します。
- SslEndpointIdentificationAlgorithm – 証明書のホスト名検証を設定します。この設定は、 バージョン 3.5.1 以降で AWS DMS サポートされています。オプションは以下のとおりです。
  - NONE: クライアント接続のブローカーのホスト名検証を無効にします。
  - HTTPS: クライアント接続でブローカーのホスト名検証を有効にします。

設定を使用すると、転送速度を上げることができます。これを行うために、 AWS DMS は Apache Kafka ターゲット　クラスターへのマルチスレッド全ロードをサポートしています。 AWS DMS は、次のようなタスク設定を使用して、このマルチスレッドをサポートします：

- MaxFullLoadSubTasks – このオプションを使用して、並列ロードするソーステーブルの最大数を指定します。 は、専用のサブタスクを使用して、各テーブルを対応する Kafka ターゲットテーブルに AWS DMS ロードします。デフォルトは 8、最大値は 49 です。
- ParallelLoadThreads – このオプションを使用して、 AWS DMS が Kafka ターゲットテーブルに各テーブルをロードするために使用するスレッドの数を指定します。Apache Kafka ターゲットの最大値は 32 です。この上限を増やすよう依頼できます。
- ParallelLoadBufferSize - Kafka ターゲットにデータをロードするために並列ロードスレッドが使用する、バッファ内に保存するレコードの最大数を指定するには、このオプションを使用します。デフォルト値は 50 です。最大値は 1000 です。この設定は ParallelLoadThreads で使用します。ParallelLoadBufferSize は、複数のスレッドがある場合にのみ有効です。

- ParallelLoadQueuesPerThread - このオプションを使用して、各同時スレッドがキューからデータレコードを取り出し、ターゲットのバッチロードを生成するためにアクセスするキューの数を指定します。デフォルト は 1 です。最大数は 512。

Kafka エンドポイントの変更データキャプチャ (CDC) のパフォーマンスを向上するには、並列スレッドと一括オペレーションのタスク設定を調整します。これを行うには、ParallelApply* タスク設定を使用して、同時スレッドの数、スレッドあたりのキュー数、バッファに格納するレコード数を指定します。例えば、CDC ロードを実行し、128 本のスレッドを並列に適用するとします。また、スレッドあたり 64 個のキューにアクセスして、バッファあたり 50 個のレコードを保存する必要があります。

CDC パフォーマンスを向上させるために、 は次のタスク設定 AWS DMS をサポートします。

- ParallelApplyThreads – CDC ロード中に AWS DMS がデータレコードを Kafka ターゲットエンドポイントにプッシュするために使用する同時スレッドの数を指定します。デフォルト値は 0 で、最大値は 32 です。
- ParallelApplyBufferSize - CDC ロード中に同時スレッドが Kafka ターゲット エンドポイントにプッシュする場合に、各バッファキューに保存するレコードの最大数を指定します。デフォルト値は 100 で、最大値は 1,000 です。このオプションは、ParallelApplyThreads で複数のスレッドを指定する場合に使用します。
- ParallelApplyQueuesPerThread - 各スレッドがキューからデータレコードを取り出し、CDC 中に Kafka エンドポイントのバッチロードを生成するためにアクセスするキューの数を指定します。デフォルト は 1 です。最大数は 512。

ParallelApply* タスク設定を使用する場合、partition-key-type のデフォルトは schema-name.table-name ではなくテーブルの primary-key です。

Transport Layer Security (TLS) を使用した Kafka への接続

このクラスターは Transport Layer Security (TLS) を使用した安全な接続のみを受け入れます。DMS では、次の 3 つのセキュリティプロトコルオプションのいずれかを使用して、Kafka エンドポイント接続をセキュリティで保護できます。

SSL 暗号化 (**server-encryption**)

クライアントは、サーバーの証明書を使用してサーバー ID を検証します。次に、サーバーとクライアント間で暗号化された接続が確立されます。

SSL 認証 (**mutual-authentication**)

サーバーとクライアントは、独自の証明書を使用して ID を相互に検証します。次に、サーバーとクライアント間で暗号化された接続が確立されます。

SASL-SSL (**mutual-authentication**)

簡易認証およびセキュリティ レイヤー (SASL) メソッドは、クライアントの証明書をユーザー名とパスワードに置き換えて、クライアント ID を検証します。具体的には、サーバーがクライアントの ID を検証できるように、サーバーが登録したユーザー名とパスワードを指定します。次に、サーバーとクライアント間で暗号化された接続が確立されます。

> **Important**
>
> Apache Kafka と Amazon MSK は解決済みの証明書を受け入れます。これは、Kafka と Amazon MSK が対処すべき既知の制限です。詳細については、「Apache Kafka の問題、KAFKA-3700」をご参照ください。
> Amazon MSK を使用している場合は、この既知の制限の回避策としてアクセスコントロールリスト (ACL) を使用することを検討してください。ACL の使用の詳細については、Amazon Managed Streaming for Apache Kafka デベロッパーガイドの Apache Kafka ACL セクションをご参照ください。
> 自己管理 Kafka クラスターを使用している場合クラスターの設定の詳細については、「2018/10/21日付のコメント」をご参照ください。

Amazon MSK または自己管理 Kafka クラスターでの SSL 暗号化の使用

SSL 暗号化を使用して、Amazon MSK または自己管理 Kafka クラスターへのエンドポイント接続をセキュリティで保護できます。SSL 暗号化認証方法を使用する場合、クライアントはサーバーの証明書を使用してサーバーの ID を検証します。次に、サーバーとクライアント間で暗号化された接続が確立されます。

SSL 暗号化を使用して Amazon MSK に接続するには

- ターゲットの Kafka エンドポイントを作成するとき、`ssl-encryption` オプションを使うセキュリティ プロトコル エンドポイントの設定 (`SecurityProtocol`) を行います。

  次の JSON 例では、セキュリティプロトコルを SSL 暗号化として設定しています。

```
"KafkaSettings": {
    "SecurityProtocol": "ssl-encryption",
}
```

自己管理 Kafka クラスターに SSL 暗号化を使用するには

1. オンプレミス Kafka クラスターでPrivate Certificate Authority (CA) を使用している場合は、プライベート CA 証明書をアップロードして Amazon リソースネーム (ARN) を取得します。
2. ターゲットの Kafka エンドポイントを作成するとき、`ssl-encryption` オプションを使うセキュリティ プロトコル エンドポイントの設定 (`SecurityProtocol`) を行います。次の JSON の例では、セキュリティプロトコルを `ssl-encryption` として設定します。

   ```
   "KafkaSettings": {
       "SecurityProtocol": "ssl-encryption",
   }
   ```

3. プライベート CA を使用している場合は、上記の最初のステップで取得した ARN に `SslCaCertificateArn` を設定します。

SSL 認証の使用

SSL 認証を使用して、Amazon MSK または自己管理 Kafka クラスターへのエンドポイント接続をセキュリティで保護できます。

SSL 認証を使用したクライアント認証と暗号化を有効にして Amazon MSK に接続するには、次の手順を実行します：

- Kafka の秘密キーと公開証明書を準備します。
- 証明書を DMS Certificate Manager にアップロードします。
- Kafka エンドポイント設定で指定された、対応する証明書 ARN を使用して Kafka ターゲットエンドポイントを作成します。

Amazon MSK の秘密キーと公開証明書を準備するには

1. EC2 インスタンスを作成し、Amazon Managed Streaming for Apache Kafka 開発者ガイドのクライアント認証セクションにあるステップ 1 ~ 9 の説明に従って、認証を使用するようにクライアントをセットアップします。

   これらの手順を完了したら、Certificate-ARN (ACM に保存された公開証明書 ARN) と、kafka.client.keystore.jks ファイル内のプライベートキーができます。

2. 公開証明書を取得し、次のコマンドを使用し証明書を signed-certificate-from-acm.pem ファイルにコピーします：

   ```
   aws acm-pca get-certificate --certificate-authority-arn Private_CA_ARN --
   certificate-arn Certificate_ARN
   ```

   コマンドは以下のような情報を返します：

   ```
   {"Certificate": "123", "CertificateChain": "456"}
   ```

   次に、"123" に同等のものを signed-certificate-from-acm.pem ファイルにコピーします。

3. 次の例に示すように,msk-rsa キーを kafka.client.keystore.jks to keystore.p12 からインポートしてプライベートキーを取得します:

   ```
   keytool -importkeystore \
   -srckeystore kafka.client.keystore.jks \
   -destkeystore keystore.p12 \
   -deststoretype PKCS12 \
   -srcalias msk-rsa-client \
   -deststorepass test1234 \
   -destkeypass test1234
   ```

4. 次のコマンドを使用してkeystore.p12 を .pem の形式にエクスポートします。

   ```
   Openssl pkcs12 -in keystore.p12 -out encrypted-private-client-key.pem –nocerts
   ```

   [Enter PEM pass phrase](PEM パスフレーズを入力してください) メッセージが表示され、証明書の暗号化に適用されるキーを識別します。

5. バッグ属性とキー属性を .pem ファイルから削除し、最初の行が次の文字列でスタートしていることを確認します。

```
                    ---BEGIN ENCRYPTED PRIVATE KEY---
```

公開証明書とプライベートキーを DMS Certificate Managerにアップロードし、Amazon MSK への接続をテストするには

1. 次のコマンドを使用して、DMS Certificate Managerにアップロードします。

```
aws dms import-certificate --certificate-identifier signed-cert --certificate-pem
 file://path to signed cert
aws dms import-certificate --certificate-identifier private-key –certificate-pem
 file://path to private key
```

2. Amazon MSK ターゲット エンドポイントを作成し、接続をテストして TLS 認証が機能することを確認します。

```
aws dms create-endpoint --endpoint-identifier $endpoint-identifier --engine-name
 kafka --endpoint-type target --kafka-settings
'{"Broker": "b-0.kafka260.aaaaa1.a99.kafka.us-east-1.amazonaws.com:0000",
 "SecurityProtocol":"ssl-authentication",
"SslClientCertificateArn": "arn:aws:dms:us-east-1:012346789012:cert:",
"SslClientKeyArn": "arn:aws:dms:us-
east-1:0123456789012:cert:","SslClientKeyPassword":"test1234"}'
aws dms test-connection -replication-instance-arn=$rep_inst_arn –endpoint-arn=
$kafka_tar_arn_msk
```

Important

SSL 認証を使用して、自己管理 Kafka クラスターへの接続をセキュリティで保護できます。場合によっては、オンプレミス Kafka クラスターでPrivate Certificate Authority (CA) を使用することがあります。その場合は、CA チェーンおよび公開証明書、プライベートキーを DMS Certificate Managerにアップロードします。次に、オンプレミス Kafka ターゲットエンドポイントを作成するときに、エンドポイント設定で対応する Amazon リソースネーム (ARN) を使用します。

自己管理 Kafka クラスターのプライベートキーと署名付き証明書を準備するには

1. 以下に示すようにキーペアを生成します。

```
keytool -genkey -keystore kafka.server.keystore.jks -validity 300 -storepass your-
keystore-password
-keypass your-key-passphrase -dname "CN=your-cn-name"
-alias alias-of-key-pair -storetype pkcs12 -keyalg RSA
```

2. 証明書署名リクエスト (CSR) を取得します。

```
keytool -keystore kafka.server.keystore.jks -certreq -file server-cert-sign-
request-rsa -alias on-premise-rsa -storepass your-key-store-password
-keypass your-key-password
```

3. クラスター トラストストアの CA を使用して CSR に署名します。CA がない場合は、独自のプライベート CA を作成できます。

```
openssl req -new -x509 -keyout ca-key -out ca-cert -days validate-days
```

4. ca-cert をサーバーのトラストストアとキーストアにインポートします。トラストストアをお持ちでない場合は、次のコマンドを使用してトラストストアを作成してこれに ca-cert をインポートします。

```
keytool -keystore kafka.server.truststore.jks -alias CARoot -import -file ca-cert
keytool -keystore kafka.server.keystore.jks -alias CARoot -import -file ca-cert
```

5. 証明書に署名します。

```
openssl x509 -req -CA ca-cert -CAkey ca-key -in server-cert-sign-request-rsa -out
 signed-server-certificate.pem
-days validate-days -CAcreateserial -passin pass:ca-password
```

6. 署名付き証明書をキーストアにインポートします。

```
keytool -keystore kafka.server.keystore.jks -import -file signed-certificate.pem -
alias on-premise-rsa -storepass your-keystore-password
-keypass your-key-password
```

7. 次のコマンドを使用して、on-premise-rsa キーを kafka.server.keystore.jks から keystore.p12 にインポートします。

```
keytool -importkeystore \
-srckeystore kafka.server.keystore.jks \
-destkeystore keystore.p12 \
-deststoretype PKCS12 \
-srcalias on-premise-rsa \
-deststorepass your-truststore-password \
-destkeypass your-key-password
```

8. 次のコマンドを使用して keystore.p12 を .pem の形式にエクスポートします。

```
Openssl pkcs12 -in keystore.p12 -out encrypted-private-server-key.pem –nocerts
```

9. encrypted-private-server-key.pem および signed-certificate.pem、ca-certを DMS Certificate Managerにアップロードします。
10. 返された ARN を使用してエンドポイントを作成します。

```
aws dms create-endpoint --endpoint-identifier $endpoint-identifier --engine-name
 kafka --endpoint-type target --kafka-settings
'{"Broker": "b-0.kafka260.aaaaa1.a99.kafka.us-east-1.amazonaws.com:9092",
 "SecurityProtocol":"ssl-authentication",
"SslClientCertificateArn": "your-client-cert-arn","SslClientKeyArn": "your-client-
key-arn","SslClientKeyPassword":"your-client-key-password",
"SslCaCertificateArn": "your-ca-certificate-arn"}'

aws dms test-connection -replication-instance-arn=$rep_inst_arn –endpoint-arn=
$kafka_tar_arn_msk
```

SASL-SSL 認証を使用して Amazon MSK に接続する

簡易認証およびセキュリティレイヤー (SASL) 方式では、ユーザー名とパスワードを使用してクライアント ID を検証し、サーバーとクライアント間で暗号化された接続を作成します。

SASL を使用するには、Amazon MSK クラスターを設定するときに、まず安全なユーザー名とパスワードを作成します。Amazon MSK クラスターの安全なユーザー名とパスワードを設定する方法については、Amazon Managed Streaming for Apache Kafka デベロッパーガイドの「Amazon MSK クラスターの SALS/SCRAM 認証の設定」をご参照ください。

次に、Kafka ターゲット エンドポイントを作成するときに、`sasl-ssl` オプションを使ってセキュリティプロトコル エンドポイントの設定 (`SecurityProtocol`) を行います。`SaslUsername` と `SaslPassword` オプションも設定します。次の JSON の例に示すように、Amazon MSK クラスターを初めてセットアップしたときに作成した安全なユーザー名とパスワードと一致していることを確認してください。

```
"KafkaSettings": {
    "SecurityProtocol": "sasl-ssl",
    "SaslUsername":"Amazon MSK cluster secure user name",
    "SaslPassword":"Amazon MSK cluster secure password"
}
```

> **Note**
>
> - 現在、 はパブリック CA ベースの SASL-SSL のみ AWS DMS をサポートしています。DMS は、プライベート CA を基盤とするセルフマネージド型 Kafka で使用する SASL-SSL はサポートしていません。
> - SASL-SSL 認証の場合、 は SCRAM-SHA-512 メカニズムをデフォルトで AWS DMS サポートします。 AWS DMS バージョン 3.5.0 以降は Plain メカニズムもサポートしています。Plain メカニズムを使用するには、`KafkaSettings` API データ型の `SaslMechanism` パラメータを `PLAIN` に設定します。

ターゲットとして Apache Kafka の CDC 行の元の値を表示するために前イメージを使用

Kafka のようなデータストリーミングターゲットに CDC 更新を書き込むときは、更新によって変更される前に、ソースデータベースの行の元の値を表示できます。これを可能にするために、 はソースデータベースエンジンによって提供されるデータに基づいて、更新イベントの前のイメージ AWS DMS を設定します。

ソースデータベースエンジンによって、前イメージに対してさまざまな量の情報が提供されます。

- Oracle では、列が変更された場合にのみ列の更新が提供されます。
- PostgreSQL は、プライマリキーの一部である列のデータ (変更されたかどうか) のみを提供します。論理レプリケーションを使用し、ソーステーブルに REPLICA IDENTITY FULL を設定した場合は、WAL に書き込まれた行の変更前と変更後の情報をすべて取得できます。このような情報はここで確認できます。
- MySQL は通常、すべての列のデータ (変更されたかどうか) を提供します。

前イメージを有効にして、ソースデータベースから元の値を AWS DMS 出力に追加するには、`BeforeImageSettings` タスク設定または `add-before-image-columns` パラメータを使用します。このパラメータは、列変換ルールを適用します。

`BeforeImageSettings` は、次に示すように、ソースデータベースシステムから収集された値を使用して、すべての更新オペレーションに新しい JSON 属性を追加します。

```
"BeforeImageSettings": {
    "EnableBeforeImage": boolean,
    "FieldName": string,
    "ColumnFilter": pk-only (default) / non-lob / all (but only one)
}
```

> **Note**
>
> 全ロード + CDCタスク (既存のデータを移行して進行中の変更をレプリケートする)、または CDC のみのタスク (データ変更のみをレプリケートする) に `BeforeImageSettings` を適用します。全ロードのタスクには `BeforeImageSettings` を適用しないでください。

BeforeImageSettings オプションには、次の項目が適用されます。

- EnableBeforeImage オプションを true に設定して、前イメージを有効にします。デフォルトは false です。
- FieldName オプションを使用して、新しい JSON 属性に名前を割り当てます。EnableBeforeImage が true の場合、FieldName は必須であり、空にすることはできません。
- ColumnFilter オプションは、前イメージを使用して追加する列を指定します。テーブルのプライマリキーの一部である列だけを追加するには、デフォルト値 pk-only を使用します。LOB タイプではない列のみを追加するには、non-lob を使用します。前イメージ値を持つ列を追加するには、all を使用します。

```
"BeforeImageSettings": {
    "EnableBeforeImage": true,
    "FieldName": "before-image",
    "ColumnFilter": "pk-only"
  }
```

前イメージ変換ルールの使用

タスク設定の代わりに、列変換ルールを適用する add-before-image-columns パラメータを使用できます。このパラメータを使用すると、Kafka のようなデータストリーミングターゲットで CDC 中に前イメージを有効にできます。

変換ルールで add-before-image-columns を使用すると、前イメージの結果のよりきめ細かい制御を適用することができます。変換ルールを使用すると、オブジェクトロケーターを使用し、ルールに選択したテーブルを制御できます。また、変換ルールを連結することもできます。これにより、テーブルごとに異なるルールを適用できます。その後、他のルールを使用して生成された列を操作できます。

> **Note**
>
> 同じタスク内で、add-before-image-columns パラメータと同時に BeforeImageSettings タスク設定を使用しないでください。代わりに、1 つのタスクにこのパラメータとこの設定のいずれかを使用し、両方を使用しないでください。

列の add-before-image-columns パラメータを持つ transformation ルールタイプは、before-image-def セクションを提供する必要があります。例を以下に示します。

```
    {
      "rule-type": "transformation",
      …
      "rule-target": "column",
      "rule-action": "add-before-image-columns",
      "before-image-def":{
        "column-filter": one-of  (pk-only / non-lob / all),
        "column-prefix": string,
        "column-suffix": string,
      }
    }
```

column-prefix の値は列名の前に付加され、column-prefix のデフォルト値は BI_ です。column-suffix の値は列名に追加され、デフォルトは空です。column-prefix と column-suffix の両方を空の文字列に設定しないでください。

column-filter の値を 1 つ選択します。テーブルのプライマリキーの一部である列だけを追加するには、pk-only を選択します。LOB タイプではない列のみを追加するように non-lob を選択します。または、前イメージの値を持つ任意の列を追加するように all を選択します。

前イメージ変換前ルールの例

次の例の変換ルールは、ターゲットに BI_emp_no という新しい列を追加します。したがって、UPDATE employees SET emp_no = 3 WHERE emp_no = 1; のようなステートメントは、BI_emp_no フィールドに 1 を設定します。CDC 更新を Amazon S3 ターゲットに書き込むと、更新された元の行は BI_emp_no 列からわかります。

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "%",
        "table-name": "%"
      },
      "rule-action": "include"
```

```
    },
    {
      "rule-type": "transformation",
      "rule-id": "2",
      "rule-name": "2",
      "rule-target": "column",
      "object-locator": {
        "schema-name": "%",
        "table-name": "employees"
      },
      "rule-action": "add-before-image-columns",
      "before-image-def": {
        "column-prefix": "BI_",
        "column-suffix": "",
        "column-filter": "pk-only"
      }
    }
  ]
}
```

add-before-image-columns ルールアクションの使用方法については、「変換ルールおよび変換アクション」をご参照ください。

のターゲットとして Apache Kafka を使用する場合の制限事項 AWS Database Migration Service

ターゲットとして Apache Kafka を使用する場合、以下の制限が適用されます。

- AWS DMS Kafka ターゲットエンドポイントは、Amazon Managed Streaming for Apache Kafka (Amazon MSK) の IAM アクセスコントロールをサポートしていません。
- 完全 LOB モードはサポートされていません。
- が新しいトピックを自動的に作成できるようにするプロパティを使用して AWS DMS 、クラスターの Kafka 設定ファイルを指定します。設定 `auto.create.topics.enable = true` を含めます。Amazon MSK を使用している場合は、Kafka クラスターを作成するときにデフォルト設定を指定し、`auto.create.topics.enable` 設定を `true` に変更できます。デフォルト設定の詳細については、Amazon Managed Streaming for Apache Kafka デベロッパーガイドの「Amazon MSK のデフォルト設定」をご参照ください。Amazon MSK を使用して作成された既存の Kafka クラスターを変更する必要がある場合は、次の例のように AWS CLI コマンド`aws kafka create-configuration`を実行して Kafka 設定を更新します。

```
14:38:41 $ aws kafka create-configuration --name "kafka-configuration" --kafka-
versions "2.2.1" --server-properties file://~/kafka_configuration
{
    "LatestRevision": {
        "Revision": 1,
        "CreationTime": "2019-09-06T14:39:37.708Z"
    },
    "CreationTime": "2019-09-06T14:39:37.708Z",
    "Name": "kafka-configuration",
    "Arn": "arn:aws:kafka:us-east-1:111122223333:configuration/kafka-
configuration/7e008070-6a08-445f-9fe5-36ccf630ecfd-3"
}
```

ここでは、//~/kafka_configuration は必要なプロパティ設定を使用して作成した設定ファイルです。

Amazon EC2 にインストールされている独自の Kafka インスタンスを使用している場合は、 がインスタンスに用意されているオプションを使用して新しいトピック AWS DMS を自動的に作成できるように、Kafka クラスターauto.create.topics.enable = true設定を変更します。

- AWS DMS は、トランザクションに関係なく、特定の Kafka トピック内の 1 つのデータレコード (メッセージ) としてソースデータベース内の 1 つのレコードに各更新を発行します。
- AWS DMS では、パーティションキーの次の 2 つの形式がサポートされています。
  - SchemaName.TableName: スキーマとテーブル名の組み合わせ。
  - ${AttributeName}: JSON のいずれかのフィールドの値、またはソースデータベースのテーブルのプライマリキー。
- BatchApply は Kafka エンドポイントではサポートされていません。Batch 適用 (例えば、BatchApplyEnabled のターゲットメタデータ タスク設定) を使用すると、Kafka ターゲットデータが失われる可能性があります。
- AWS DMS は、16 桁を超えるBigIntデータ型の値の移行をサポートしていません。この制限を回避するには、次の変換ルールを使用して BigInt 列を文字列に変換できます。変換ルールの詳細については、「 変換ルールおよび変換アクション」を参照してください。

```
{
    "rule-type": "transformation",
    "rule-id": "id",
    "rule-name": "name",
    "rule-target": "column",
```

```
    "object-locator": {
        "schema-name": "valid object-mapping rule action",
        "table-name": "",
        "column-name": ""
    },
    "rule-action": "change-data-type",
    "data-type": {
        "type": "string",
        "length": 20
    }
}
```

データを Kafka トピックに移行するためのオブジェクトマッピングの使用

AWS DMS は、テーブルマッピングルールを使用して、ソースからターゲット Kafka トピックにデータをマッピングします。ターゲットトピックにデータをマッピングするために、オブジェクトマッピングと呼ばれるテーブルマッピングルールのタイプを使用します。オブジェクトマッピングを使用して、ソースのデータレコードがどのように Kafka トピックに発行されたデータレコードにマッピングされるかを定義します。

Kafka トピックには、パーティションキー以外にプリセット構造はありません。

> **Note**
>
> オブジェクトマッピングは必ずしも使用する必要はありません。通常のテーブルマッピングは、さまざまな変換に使用できます。ただし、パーティションキータイプは次のデフォルト動作に従います。
>
> - プライマリキーはフルロードのパーティションキーとして使用されます。
> - 並行適用タスク設定が使用されていない場合は、`schema.table` が CDC のパーティションキーとして使用されます。
> - 並列適用タスク設定を使用する場合、プライマリキーは CDC のパーティションキーとして使用されます。

オブジェクトマッピングルールを作成するには、`object-mapping` として `rule-type` を指定します。このルールが、使用したいオブジェクトマッピングのタイプを指定します。

ルールの構造は次のとおりです。

```
{
    "rules": [
        {
            "rule-type": "object-mapping",
            "rule-id": "id",
            "rule-name": "name",
            "rule-action": "valid object-mapping rule action",
            "object-locator": {
                "schema-name": "case-sensitive schema name",
                "table-name": ""
            }
        }
    ]
}
```

AWS DMS は現在、 rule-actionパラメータの唯一の有効な値map-record-to-documentとして map-record-to-recordと をサポートしています。これらの設定は、exclude-columns 属性リストの一部として除外されない値に影響します。map-record-to-record および map-record-to-documentの値は、 がデフォルトでこれらのレコード AWS DMS を処理する方法を指定します。これらの値は、どのような方法でも属性マッピングに影響しません。

リレーショナルデータベースから Kafka トピックに移行する際に map-record-to-record を使用します。このルールタイプでは、Kafka トピックのパーティションキーとしてリレーショナルデータベースから taskResourceId.schemaName.tableName 値を使用し、ソースデータベース内の各列の属性を作成します。

map-record-to-record を使用する場合は、次の点に注意します。

- この設定は、exclude-columns リストで除外されている列にのみ影響します。
- このような列ごとに、 はターゲットトピックに対応する属性 AWS DMS を作成します。
- AWS DMS は、ソース列が属性マッピングで使用されているかどうかにかかわらず、対応するこの属性を作成します。

map-record-to-record を理解するための 1 つの方法は、実際の動作を確認することです。この例では、次の構造とデータを含むリレーショナルデータベースのテーブルの行から始めると想定してください。

FirstName	LastName	StoreId	HomeAddress	HomePhone	WorkAddress	WorkPhone	DateofBirth
Randy	Marsh	5	221B Baker Street	1234567890	31 Spooner Street, Quahog	9876543210	02/29/1988

この情報を Test という名前のスキーマから Kafka トピックに移行するには、データをターゲットストリームにマッピングするルールを作成します。以下のルールはマッピングを示しています。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "rule-action": "include",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            }
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",
            "rule-name": "DefaultMapToKafka",
            "rule-action": "map-record-to-record",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Customers"
            }
        }
    ]
}
```

Kafka トピックとパーティション キー (この場合は taskResourceId.schemaName.tableName) を指定すると、以下の説明は Kafka ターゲットトピックのサンプルデータを使用した結果のレコード形式を示します。

```
{
   "FirstName": "Randy",
   "LastName": "Marsh",
   "StoreId":  "5",
   "HomeAddress": "221B Baker Street",
   "HomePhone": "1234567890",
   "WorkAddress": "31 Spooner Street, Quahog",
   "WorkPhone": "9876543210",
   "DateOfBirth": "02/29/1988"
}
```

トピック

- 属性マッピングを使用したデータの再構築
- オブジェクトマッピングを使用したマルチトピックレプリケーション
- Apache Kafka のメッセージ形式

属性マッピングを使用したデータの再構築

属性マップを使用してデータを Kafka トピックに移行している間にデータを再構築できます。例えば、ソース内の複数のフィールドを結合してターゲット内に 1 つのフィールドを構成することもできます。以下の属性マップはデータを再構築する方法を示しています。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "rule-action": "include",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            }
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",
            "rule-name": "TransformToKafka",
            "rule-action": "map-record-to-record",
```

```
            "target-table-name": "CustomerData",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Customers"
            },
            "mapping-parameters": {
                "partition-key-type": "attribute-name",
                "partition-key-name": "CustomerName",
                "exclude-columns": [
                    "firstname",
                    "lastname",
                    "homeaddress",
                    "homephone",
                    "workaddress",
                    "workphone"
                ],
                "attribute-mappings": [
                    {
                        "target-attribute-name": "CustomerName",
                        "attribute-type": "scalar",
                        "attribute-sub-type": "string",
                        "value": "${lastname}, ${firstname}"
                    },
                    {
                        "target-attribute-name": "ContactDetails",
                        "attribute-type": "document",
                        "attribute-sub-type": "json",
                        "value": {
                            "Home": {
                                "Address": "${homeaddress}",
                                "Phone": "${homephone}"
                            },
                            "Work": {
                                "Address": "${workaddress}",
                                "Phone": "${workphone}"
                            }
                        }
                    }
                ]
            }
        }
    ]
}
```

partition-key の定数値を設定するには、partition-key 値を指定します。たとえば、すべてのデータを 1 つのパーティションに強制的に格納するためにこれを行うことができます。以下のマッピングはこの方法を示しています。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "1",
            "rule-name": "TransformToKafka",
            "rule-action": "map-record-to-document",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Customer"
            },
            "mapping-parameters": {
                "partition-key": {
                    "value": "ConstantPartitionKey"
                },
                "exclude-columns": [
                    "FirstName",
                    "LastName",
                    "HomeAddress",
                    "HomePhone",
                    "WorkAddress",
                    "WorkPhone"
                ],
                "attribute-mappings": [
                    {
                        "attribute-name": "CustomerName",
                        "value": "${FirstName},${LastName}"
                    },
                    {
```

```
                        "attribute-name": "ContactDetails",
                        "value": {
                            "Home": {
                                "Address": "${HomeAddress}",
                                "Phone": "${HomePhone}"
                            },
                            "Work": {
                                "Address": "${WorkAddress}",
                                "Phone": "${WorkPhone}"
                            }
                        }
                    },
                    {
                        "attribute-name": "DateOfBirth",
                        "value": "${DateOfBirth}"
                    }
                ]
            }
        }
    ]
}
```

> **Note**
>
> 特定のテーブル用のコントロールレコードの partition-key 値は、TaskId.SchemaName.TableName です。特定のタスク用のコントロールレコードの partition-key 値は、そのレコードの TaskId です。オブジェクトマッピングの partition-key 値を指定することは、コントロールレコードの partition-key には影響しません。

オブジェクトマッピングを使用したマルチトピックレプリケーション

デフォルトでは、 AWS DMS タスクはすべてのソースデータを次の Kafka トピックの 1 つに移行します。

- AWS DMS ターゲットエンドポイントのトピックフィールドで指定されているとおり。
- ターゲットエンドポイントの [トピック] フィールドが入力されておらず、Kafka auto.create.topics.enable 設定が true に設定されている場合、kafka-default-topic の指定に従う。

AWS DMS エンジンバージョン 3.4.6 以降では、 kafka-target-topic 属性を使用して、移行された各ソーステーブルを個別のトピックにマッピングできます。例えば、次のオブジェクトマッピングルールは、ソーステーブルを Customer とAddress をそれぞれ Kafka トピック customer_topic と address_topic に移行します。同時に、 は、Testスキーマ内のテーブルを含む他のすべてのソースBillsテーブルを、ターゲットエンドポイントで指定されたトピック AWS DMS に移行します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "rule-action": "include",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            }
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",
            "rule-name": "MapToKafka1",
            "rule-action": "map-record-to-record",
            "kafka-target-topic": "customer_topic",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Customer"
            },
            "partition-key": {"value": "ConstantPartitionKey" }
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "3",
            "rule-name": "MapToKafka2",
            "rule-action": "map-record-to-record",
            "kafka-target-topic": "address_topic",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Address"
            },
            "partition-key": {"value": "HomeAddress" }
        },
```

```
        {
            "rule-type": "object-mapping",
            "rule-id": "4",
            "rule-name": "DefaultMapToKafka",
            "rule-action": "map-record-to-record",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Bills"
            }
        }
    ]
}
```

Kafka マルチトピックレプリケーションを使用すると、単一のレプリケーションタスクでソーステーブルをグループ化して個別の Kafka トピックに移行できます。

Apache Kafka のメッセージ形式

JSON 出力は、単にキーと値のペアのリストです。

RecordType

レコードタイプはデータまたはコントロールのいずれかです。データレコードは、ソースの実際の行を表します。コントロールレコードは、タスクの再起動など、ストリーム内の重要なイベント用です。

操作

データレコードの場合、オペレーションは `load`、`insert`、`update`、または `delete` です。

コントロールレコードの場合、オペレーションは `create-table`、`rename-table`、`drop-table`、`change-columns`、`add-column`、`drop-column`、`rename-column`、`column-type-change` です。

SchemaName

レコードのソーススキーマ。コントロールレコードの場合、このフィールドは空です。

TableName

レコードのソーステーブル。コントロールレコードの場合、このフィールドは空です。

タイムスタンプ

JSON メッセージが構築された時刻のタイムスタンプ。このフィールドは ISO 8601 形式でフォーマットされます。

次の JSON メッセージの例は、追加メタデータをすべて含むデータ型メッセージを示しています。

```
{
   "data":{
      "id":100000161,
      "fname":"val61s",
      "lname":"val61s",
      "REGION":"val61s"
   },
   "metadata":{
      "timestamp":"2019-10-31T22:53:59.721201Z",
      "record-type":"data",
      "operation":"insert",
      "partition-key-type":"primary-key",
      "partition-key-value":"sbtest.sbtest_x.100000161",
      "schema-name":"sbtest",
      "table-name":"sbtest_x",
      "transaction-id":9324410911751,
      "transaction-record-id":1,
      "prev-transaction-id":9324410910341,
      "prev-transaction-record-id":10,
      "commit-timestamp":"2019-10-31T22:53:55.000000Z",
      "stream-position":"mysql-bin-
changelog.002171:36912271:0:36912333:9324410911751:mysql-bin-changelog.002171:36912209"
   }
}
```

次の JSON メッセージの例は、コントロールタイプのメッセージを示しています。

```
{
   "control":{
      "table-def":{
         "columns":{
            "id":{
               "type":"WSTRING",
               "length":512,
               "nullable":false
            },
            "fname":{
               "type":"WSTRING",
               "length":255,
               "nullable":true
```

```
            },
            "lname":{
               "type":"WSTRING",
               "length":255,
               "nullable":true
            },
            "REGION":{
               "type":"WSTRING",
               "length":1000,
               "nullable":true
            }
         },
         "primary-key":[
            "id"
         ],
         "collation-name":"latin1_swedish_ci"
      }
   },
   "metadata":{
      "timestamp":"2019-11-21T19:14:22.223792Z",
      "record-type":"control",
      "operation":"create-table",
      "partition-key-type":"task-id",
      "schema-name":"sbtest",
      "table-name":"sbtest_t1"
   }
}
```

AWS Database Migration Service のターゲットとしての Amazon OpenSearch Service クラスターの使用

AWS DMS を使用して、データを Amazon OpenSearch Service (OpenSearch Service) に移行できます。OpenSearch Service は、OpenSearch Service クラスターのデプロイ、運用、スケーリングが簡単に実行できるマネージドサービスです。

OpenSearch Service では、インデックスとドキュメントを操作します。インデックスは、ドキュメントのコレクションであり、ドキュメントは、スカラー値、配列、その他のオブジェクトなどの JSON オブジェクトです。OpenSearch は、JSON ベースのクエリ言語であり、インデックス内のデータをクエリして、対応するドキュメントを取得できます。

AWS DMS が OpenSearch Service のターゲットエンドポイントのインデックスを作成する際、ソースエンドポイントのテーブルごとに単一のインデックスを作成します。OpenSearch Service イン

デックスの作成コストは、いくつかの要因によって異なります。このような要因には、作成されたインデックスの数、インデックス内のデータの合計量、OpenSearch がドキュメントごとに保存する少量のメタデータがあります。

移行の範囲に適したコンピューティングリソースとストレージリソースを使用して OpenSearch サービスクラスターを設定します。使用するレプリケーションタスクに応じて、次の要素を考慮することをお勧めします。

- フルデータロードの場合、移行するデータの合計量と転送の速度を考慮します。
- 継続的に変更をレプリケートする場合、更新頻度、エンドツーエンドのレイテンシーの要件を考慮します。

また、OpenSearch クラスターでインデックスを設定し、ドキュメントの数に細心の注意を払います。

マルチスレッドのフルロードタスク設定

転送速度を向上するため、AWS DMS は、OpenSearch Service ターゲットクラスターへのマルチスレッドのフルロードをサポートしています。AWS DMS は、次を含むタスク設定でこのマルチスレッドをサポートします。

- `MaxFullLoadSubTasks` – 並列ロードするソーステーブルの最大数を指定するには、このオプションを使用します。DMS は、専用のサブタスクを使用して、対応する OpenSearch Service ターゲットインデックスに各テーブルをロードします。デフォルトは 8、最大値は 49 です。
- `ParallelLoadThreads` – AWS DMS が各テーブルを OpenSearch Service ターゲットインデックスにロードするために使用するスレッドの数を指定するには、このオプションを使用します。OpenSearch Service ターゲットの最大値は 32 です。この上限を引き上げるように要請できます。

  > **Note**
  >
  > `ParallelLoadThreads` をデフォルト値 (0) から変更しない場合、AWS DMS は一度に単一のレコードを転送します。この方法の場合、OpenSearch Service クラスターに過度の負荷がかかります。このオプションを 1 以上に設定していることを確認します。

- `ParallelLoadBufferSize` – 並列ロードスレッドが OpenSearch Service ターゲットにデータをロードするために使用するバッファに保存するレコードの最大数を指定するには、このオプションを使用します。デフォルト値は 50。最大値は 1,000 です。この設定は

ParallelLoadThreads で使用します。ParallelLoadBufferSize は、複数のスレッドがある場合にのみ有効です。

DMS がマルチスレッドを使用して OpenSearch Service クラスターをロードする方法の詳細については、AWS ブログ記事「Scale Amazon OpenSearch Service for AWS Database Migration Service migrations」を参照してください。

マルチスレッド CDC ロードタスクの設定

タスク設定を使用して PutRecords API コールの動作を変更するなど、OpenSearch Service ターゲットクラスターの変更データキャプチャ (CDC) のパフォーマンスを向上できます。これを行うには、ParallelApply* タスク設定を使用して、同時スレッドの数、スレッドあたりのキューの数、バッファに格納するレコードの数を指定します。例えば、CDC ロードを実行し、32 本のスレッドを並列に適用するとします。また、スレッドあたり 64 のキューにアクセスして、バッファあたり 50 のレコードを保存する必要があります。

> **Note**
>
> CDC から Amazon OpenSearch Service ターゲットエンドポイントへの ParallelApply* タスク設定の使用のサポートは、AWS DMS バージョン 3.4.0 以降で利用できます。

CDC のパフォーマンス向上のため、AWS DMS は次のタスク設定をサポートしています。

- ParallelApplyThreads – データレコードを OpenSearch Service ターゲットエンドポイントにプッシュするために CDC ロード中に AWS DMS が使用する同時スレッドの数を指定します。デフォルト値はゼロ (0)、最大値は 32 です。
- ParallelApplyBufferSize – CDC ロード中に OpenSearch Service ターゲットエンドポイントにプッシュする同時スレッドの各バッファキューに格納するレコードの最大数を指定します。デフォルト値は 100、最大値は 1,000 です。このオプションは、ParallelApplyThreads が複数のスレッドを指定する場合に使用します。
- ParallelApplyQueuesPerThread – CDC 中に各スレッドがキューからデータレコードを取り出して OpenSearch Service エンドポイントのバッチロードを生成するためにアクセスするキューの数を指定する。

ParallelApply* タスク設定を使用する場合、partition-key-type のデフォルトは schema-name.table-name ではなく、テーブルの primary-key となります。

リレーショナルデータベースから OpenSearch Service インデックスへの移行

AWS DMS は、OpenSearch Service のスカラーデータ型へのデータの移行をサポートしています。Oracle や MySQL などのリレーショナルデータベースから OpenSearch Service に移行する場合は、データを格納する方法を再編成が必要となる場合があります。

AWS DMS は、次の OpenSearch Service スカラーデータ型をサポートしています。

- Boolean
- Date
- Float
- Int
- String

AWS DMS は、日付型のデータを文字列に変換します。日付を解釈するカスタムマッピングは指定できます。

AWS DMS は LOB データ型の移行をサポートしていません。

Amazon OpenSearch Service を AWS Database Migration Service のターゲットとして使用するための前提条件

OpenSearch Service データベースを AWS DMS のターゲットとして使用する作業を開始する前に、必ず AWS Identity and Access Management (IAM) ロールを作成します。このロールにより、AWS DMS がターゲットエンドポイントで OpenSearch Service インデックスにアクセスできるようになります。アクセス許可の最小設定は、次の IAM ポリシーに示されています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "Service": "dms.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
```

```
        }
    ]
}
```

OpenSearch Service への移行に使用するロールには、次のアクセス権限が必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                      "es:ESHttpDelete",
                      "es:ESHttpGet",
                      "es:ESHttpHead",
                      "es:ESHttpPost",
                      "es:ESHttpPut"
                    ],
            "Resource": "arn:aws:es:region:account-id:domain/domain-name/*"
        }
    ]
}
```

上記の例で、*region* を AWS リージョン識別子、*account-id* を AWS アカウント ID、*domain-name* を Amazon OpenSearch Service ドメイン名に置き換えます。例えば、`arn:aws:es:us-west-2:123456789012:domain/my-es-domain`と指定します。

OpenSearch Service を AWS DMS のターゲットとして使用する場合のエンドポイントの設定

追加の接続属性の使用と同様、エンドポイントの設定を使用して、ターゲットの OpenSearch Service データベースを設定できます。ターゲットエンドポイントを作成する際に、AWS DMS コンソールを使用するか、AWS CLI で `--elasticsearch-settings '{"EndpointSetting": "value", ...}'` の JSON 構文を指定して `create-endpoint` コマンドを使用して設定を指定します。

次の表は、ターゲットとして OpenSearch Service を使用できるエンドポイント設定を説明しています。

属性名	有効値	デフォルト値と説明
FullLoadErrorPercentage	0 より大きく、100 より小さい正の整数。	10 – フルロードタスクの場合、この属性はタスクが失敗する前に許容されるエラーのしきい値を決定する。例えば、ソースエンドポイントに 1,500 行あり、このパラメータが 10 に設定されているとする場合、ターゲットエンドポイントに書き込み時に AWS DMS で 150 を超えるエラー (行数の 10%) が発生した場合、そのタスクは失敗する。
ErrorRetryDuration	0 より大きい正の整数。	300 – ターゲットエンドポイントでエラーが発生した場合、AWS DMS はこの秒数で再試行する。再試行しない場合、タスクは失敗する。

Amazon OpenSearch Service を AWS Database Migration Service のターゲットとして使用する場合の制限

Amazon OpenSearch Service をターゲットとして使用する場合、次の制限が適用されます。

- OpenSearch Service は、動的マッピング (自動推測) を使用して、移行されたデータに使用するデータ型を決定します。
- OpenSearch サービスは、各ドキュメントを一意の ID で保存します。ID の例は次のとおりです。

  ```
  "_id": "D359F8B537F1888BC71FE20B3D79EAE6674BE7ACA9B645B0279C7015F6FF19FD"
  ```

  各ドキュメント ID の長さは 64 バイトであるため、これをストレージ要件として想定します。例えば、AWS DMS ソースから 100,000 行を移行する場合、結果の OpenSearch Service インデックスにはさらに 6,400,000 バイトのストレージが必要になります。
- OpenSearch Service では、プライマリキー属性を更新できません。これによりターゲットで不要なデータが発生する可能性があるため、この制限は変更データキャプチャ (CDC) で継続的なレプリケーションを使用する場合に重要です。CDC モードでは、プライマリキーは 32 バイト長の SHA256 値にマップされます。この値はユーザーが判読できる 64 バイト文字列に変換され、OpenSearch サービスのドキュメント ID として使用されます。
- AWS DMS は移行できないアイテムを検出すると、Amazon CloudWatch Logs にエラーメッセージを書き込みます。この動作は、例外テーブルにエラーを書き込むその他の AWS DMS ターゲットエンドポイントの動作とは異なります。

- AWS DMS は、管理ユーザーとパスワードによるきめ細かいアクセス制御が有効になっている Amazon ES クラスターへの接続をサポートしていません。
- AWS DMSは OpenSearch Service サーバーレスをサポートしていません。
- OpenSearch サービスは、既存のインデックスへのデータの書き込みをサポートしていません。

Amazon OpenSearch Service のターゲットデータ型

異種データベースからデータを移行する場合、AWS DMS サービスはソースデータベースのデータ型を、AWS DMS データ型と呼ばれる中間のデータ型にマップします。その後このサービスは、中間データ型をターゲットデータ型にマップします。次の表は、AWS DMS の各データ型とマップ先の OpenSearch Service のデータ型を説明しています。

AWS DMS のデータ型	OpenSearch Service のデータ型
Boolean	boolean
Date	string
Time	date
Timestamp	date
INT4	integer
Real4	float
UINT4	integer

AWS DMS のデータ型の詳細については、「AWS Database Migration Service のデータ型」を参照してください。

AWS Database Migration Service のターゲットとしての Amazon DocumentDB の使用

AWS DMS がサポートする Amazon DocumentDB (MongoDB 互換) のバージョンについては、「のターゲット AWS DMS」を参照してください。AWS DMS を使用して、AWS DMS がサポートするソースデータエンジンのいずれかから、任意の Amazon DocumentDB (MongoDB 互換) データベー

スにデータを移行できます。ソースエンジンは、Amazon RDS、Aurora、Amazon S3 などの AWS が管理するサービス上に配置できます。データベースエンジンは、Amazon EC2 またはオンプレミスで実行される MongoDB などのセルフマネージド型データベースに配置することもできます。

AWS DMS を使用すると、Amazon DocumentDB データベース、コレクション、またはドキュメントにソースデータをレプリケートできます。

> **Note**
>
> ソースエンドポイントが MongoDB または Amazon DocumentDB の場合は、[ドキュメントモード] で移行します。

MongoDB は、データをバイナリ JSON 形式 (BSON) で保存します。AWS DMS は、Amazon DocumentDB でサポートされているすべての BSON データ型をサポートします。このようなデータ型のリストについては、「Amazon DocumentDB デベロッパーガイド」の「Supported MongoDB APIs, operations, and data types」を参照してください。

ソースエンドポイントがリレーショナルデータベースである場合は、AWS DMS は次のとおりデータベースオブジェクトを Amazon DocumentDBにマップします。

- リレーショナルデータベースまたはデータベーススキーマは、Amazon DocumentDB データベースにマップされます。
- リレーショナルデータベースのテーブルは、Amazon DocumentDB 内のコレクションにマップされます。
- リレーショナルテーブルのレコードは、Amazon DocumentDB 内のドキュメントにマップされます。各ドキュメントはソースレコードのデータから構築されます。

ソースエンドポイントが Amazon S3 の場合、結果として得られる Amazon DocumentDB オブジェクトは、Amazon S3 の AWS DMS マッピングルールに対応しています。例えば次のような URI がある場合、

```
s3://mybucket/hr/employee
```

AWS DMS は、`mybucket` 内のオブジェクトを次のとおり Amazon DocumentDB にマップします。

- 最上位の URI パート (`hr`) は、 Amazon DocumentDB データベースにマップされます。

- 次の URI パート (employee) は、Amazon DocumentDB のコレクションにマップされます。
- employee の各オブジェクトは Amazon DocumentDB 内のドキュメントにマップされます。

Amazon S3 のマッピングルールの詳細については、「のソースとしての Amazon S3 の使用 AWS DMS」を参照してください。

Amazon DocumentDB のエンドポイント設定

AWS DMS バージョン 3.5.0 以降では、p並列スレッドと一括オペレーションのタスク設定を調整することで、Amazon DocumentDB エンドポイントの変更データキャプチャ (CDC) のパフォーマンスを向上できます。これを行うには、ParallelApply* タスク設定を使用して、同時スレッドの数、スレッドあたりのキューの数、バッファに格納するレコードの数を指定します。例えば、CDC ロードを実行し、128 本のスレッドを並列に適用するとします。また、スレッドあたり 64 のキューにアクセスして、バッファあたり 50 のレコードを保存する必要があります。

CDC のパフォーマンス向上のため、AWS DMS は次のタスク設定をサポートしています。

- ParallelApplyThreads – データレコードを Amazon DocumentDB ターゲットエンドポイントにプッシュするために CDC ロード中に AWS DMS が使用する同時スレッドの数を指定します。デフォルト値はゼロ (0)、最大値は 32 です。
- ParallelApplyBufferSize – CDC ロード中に Amazon DocumentDB ターゲットエンドポイントにプッシュする同時スレッドの各バッファキューに格納するレコードの最大数を指定します。デフォルト値は 100、最大値は 1,000 です。このオプションは、ParallelApplyThreads が複数のスレッドを指定する場合に使用します。
- ParallelApplyQueuesPerThread – CDC 中に各スレッドがキューからデータレコードを取り出して Amazon DocumentDB エンドポイントのバッチロードを生成するためにアクセスするキューの数を指定する。デフォルトは 1 です。最大数は 512 です。

AWS DMS のターゲットとして Amazon DocumentDB を使用する方法の詳細については、次のセクションを参照してください。

トピック

- ソースから Amazon DocumentDB ターゲットへのデータのマッピング
- ターゲットとしての Amazon DocumentDB Elastic Cluster への接続
- Amazon DocumentDB をターゲットとした継続的なレプリケーション
- Amazon DocumentDB をターゲットとして使用する場合の制限

- ターゲットとしての Amazon DocumentDB のエンドポイントの設定
- Amazon DocumentDB のターゲットデータ型

> **Note**
>
> ステップバイステップの移行ウォークスルーについては、「AWS Database Migration Service Step-by-Step Migration Guide」の「Migrating from MongoDB to Amazon DocumentDB」を参照してください。

ソースから Amazon DocumentDB ターゲットへのデータのマッピング

AWS DMS は、ソースエンドポイントからレコードを読み取り、読み取ったデータに基づいて JSON ドキュメントを構築します。各 JSON ドキュメントでは、AWS DMS は、一意の識別子として機能する _id フィールドを決定する必要があります。次に、その _id フィールドをプライマリキーとして使用して、JSON ドキュメントを Amazon DocumentDB コレクションに書き込みます。

単一列のソースデータ

ソースデータが 1 つの列で構成されている場合、そのデータは文字列型である必要があります。(ソースエンジンによっては、実際のデータ型が VARCHAR、NVARCHAR、TEXT、LOB、CLOB などになります)。AWS DMS は、データが有効な JSON ドキュメントであると想定し、データをそのまま Amazon DocumentDB にレプリケートします。

結果の JSON ドキュメントに _id という名前のフィールドが含まれている場合は、そのフィールドが Amazon DocumentDB で一意の _id として使用されます。

JSON に _id フィールドが含まれていない場合は、Amazon DocumentDB が自動的に _id 値を生成します。

複数列のソースデータ

ソースデータが複数の列で構成されている場合、AWS DMS はそのすべての列から JSON ドキュメントを構築します。AWS DMS は、ドキュメントの _id フィールドを特定するために、次のとおり処理します。

- いずれかの列が _id という名前の場合は、その列のデータがターゲット _id として使用されます。

- ソースデータに _id 列はなく、プライマリキーまたは一意のインデックスがある場合、AWS DMS はそのキーまたはインデックスの値を _id 値として使用します。プライマリキーまたは一意のインデックスのデータも、JSON ドキュメント内の明示的なフィールドとして表示されます。
- _id 列がなく、プライマリキーも、一意のインデックスもない場合は、Amazon DocumentDB は自動的に _id 値を自動的に生成します。

ターゲットエンドポイントでのデータ型の強制変換

AWS DMS では、Amazon DocumentDB ターゲットエンドポイントに書き込む際にデータ構造を変更できます。このような変更をリクエストするには、ソースエンドポイントで列名やテーブル名を変更するか、タスクの実行時に適用される変換ルールを指定します。

ネストされた JSON ドキュメント (json_ prefix) の使用

データ型を強制変換するには、ソース列名に `json_` のプレフィックスを付けます (`json_`*`columnName`*など)。これは、手動でも、変換を使用しても実行できます。この場合、列は文字列フィールドとしてではなく、ターゲットドキュメント内にネストされた JSON ドキュメントとして作成されます。

例えば、MongoDB ソースエンドポイントから次のドキュメントを移行するとします。

```
{
    "_id": "1",
    "FirstName": "John",
    "LastName": "Doe",
    "ContactDetails": "{"Home": {"Address": "Boston","Phone": "1111111"},"Work":
 { "Address": "Boston", "Phone": "2222222222"}}"
}
```

ソースデータ型を強制変換しない場合、埋め込まれている `ContactDetails` ドキュメントは文字列として移行されます。

```
{
    "_id": "1",
    "FirstName": "John",
    "LastName": "Doe",
    "ContactDetails": "{\"Home\": {\"Address\": \"Boston\",\"Phone\": \"1111111\"},
\"Work\": { \"Address\": \"Boston\", \"Phone\": \"2222222222\"}}"
}
```

変換ルールを追加すると、ContactDetails を JSON オブジェクトに強制変換することができます。例えば、元のソース列名が ContactDetails の場合、データ型をネストされた JSON に強制変換するには、ソースに「*json_*」プレフィックスを手動で追加するか、変換ルールを使用して、ソースエンドポイントの列名前「JSON_ContactDetails」に変更する必要があります。例えば、次の変換ルールを使用できます。

```
{
    "rules": [
    {
    "rule-type": "transformation",
    "rule-id": "1",
    "rule-name": "1",
    "rule-target": "column",
    "object-locator": {
    "schema-name": "%",
    "table-name": "%",
    "column-name": "ContactDetails"
     },
    "rule-action": "rename",
    "value": "json_ContactDetails",
    "old-value": null
    }
    ]
}
```

AWS DMS は、次のとおり ContactDetails フィールドをネストされた JSON としてレプリケートします。

```
{
    "_id": "1",
    "FirstName": "John",
    "LastName": "Doe",
    "ContactDetails": {
        "Home": {
            "Address": "Boston",
            "Phone": "1111111111"
        },
        "Work": {
            "Address": "Boston",
            "Phone": "2222222222"
```

```
        }
    }
}
```

JSON 配列 (array_ prefix) の使用

データ型を強制変換するには、列名に `array_` のプレフィックスを付けます (`array_`*`columnName`*など)。これは、手動でも、変換を使用しても実行できます。この場合、AWS DMS は列を JSON 配列と見なし、ターゲットドキュメント内に JSON 配列を作成します。

MongoDB ソースエンドポイントから次のドキュメントを移行するとします。

```
{
    "_id" : "1",
    "FirstName": "John",
    "LastName": "Doe",

    "ContactAddresses": ["Boston", "New York"],

    "ContactPhoneNumbers": ["1111111111", "2222222222"]
}
```

ソースデータ型を強制変換しない場合、埋め込まれている `ContactDetails` ドキュメントは文字列として移行されます。

```
{
    "_id": "1",
    "FirstName": "John",
    "LastName": "Doe",

    "ContactAddresses": "[\"Boston\", \"New York\"]",

    "ContactPhoneNumbers": "[\"1111111111\", \"2222222222\"]"

}
```

変換ルールを追加すると、`ContactAddress` と `ContactPhoneNumbers` を次の表の例のとおり、JSON 配列に強制変換できます。

元のソース列名	変更後のソース列名
ContactAddress	array_ContactAddress
ContactPhoneNumbers	array_ContactPhoneNumbers

これにより、AWS DMS は、次のようにContactAddress と ContactPhoneNumbers をレプリケートします。

```
{
    "_id": "1",
    "FirstName": "John",
    "LastName": "Doe",
    "ContactAddresses": [
        "Boston",
        "New York"
    ],
    "ContactPhoneNumbers": [
        "1111111111",
        "2222222222"
    ]
}
```

TLS を使用した Amazon DocumentDB への接続

デフォルトでは、新しく作成された Amazon DocumentDB クラスターは、Transport Layer Security (TLS) を使用したセキュアな接続のみを受け入れます。TLS が有効になっている場合、Amazon DocumentDB へのすべての接続でパブリックキーが必要になります。

Amazon DocumentDB のパブリックキーを取得するには、AWS がホストする Amazon S3 バケットから rds-combined-ca-bundle.pem ファイルをダウンロードします。このファイルのダウンロードの詳細については、「Amazon DocumentDB デベロッパーガイド」の「Encrypting connections using TLS」を参照してください。

この .pem ファイルをダウンロードした後、ファイルに含まれるパブリックキーを AWS DMS に次のとおりインポートできます。

AWS Management Console

パブリックキー (.pem) ファイルをインポートするには

1. https://console.aws.amazon.com/dms で AWS DMS コンソールを開きます。
2. ナビゲーションペインで [証明書] を選択します。
3. [証明書をインポート] をクリックして、次を実行します。

   - [証明書の識別子] で、この証明書の一意の名前を入力します。例えば `docdb-cert` と入力します。
   - [ファイルをインポート] をクリックして、.pem ファイルを保存した場所に移動します。

   すべての設定が正しいことを確認したら、[新しい CA 証明書の追加] をクリックします。

AWS CLI

次の例のとおり、`aws dms import-certificate` コマンドを使用します。

```
aws dms import-certificate \
    --certificate-identifier docdb-cert \
    --certificate-pem file://./rds-combined-ca-bundle.pem
```

AWS DMS のターゲットエンドポイントを作成する際、証明書の識別子 (`docdb-cert` など) を指定します。また、SSL モードパラメータを [`verify-full`] に設定します。

ターゲットとしての Amazon DocumentDB Elastic Cluster への接続

AWS DMS バージョン 3.4.7 以降では、ターゲットの Amazon DocumentDB エンドポイントを Elastic クラスターとして作成できます。ターゲットエンドポイントを Elastic クラスターとして作成する場合、既存の SSL 証明書は機能しないため、Amazon DocumentDB Elastic クラスターエンドポイントに新しい SSL 証明書をアタッチする必要があります。

新しい SSL 証明書を Amazon DocumentDB Elastic クラスターエンドポイントにアタッチするには

1. ブラウザで https://www.amazontrust.com/repository/SFSRootCAG2.pem を開いて、コンテンツを `.pem` に一意のファイル名を付けて保存します (`SFSRootCAG2.pem` など)。この証明書ファイルは、後半の手順でインポートする必要がありますす。
2. Elastic クラスターエンドポイントを作成して、次のオプションを設定します。

a. [エンドポイント設定] の下で、[新しい CA 証明書の追加] をクリックします。

b. [証明書の識別子] には、**SFSRootCAG2.pem** を入力します。

c. [証明書ファイルのインポート] では、[ファイルを選択] をクリックして、前のタスクでダウンロードした SFSRootCAG2.pem ファイルに移動します。

d. ファイルを選択して、ダウンロードした SFSRootCAG2.pem ファイルを選択します。

e. [証明書をインポート] をクリックします。

f. [証明書の選択] ドロップダウンで、[SFSRootCAG2.pem] を選択します。

ダウンロードした SFSRootCAG2.pem ファイルからの新しい SSL 証明書が、Amazon DocumentDB Elastic クラスターエンドポイントにアタッチされます。

Amazon DocumentDB をターゲットとした継続的なレプリケーション

Amazon DocumentDB をターゲットとした継続的なレプリケーション (変更データキャプチャ、CDC) が有効になっている場合、AWS DMS バージョン 3.5.0 以降では以前のリリースと比べてパフォーマンスが 20 倍向上しています。以前のリリースでは、AWS DMS のレコード処理は 1 秒あたり最大 250 レコードでしたが、現在 AWS DMS は 1 秒あたり約 5,000 レコードを処理できるようになりました。AWS DMS を使用すると、Amazon DocumentDB 内のドキュメントのソースとの同期も確実に維持されます。ソースレコードが作成または更新されると、AWS DMS はまず、次の操作を実行し、影響を受ける Amazon DocumentDB レコードを判断する必要があります。

- ソースレコードに _id という名前の列がある場合、その列の値を使用して Amazon DocumentDB コレクションの対応する _id が特定されます。
- ソースデータに _id 列はなく、プライマリキーまたは一意のインデックスがある場合、AWS DMS はそのキーまたはインデックスの値を Amazon DocumentDB コレクションの _id として使用します。
- ソースレコードに _id 列、プライマリキー、一意のインデックスもない場合、AWS DMS はすべてのソース列を Amazon DocumentDB コレクションに対応するフィールドと照合します。

新しいソースレコードが作成されると、AWS DMS は対応するドキュメントを Amazon DocumentDB に書き込みます。既存のソースレコードが更新されると、AWS DMS は Amazon DocumentDB のターゲットドキュメントの対応するフィールドを更新します。ターゲットドキュメントには存在してもソースレコードには存在しないフィールドは、そのままにします。

ソースレコードが削除されると、AWS DMS は対応するドキュメントを Amazon DocumentDB から削除します。

ソースでの構造の変更 (DDL)

継続的なレプリケーションでは、ソースのデータ構造 (テーブル、列など) の変更が Amazon DocumentDB の対応するデータ構造にプロパゲートされます。リレーショナルデータベースでは、このような変更をデータ定義言語 (DDL) ステートメントを使用して開始します。次の表は、AWS DMS でこのような変更がどのように Amazon DocumentDB にプロパゲートされるかを説明しています。

ソースの DDL	Amazon DocumentDB ターゲットへの影響
`CREATE TABLE`	空のコレクションが作成される。
テーブル名を変更するステートメント (`RENAME TABLE`、`ALTER TABLE...RENAME` など)	コレクション名を変更する。
`TRUNCATE TABLE`	コレクションからすべてのドキュメントが削除される。ただし、`HandleSourceTableTruncated` が `true` の場合のみ。詳細については、「変更処理の DDL 処理のタスク設定」を参照してください。
`DROP TABLE`	コレクションを削除する。ただし、`HandleSourceTableDropped` が `true` の場合のみ。詳細については、「変更処理の DDL 処理のタスク設定」を参照してください。
テーブルに列を追加するステートメント (`ALTER TABLE...ADD` など)	この DDL ステートメントは無視され、警告が表示される。初めて `INSERT` がソースで実行される際、ターゲットドキュメントに新しいフィールドが追加される。
`ALTER TABLE...RENAME COLUMN`	この DDL ステートメントは無視され、警告が表示される。初めて `INSERT` がソースで実行さ

ソースの DDL	Amazon DocumentDB ターゲットへの影響
	れる際、ターゲットドキュメントに新しい名前のフィールドが追加される。
`ALTER TABLE...DROP COLUMN`	この DDL ステートメントは無視され、警告が表示される。
列データ型を変更するステートメント (`ALTER COLUMN...MODIFY` など)	この DDL ステートメントは無視され、警告が表示される。初めて新しいデータ型を使用した `INSERT` がソースで実行された際、その新しいデータ型のフィールドを使用してターゲットドキュメントが作成される。

Amazon DocumentDB をターゲットとして使用する場合の制限

Amazon DocumentDB を AWS DMS のターゲットとして使用する場合、次の制限が適用されます。

- Amazon DocumentDB では、コレクション名とキー名にはドル記号 ($) を使用できません。また、データベース名に Unicode 文字は使用できません。
- AWS DMS は、複数のソーステーブルの単一の Amazon DocumentDB コレクションへのマージをサポートしていません。
- AWS DMS は、プライマリキーがないソーステーブルからの変更を処理する際、テーブルの LOB 列を無視します。
- [Change table] オプションが有効になっていて、AWS DMS が「_id」という名前のソース列を検出した場合、この列は変更テーブル「__id」(アンダースコア 2 つ) として表示されます。
- ソースエンドポイントとして Oracle を選択する場合は、Oracle ソースで完全なサプリメンタルロギングが有効になっている必要があります。有効になっていない場合、ソースに変更されていない列があると、データは null 値として Amazon DocumentDB にロードされます。
- レプリケーションタスク設定 `TargetTablePrepMode:TRUNCATE_BEFORE_LOAD` は、DocumentDB ターゲットエンドポイントでの使用がサポートされていません。

ターゲットとしての Amazon DocumentDB のエンドポイントの設定

追加の接続属性の使用と同様、エンドポイントの設定を使用して、ターゲットの Amazon DocumentDB データベースを設定できます。ターゲットエンドポイントを作成する際に、AWS

DMS コンソールを使用するか、AWS CLI で `--doc-db-settings '{"EndpointSetting": "value", ...}'` の JSON 構文を指定して `create-endpoint` コマンドを使用して設定を指定します。

次の表は、ターゲットとして Amazon DocumentDB を使用できるエンドポイント設定を説明しています。

属性名	有効値	デフォルト値と説明

`replicateShardCollections`	boolean
	
	`true`
	
	`false`
		`true` の場合、このエンドポイント設定には次のような影響がおよび、次のとおりの制限がある。
		• AWS DMS は、データをターゲットシャードコレクションにレプリケートできる。この設定は、ターゲットの DocumentDB エンドポイントが Elastic クラスターである場合にのみ適用される。
		• `TargetTablePrepMode` を `DO_NOTHING` に設定する必要がある。
		• AWS DMS は、移行中に`useUpdateLookup` を 自動的に `false` に設定する。

Amazon DocumentDB のターゲットデータ型

次の表は、AWS DMS を使用する場合にサポートされるターゲットの Amazon DocumentDB データベースのデータ型と、AWS DMS のデータ型のデフォルトマッピングを説明しています。AWS DMS データ型の詳細については、「AWS Database Migration Service のデータ型」を参照してください。

AWS DMS のデータ型	Amazon DocumentDB のデータ型
BOOLEAN	Boolean
BYTES	Binary data
DATE	Date
TIME	String (UTF8)

AWS DMS のデータ型	Amazon DocumentDB のデータ型
DATETIME	Date
INT1	32 ビット整数
INT2	32 ビット整数
INT4	32 ビット整数
INT8	64 ビット整数
NUMERIC	String (UTF8)
REAL4	Double
REAL8	Double
STRING	データが JSON として認識された場合、AWS DMS はドキュメントとして Amazon DocumentDB に移行する。それ以外の場合、データは文字列 (UTF8) にマップされる。
UINT1	32 ビット整数
UINT2	32 ビット整数
UINT4	64 ビット整数
UINT8	String (UTF8)
WSTRING	データが JSON として認識された場合、AWS DMS はドキュメントとして Amazon DocumentDB に移行する。それ以外の場合、データは文字列 (UTF8) にマップされる。
BLOB	Binary
CLOB	データが JSON として認識された場合、AWS DMS はドキュメントとして Amazon DocumentDB に移行する。それ以外の場合、データは文字列 (UTF8) にマップされる。

AWS DMS のデータ型	Amazon DocumentDB のデータ型
NCLOB	データが JSON として認識された場合、AWS DMS はドキュメントとして Amazon DocumentDB に移行する。それ以外の場合、データは文字列 (UTF8) にマップされる。

AWS Database Migration Service のターゲットとしての Amazon Neptune の使用

Amazon Neptune は、高速で信頼性に優れたフルマネージド型のグラフデータベースサービスです。Amazon Neptune を使用すると、高度に接続されたデータセットを使用するアプリケーションを容易に構築して実行できます。Neptune の中核をなすのは、専用の高パフォーマンスグラフデータベースエンジンです。このエンジンは、数 10 億の関係を保存し、ミリ秒のレイテンシーでグラフをクエリするために最適化されています。Neptune は、人気の高いグラフクエリ言語 Apache TinkerPop Gremlin と W3C の SPARQL をサポートしています。Amazon Neptune の詳細については、「Amazon Neptune ユーザーガイド」の「What is Amazon Neptune?」を参照してください。

Neptune のようなグラフデータベースを使用しない場合、リレーショナルデータベースで高度に接続されたデータをモデル化することになります。データには動的接続がある可能性があるため、そのようなデータソースを使用するアプリケーションは、接続されたデータクエリを SQL でモデル化する必要があります。この方法の場合、グラフクエリの SQL への変換に追加のレイヤーを構築する必要があります。また、リレーショナルデータベースは、スキーマが柔軟でないという点があります。接続をモデル変更するためにスキーマを変更すると、新しいスキーマをサポートするためにダウンタイムとクエリ変換の追加のメンテナンスが必要になります。クエリのパフォーマンスも、アプリケーションの設計の際に考慮すべき大きな制約です。

グラフデータベースを使用すると、このような状況が大幅に簡素化されます。スキーマが不要なため、豊富なグラフクエリレイヤー (Gremlin または SPARQL) とグラフクエリ向けに最適化されたインデックスにより、柔軟性とパフォーマンスが向上します。Amazon Neptune グラフデータベースは、保存時の暗号化、安全な認証レイヤー、デフォルトのバックアップ、マルチ AZ サポート、リードレプリカのサポートなどのエンタープライズ向け機能も提供します。

AWS DMS を使用すると、高度に接続されたグラフをモデル化するリレーショナルデータを、サポートされている任意の SQL データベースの DMS ソースエンドポイントから Neptune ターゲットエンドポイントに移行できます。

詳細については、次を参照してください。

トピック

ターゲットとしての Amazon Neptune への移行の概要

Neptune ターゲットへの移行を開始する前に、AWS アカウントに次のリソースを作成します。

- ターゲットエンドポイントの Neptune クラスター
- ソースエンドポイント用の AWS DMS がサポートする SQL リレーショナルデータベース
- ターゲットエンドポイントの Amazon S3 バケット。この S3 バケットは Neptune クラスターと同じ AWS リージョンに作成します。AWS DMS は、この S3 バケットを、Neptuneデータベースに一括ロードするターゲットデータの中間ファイル ストレージとして使用します。S3 バケットの作成の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットを作成する」を参照。
- Neptune クラスターと同じ VPC 内の S3 の 仮想プライベートクラウド (VPC) エンドポイント
- IAM ポリシーを持つ AWS Identity and Access Management (IAM) ロール。このポリシーでは、ターゲットエンドポイントの S3 バケットに対する、`GetObject`、`PutObject`、`DeleteObject`、`ListObject` アクセス許可を指定する必要があります。このロールは、ターゲットS3 バケットと Neptuneデータベースの両方への IAM アクセスを持つ AWS DMS と Neptune の両方で引き受けられます。詳細については、「Amazon Neptune にターゲットとしてアクセスするための IAM サービスロールの作成」を参照してください。

上記のリソースを取得した後、Neptune ターゲットへの移行のセットアップと開始は、コンソールまたは DMS API を使用したフルロード移行と同様です。ただし、Neptune ターゲットへの移行にはいくつか独自のステップが必要です。

AWS DMS リレーショナルデータベースを Neptune に移行するには

1. 「レプリケーション インスタンスの作成」の説明に従って、レプリケーションインスタンスを作成します。
2. AWS DMS でサポートされるソースエンドポイントの SQL リレーショナルデータベースを作成して、テストします。
3. Neptune データベースのターゲットエンドポイントを作成してテストします。

   ターゲットエンドポイントを Neptuneデータベースに接続するには、 Neptune クラスターエンドポイントまたは Neptune ライターインスタンスエンドポイントのサーバー名を指定します。また、Neptune データベースに一括ロードする際の中間ファイルを保存するために、AWS DMS の S3 バケットフォルダも指定します。

   移行中、AWS DMS は、指定した最大ファイルサイズに達するまで、移行するすべてのターゲットデータをこの S3 バケットフォルダに保存します。このファイルストレージが最大サイズに達すると、AWS DMS は保存した S3 データをターゲットデータベースに一括ロードします。DMS はフォルダを消去して、後でターゲットデータベースにロードするために、追加のターゲットデータの保存を有効にします。上記の設定の指定の詳細については、「ターゲットとしての Amazon Neptune のエンドポイント設定の指定」を参照してください。
4. ステップ 1 ~ 3 で作成したリソースを使用してフルロードレプリケーションタスクを作成し、次を実行します。

   a. 通常のタスクテーブルマッピングを使用して、適切な選択ルールと変換ルールでリレーショナルデータベースから移行する特定のソーススキーマ、テーブルおよびビューを指定します。詳細については、「テーブルマッピングを使用して、タスクの設定を指定する」を参照してください。

   b. 次のいずれかを選択してターゲットマッピングを指定し、ソーステーブルとビューから Neptune ターゲットデータベースグラフへのマッピングルールを指定します。

      - Gremlin JSON – Gremlin JSON を使用して Neptune データベースをロードする方法については、「Amazon Neptune ユーザーガイド」の「Gremlin load data format」を参照してください。
      - SPARQL RDB からリソース記述フレームワークへのマッピング言語 (R2RML) – SPARQL R2RML の使用方法については、W3C 仕様「R2RML: RDB to RDF mapping language」を参照してください。

   c. 次のいずれかを実行します。

- AWS DMS コンソールを使用して、[データベース移行タスクの作成] ページで [グラフマッピングルール] を使用してグラフマッピングオプションを指定します。
- AWS DMS API を使用して、CreateReplicationTask API コールの TaskData リクエストパラメータを使用してこれらのオプションを指定します。

Gremlin JSON と SPARQL R2RML を使用してグラフマッピングルールを指定する詳細と例については、「Amazon Neptune の Gremlin と R2RML をターゲットとして使用するグラフマッピングルールの指定」を参照してください。

5. 移行タスクのレプリケーションを開始します。

ターゲットとしての Amazon Neptune のエンドポイント設定の指定

ターゲットエンドポイントを作成または変更するには、コンソール、CreateEndpoint、または ModifyEndpoint API オペレーションを使用します。

AWS DMS コンソールの Neptune ターゲットでは、[エンドポイントの作成] または [エンドポイントの変更] コンソールページの [エンドポイント固有の設定] を指定します。CreateEndpoint と ModifyEndpoint では、NeptuneSettings オプションのリクエストパラメータを指定します。CLI を使用してこれを実行する方法の例は次のとおりです。

```
dms create-endpoint --endpoint-identifier my-neptune-target-endpoint
--endpoint-type target --engine-name neptune
--server-name my-neptune-db.cluster-cspckvklbvgf.us-east-1.neptune.amazonaws.com
--port 8192
--neptune-settings
     '{"ServiceAccessRoleArn":"arn:aws:iam::123456789012:role/myNeptuneRole",
       "S3BucketName":"my-bucket",
       "S3BucketFolder":"my-bucket-folder",
       "ErrorRetryDuration":57,
       "MaxFileSize":100,
       "MaxRetryCount": 10,
       "IAMAuthEnabled":false}‘
```

ここでは、CLI --server-name オプションで、Neptune クラスターライターエンドポイントのサーバー名を指定します。または、Neptune ライターインスタンスエンドポイントのサーバー名を指定することもできます。

--neptune-settings オプションリクエストのパラメータは次のとおりです。

- `ServiceAccessRoleArn` – (必須) Neptune ターゲットエンドポイント向けに作成したサービスロールの Amazon リソースネーム (ARN)。詳細については、「Amazon Neptune にターゲットとしてアクセスするための IAM サービスロールの作成」を参照してください。
- `S3BucketName` – (必須) Neptune ターゲットデータベースに一括ロードする前に、移行グラフデータを DMS が .csv ファイルに一時的に保存できる S3 バケット名。DMS は、これらの .csv ファイルに保存する前に SQL ソースデータをグラフデータにマップします。
- `S3BucketFolder` – (必須) DMS が `S3BucketName` で指定された S3 バケットに移行されたグラフデータを保存するフォルダのパス
- `ErrorRetryDuration` – エラーが発生する前に、移行されたグラフデータの Neptune ターゲットデータベースへの一括ロードを再試行するために DMS が待機するミリ秒数。デフォルトは 250 です。
- `MaxFileSize` – (オプション) DMS が Neptune ターゲットデータベースにデータを一括ロードする前に、.csv ファイルに保存される移行されたグラフデータの KB 単位の最大サイズ。デフォルトは 1,048,576 KB (1 GB) です。正常に完了すると、DMS はバケットを消去して、移行したグラフデータの次のバッチを保存する準備を整えます。
- `MaxRetryCount` – (オプション) エラーが発生する前に、DMS が移行されたグラフデータの Neptune ターゲットデータベースへの一括ロードを再試行する回数。デフォルトは 5 です。
- `IAMAuthEnabled` – (オプション) このエンドポイントに対して IAM 認証を有効にする場合は、このパラメータを `true` に設定して、`ServiceAccessRoleArn` で指定されたサービスロールに適切な IAM ポリシードキュメントをアタッチします。デフォルトは `false` です。

Amazon Neptune にターゲットとしてアクセスするための IAM サービスロールの作成

Neptune にターゲットとしてアクセスするには、IAM を使用してサービスロールを作成します。Neptune エンドポイントの設定に応じて、次の IAM ポリシーと信頼ドキュメントの一部またはすべてをこのロールにアタッチします。Neptune エンドポイントを作成する際は、このサービスロールの ARN を指定します。これにより AWS DMS と Amazon Neptune は、Neptune と関連付けられた Amazon S3 バケットの両方にアクセスするアクセス許可を引き受けることができます。

Neptune エンドポイント設定で、`NeptuneSettings` の `IAMAuthEnabled` パラメータを `true` に設定した場合は、サービスロールに次のような IAM ポリシーをアタッチします。`IAMAuthEnabled` を `false` に設定すると、このポリシーを無視できます。

```
// Policy to access Neptune
```

```
    {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "VisualEditor0",
                "Effect": "Allow",
                "Action": "neptune-db:*",
                "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
CLG7H7FHK54AZGHEH6MNS55JKM/*"
            }
        ]
    }
```

上記の IAM ポリシーは、Resource で指定された Neptune ターゲットクラスターへのフルアクセスを許可します。

サービスロールに次のような IAM ポリシーをアタッチします。このポリシーにより、DMS は、Neptune ターゲットデータベースへの一括ロードのために作成した S3 バケットに移行されたグラフデータを一時的に保存できます。

```
//Policy to access S3 bucket

{
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "ListObjectsInBucket0",
   "Effect": "Allow",
   "Action": "s3:ListBucket",
   "Resource": [
    "arn:aws:s3:::my-bucket"
   ]
  },
  {
   "Sid": "AllObjectActions",
   "Effect": "Allow",
   "Action": ["s3:GetObject",
    "s3:PutObject",
    "s3:DeleteObject"
   ],

   "Resource": [
    "arn:aws:s3:::my-bucket/"
   ]
```

```
  },
  {
   "Sid": "ListObjectsInBucket1",
   "Effect": "Allow",
   "Action": "s3:ListBucket",
   "Resource": [
    "arn:aws:s3:::my-bucket",
    "arn:aws:s3:::my-bucket/"
   ]
  }
 ]
}
```

上記の IAM ポリシーでは、Neptune ターゲットのために作成された S3 バケット (arn:aws:s3:::my-bucket) の内容をアカウントからクエリできます。また、アカウントはすべてのバケットファイルとフォルダ (arn:aws:s3:::my-bucket/) の内容を完全に操作できるようになります。

信頼関係を編集して、次の IAM ロールをサービスロールにアタッチし、AWS DMS と Amazon Neptune データベースサービスの両方がロールを引き受けることを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "dms.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Sid": "neptune",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Neptune ターゲットエンドポイントにこのサービスロールを指定する方法については、「ターゲットとしての Amazon Neptune のエンドポイント設定の指定」を参照してください。

Amazon Neptune の Gremlin と R2RML をターゲットとして使用するグラフマッピングルールの指定

作成するグラフマッピングルールは、SQL リレーショナルデータベースのソースから抽出されたデータを Neptune データベースクラスターのターゲットにロードする方法を指定します。このようなマッピングルールの形式は、ルールが Apache TinkerPop や Gremlinを使用してプロパティグラフデータをロードするためか、 R2RML を使用してリソース記述フレームワーク (RDF) データをロードするためかにより異なります。形式に関する情報と詳細情報は、次に記載されています。

このようなマッピングルールは、コンソールまたは DMS API を使用して移行タスクを作成する際に指定できます。

コンソールを使用して、[データベース移行タスクの作成] ページで [グラフマッピングルール] を使用してグラフマッピングオプションを指定します。[グラフマッピングルール] では、提供されているエディタを使用して直接マッピングルールを入力したり編集したりできます。適切なグラフマッピング形式のマッピングルールを含むファイルを参照することもできます。

API を使用する場合は、TaskData API コールの CreateReplicationTask リクエストパラメータを使用してこのようなオプションを指定します。TaskData を適切なグラフマッピング形式のマッピングルールを含むファイルのパスに設定します。

Gremlin を使用してプロパティグラフデータを生成するためのグラフマッピングルール

Gremlin を使用してプロパティグラフデータを生成して、ソースデータから生成される各グラフエンティティのマッピングルールを含む JSON オブジェクトを指定します。この JSON の形式は、Amazon Neptune の一括ロード向けに定義されています。次のテンプレートは、このオブジェクトの各ルールがどのようなものになるかを示しています。

```
{
    "rules": [
        {
            "rule_id": "(an identifier for this rule)",
            "rule_name": "(a name for this rule)",
            "table_name": "(the name of the table or view being loaded)",
            "vertex_definitions": [
                {
                    "vertex_id_template": "{col1}",
```

```
                    "vertex_label": "(the vertex to create)",
                    "vertex_definition_id": "(an identifier for this vertex)",
                    "vertex_properties": [
                        {
                            "property_name": "(name of the property)",
                            "property_value_template": "{col2} or text",
                            "property_value_type": "(data type of the property)"
                        }
                    ]
                }
            ]
        },
        {
            "rule_id": "(an identifier for this rule)",
            "rule_name": "(a name for this rule)",
            "table_name": "(the name of the table or view being loaded)",
            "edge_definitions": [
                {
                    "from_vertex": {
                        "vertex_id_template": "{col1}",
                        "vertex_definition_id": "(an identifier for the vertex
referenced above)"
                    },
                    "to_vertex": {
                        "vertex_id_template": "{col3}",
                        "vertex_definition_id": "(an identifier for the vertex
referenced above)"
                    },
                    "edge_id_template": {
                        "label": "(the edge label to add)",
                        "template": "{col1}_{col3}"
                    },
                    "edge_properties":[
                        {
                            "property_name": "(the property to add)",
                            "property_value_template": "{col4} or text",
                            "property_value_type": "(data type like String, int,
double)"
                        }
                    ]
                }
            ]
        }
    ]
```

```
}
```

頂点ラベルがある場合、頂点がここで作成されていることを意味します。頂点ラベルがない場合は、頂点が別のソースによって作成され、この定義では頂点プロパティのみが追加されることを意味します。必要な数の頂点定義とエッジ定義を指定して、リレーショナルデータベースソース全体のマッピングを指定します。

次の employee テーブルのルール例で説明します。

```
{
    "rules": [
        {
            "rule_id": "1",
            "rule_name": "vertex_mapping_rule_from_nodes",
            "table_name": "nodes",
            "vertex_definitions": [
                {
                    "vertex_id_template": "{emp_id}",
                    "vertex_label": "employee",
                    "vertex_definition_id": "1",
                    "vertex_properties": [
                        {
                            "property_name": "name",
                            "property_value_template": "{emp_name}",
                            "property_value_type": "String"
                        }
                    ]
                }
            ]
        },
        {
            "rule_id": "2",
            "rule_name": "edge_mapping_rule_from_emp",
            "table_name": "nodes",
            "edge_definitions": [
                {
                    "from_vertex": {
                        "vertex_id_template": "{emp_id}",
                        "vertex_definition_id": "1"
                    },
                    "to_vertex": {
```

```
                    "vertex_id_template": "{mgr_id}",
                    "vertex_definition_id": "1"
                },
                "edge_id_template": {
                    "label": "reportsTo",
                    "template": "{emp_id}_{mgr_id}"
                },
                "edge_properties":[
                    {
                        "property_name": "team",
                        "property_value_template": "{team}",
                        "property_value_type": "String"
                    }
                ]
            }
        ]
    }
]
}
```

ここで、頂点とエッジの定義は、従業員 ID (EmpID) を持つ employee ノードとマネージャー ID (managerId) を持つ employee ノードの上下関係をマップします。

Gremlin JSON を使用したグラフマッピングルール作成の詳細については、「Amazon Neptune ユーザーガイド」の「Gremlin load data format」を参照してください。

RDF/SPARQL データを生成するためのグラフマッピングルール

SPARQL を使用してクエリする RDF データをロードする場合は、R2RML でグラフマッピングルールを記述します。R2RML は、リレーショナルデータを RDF にマップするための標準の W3C 言語です。R2RML ファイルで、トリプルマップ (以下の <#TriplesMap1> など) は、論理テーブルの各行を 0 個以上の RDF トリプルに変換するためのルールを指定します。サブジェクトマップ (次の rr:subjectMap のいずれか) は、トリプルマップによって生成される RDF トリプルのサブジェクトを生成するためのルールを指定します。述語オブジェクトマップ (次の rr:predicateObjectMap のいずれか) は、論理テーブルの論理テーブル行ごとに単一または複数の述語とオブジェクトのペアを作成する関数です。

nodes テーブルのシンプルな例を次に示します。

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
```

```
@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "nodes" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{id}";
        rr:class ex:Employee;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "label" ];
    ]
```

上記の例では、マッピングは従業員のテーブルからマップされたグラフノードを定義します。

Student テーブルの別のシンプルな例を次に示します。

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

<#TriplesMap2>
    rr:logicalTable [ rr:tableName "Student" ];
    rr:subjectMap   [ rr:template "http://example.com/{ID}{Name}";
                      rr:class foaf:Person ];
    rr:predicateObjectMap [
        rr:predicate ex:id ;
        rr:objectMap  [ rr:column "ID";
                        rr:datatype xsd:integer ]
    ];
    rr:predicateObjectMap [
        rr:predicate foaf:name ;
        rr:objectMap  [ rr:column "Name" ]
    ].
```

上記の例では、マッピングは、Student テーブル内の人物間の友人関係をマップするグラフノードを定義します。

SPARQL R2RML を使用したグラフマッピングルール作成の詳細については、W3C 仕様の「R2RML: RDB to RDF mapping language」を参照してください。

ターゲットの Amazon Neptune に移行する Gremlin と R2RML のデータ型

AWS DMS は、2 つの方法のいずれかで、SQL ソースエンドポイントから Neptune ターゲットへのデータ型マッピングを実行します。どちらの方法を使用するかは、Neptune データベースのロードに使用するグラフマッピング形式によって異なります。

- Apache TinkerPop Gremlin。移行データの JSON 表現を使用します。
- W3C の SPARQL。移行データの R2RML 表現を使用します。

上記 2 つのグラフマッピング形式の詳細については、「Amazon Neptune の Gremlin と R2RML をターゲットとして使用するグラフマッピングルールの指定」を参照してください。

各形式のデータ型マッピングを次に説明します。

SQL ソースから Gremlin ターゲットへのデータ型マッピング

次の表は、SQL ソースから Gremlin 形式のターゲットへのデータ型のマッピングを説明しています。

AWS DMS は、リストされていない SQL ソースデータ型を Gremlin `String` にマップします。

ソースの SQL のデータ型	ターゲットの Gremlin のデータ型
NUMERIC (とバリアント)	Double
DECIMAL	
TINYINT	Byte
SMALLINT	Short
INT, INTEGER	Int
BIGINT	Long
FLOAT	Float
DOUBLE PRECISION	
REAL	Double

ソースの SQL のデータ型	ターゲットの Gremlin のデータ型
BIT	Boolean
BOOLEAN	
DATE	Date
TIME	
TIMESTAMP	
CHARACTER (とバリアント)	String

Neptune をロードするための Gremlinデータ型の詳細については、「Neptune ユーザー ガイド」の「Gremlinデータ型」を参照してください。

SQL ソースからターゲットの R2RML (RDF) へのデータ型マッピング

次の表は、SQL ソースから R2RML 形式のターゲットへのデータ型のマッピングを説明しています。

RDF リテラルを除き、リストされているすべての RDFデータ型は大文字と小文字が区別されます。AWS DMS は、リストされていない SQL ソースデータ型を RDF リテラルにマップします。

RDF リテラルは、さまざまなリテラルの語彙形式とデータ型の 1 つです。詳細については、W3C 仕様の「Resource Description Framework (RDF): Concepts and Abstract Syntax」の「RDF literals」を参照してください。

ソースの SQL のデータ型	ターゲットの R2RML (RDF) のデータ型
BINARY (とバリアント)	xsd:hexBinary
NUMERIC (とバリアント)	xsd:decimal
DECIMAL	
TINYINT	xsd:integer
SMALLINT	

ソースの SQL のデータ型	ターゲットの R2RML (RDF) のデータ型
INT, INTEGER	
BIGINT	
FLOAT	xsd:double
DOUBLE PRECISION	
REAL	
BIT	xsd:boolean
BOOLEAN	
DATE	xsd:date
TIME	xsd:time
TIMESTAMP	xsd:dateTime
CHARACTER (とバリアント)	RDF リテラル

Neptune をロードするための RDF データ型と SQL ソースデータ型へのマッピングの詳細については、W3C 仕様の「R2RML: RDB to RDF Mapping Language」の「Datatype conversions」を参照してください。

Amazon Neptune をターゲットとして使用する場合の制限

Neptune をターゲットとして使用する場合、次の制限が適用されます。

- 現時点では、AWS DMS の Neptune ターゲットへの移行でサポートされているのはフルロードのみです。Neptune ターゲットへの変更データキャプチャ (CDC) の移行はサポートされていません。
- 次の例のとおり、移行タスクを開始する前に、ターゲットの Neptune データベースからすべてのデータが手動でクリアされていることを確認します。

  グラフ内のすべてのデータ (頂点とエッジ) をドロップするには、次の Gremlin コマンドを実行します。

```
gremlin> g.V().drop().iterate()
```

`'customer'` というラベルが付いている頂点をドロップするには、次の Gremlin コマンドを実行します。

```
gremlin> g.V().hasLabel('customer').drop()
```

> **Note**
>
> 大規模なデータセットをドロップするには時間がかかる場合があります。`limit(1000)` などの制限を使用して、`drop()` を反復処理することもできます。

`'rated'` というラベルのエッジをドロップするには、次の Gremlin コマンドを実行します。

```
gremlin> g.E().hasLabel('rated').drop()
```

> **Note**
>
> 大規模なデータセットをドロップするには時間がかかる場合があります。`limit(1000)` などの制限を使用して`drop()` を反復処理することもできます。

- DMS API オペレーションの `DescribeTableStatistics` は、Neptune グラフデータ構造の性質上、特定のテーブル関して不正確な結果を返す場合があります。

  AWS DMS は移行中に、各ソーステーブルをスキャンし、グラフマッピングを使用してソースデータを Neptune グラフに変換します。変換したデータは、まずターゲットエンドポイントに指定された S3 バケットフォルダに保存されます。ソースのスキャンと、この中間の S3 データの生成が正常に完了すると、`DescribeTableStatistics` はデータが Neptune ターゲットデータベースに正常にロードされたと見なします。ただし、これは常に実際に正常に完了したとは限りません。特定のテーブルにデータが適切にロードされたことを確認するには、そのテーブルの移行の両端で `count()` の戻り値を比較します。

  次の例では、AWS DMS はソースデータベースから `customer` テーブルをロードしています。これは、ターゲットの Neptune データベースでは、`'customer'` というラベルが割り当てられています。このラベルがターゲットデータベースに書き込まれていることを確認できます。これ

を実行するには、ソースデータベースからの使用可能な `customer` 行の数と、タスクの完了後に Neptune ターゲットデータベースにロードされた `'customer'` ラベル付きの行の数を比較します。

SQL を使用してソースデータベースから使用可能な customer 行の数を取得するには、次のコマンドを実行します。

```
select count(*) from customer;
```

Gremlin を使用してターゲットデータベースグラフにロードされた `'customer'` ラベル付きの行の数を取得するには、次のコマンドを実行します。

```
gremlin> g.V().hasLabel('customer').count()
```

- 現時点では、単一のテーブルのロードに失敗すると、タスク全体が失敗します。リレーショナルデータベース ターゲットとは異なり、Neptune のデータは高度に接続されているため、多くの場合タスクを再開できません。このようなタイプのデータロードの失敗が原因でタスクを正常に再開できない場合は、ロードに失敗したテーブルをロードする新しいタスクを作成します。この新しいタスクを実行する前に、部分的にロードされたテーブルを Neptune ターゲットから手動でクリアします。

  > **Note**
  >
  > 失敗が回復可能な場合 (ネットワーク転送エラーなど) は、Neptune ターゲットへの移行に失敗したタスクを再開できます。

- AWS DMS は、R2RML のほとんどの標準をサポートしています。ただし、AWS DMS 逆数式、結合、ビューなどの特定の R2RML 標準はサポートしていません。R2RML ビューについては、ソースデータベースに対応するカスタム SQL ビューを作成することが回避策です。移行タスクでテーブルマッピングを使用して、ビューを入力として選択します。次に、ビューをテーブルにマップすると、R2RML が使用して、グラフデータが生成されます。
- サポートされていない SQL データ型を使用してソースデータを移行すると、ターゲットデータの精度の低下につながる可能性があります。詳細については、「ターゲットの Amazon Neptune に移行する Gremlin と R2RML のデータ型」を参照してください。
- AWS DMS は、LOB データの Neptune ターゲットへの移行をサポートしていません。

AWS Database Migration Service のターゲットとしての Redis の使用

Redis は、データベース、キャッシュ、メッセージブローカーとして使用されるオープンソースのメモリ内データ構造ストアです。メモリ内でデータを管理することにより、読み取りや書き込みオペレーションにかかる時間が 1 ミリ秒未満となり、毎秒数億回もの操作の実行を実現します。Redis はインメモリデータストアとして、ミリ秒未満の応答時間を必要とする最も要求の厳しいアプリケーションをサポートしています。

AWS DMS を使用すると、サポートされているソースデータベースからターゲットの Redis データストアに最小限のダウンタイムでデータを移行できます。Redis の詳細については、「Redis ドキュメント」を参照してください。

オンプレミスの Redis に加えて、AWS Database Migration Service は次をサポートします。

- ターゲットデータストアとしての Amazon ElastiCache for Redis。ElastiCache for Redis は Redis クライアントと連携して、オープンな Redis データ形式を使用してデータを保存します。
- ターゲットデータストアとしての Amazon MemoryDB for Redis。MemoryDB は Redis と互換性があり、現在使用されているすべての Redis データ構造、API、コマンドを使用してアプリケーションを構築できます。

AWS DMS のターゲットとして Redis を使用する方法の詳細については、次のセクションを参照してください。

トピック

- Redis クラスターを AWS DMS のターゲットとして使用するための前提条件
- Redis を AWS Database Migration Service のターゲットとして使用する場合の制限
- リレーショナルデータベースまたは非リレーショナルデータベースからターゲットの Redis へのデータ移行
- ターゲットとしての Redis エンドポイント設定の指定

Redis クラスターを AWS DMS のターゲットとして使用するための前提条件

DMS は、スタンドアロン構成のオンプレミスの Redis ターゲットをサポートします。また、データが複数のノード間で自動的にシャード化される Redis クラスターとしてもサポートします。シャード化とは、データを複数のサーバーまたはノードにまたがるシャードと呼ばれる小さなチャンクに分割するプロセスです。実際、シャードはデータセット全体のサブセットを含むデータパーティションであり、ワークロード全体のスライスとして機能します。

Redis はキーバリュー型の NoSQL データストアです。ソースがリレーショナルデータベースの場合に使用する Redis キーの命名規則は、schema-name.table-name.primary-key となります。Redis では、キーと値に特殊文字の % は使用できません。使用すると、DMS は該当するレコードをスキップします。

> **Note**
>
> ElastiCache for Redis をターゲットとして使用している場合、DMS がサポートするのは、クラスターモードが有効になっている設定のみです。ElastiCache for Redis バージョン 6.x 以降を使用してクラスターモードが有効になっているターゲットデータストアを作成する方法の詳細については、「Amazon ElastiCache for Redis ユーザーガイド」の「使用開始」を参照してください。

データベースの移行を開始する前に、次の基準を使用して Redis クラスターを起動します。

- クラスターには単一または複数のシャードがあります。
- ElastiCache for Redis ターゲットを使用している場合は、クラスターが IAM ロールベースのアクセスコントロールを使用していないことを確認してください。この代わりに、Redis Auth を使用してユーザーを認証します。
- マルチ AZ (アベイラビリティーゾーン) を有効にします。
- データベースから移行するデータを処理するのに十分なメモリがクラスターにあることを確認します。
- 初めて移行タスクをスタートする前に、ターゲットの Redis クラスターになにもデータがないことを確認します。

クラスターの設定を行う前に、データ移行のセキュリティ要件を決定する必要があります。DMS は、暗号化の設定を問わず、ターゲットのレプリケーショングループへの移行をサポートします。ただし、暗号化を有効または無効にできるのは、クラスターの設定時のみです。

Redis を AWS Database Migration Service のターゲットとして使用する場合の制限

Redis をターゲットとして使用する場合、次の制限が適用されます。

- Redis はキーバリューの no-sql データストアです。ソースがリレーショナルデータベースの場合に使用する Redis キーの命名規則は、`schema-name.table-name.primary-key` となります。

- Redis では、キーバリューに特殊文字の % は使用できません。使用すると、DMS は該当するレコードをスキップします。
- DMS は特殊文字を含む行を移行しません。
- DMS は、フィールド名に特殊文字が含まれるフィールドを移行しません。
- 完全 LOB モードはサポートされていません。
- ElastiCache for Redis をターゲットとして使用する場合、Private Certificate Authority (CA) はサポートされません。

リレーショナルデータベースまたは非リレーショナルデータベースからターゲットの Redis へのデータ移行

任意のソース SQL または NoSQL データストアからターゲットの Redis には直接データを移行できます。Redis ターゲットへの移行の設定とスタートは、DMS コンソールまたは API を使用したフルロードと変更データキャプチャの移行と同様です。Redis ターゲットへのデータベースの移行を実行するには、次を実行します。

- 移行のすべてのプロセスを実行するレプリケーションインスタンスを作成します。詳細については、「レプリケーション インスタンスの作成」を参照してください。
- ソースエンドポイントを指定します。詳細については、「ソースおよびターゲットエンドポイントの作成」を参照してください。
- クラスターの DNS 名とポート番号を取得します。
- SSL 接続の確認のために使用できる証明書バンドルをダウンロードします。
- 次の説明のとおり、ターゲットエンドポイントを指定します。
- 使用するテーブルとレプリケーションプロセスを定義するタスクまたはタスクセットを作成して、レプリケーションを開始します。詳細については、「[Creating a task] (タスクの作成)」を参照してください。
- ソースデータベースからターゲットクラスターにデータを移行します。

データベースの移行は、次の 2 つの方法のいずれかで開始します。

1. AWS DMS コンソールをクリックして、コンソールで各ステップを実行します。
2. AWS Command Line Interface (AWS CLI) を使用することができます。CLI を AWS DMS と組み合わせて使用する方法については、「AWS CLI for AWS DMS」を参照してください。

クラスターの DNS 名とポート番号を取得するには

- 次の AWS CLI コマンドを使用して、replication-group-id にレプリケーショングループ名を提供します。

```
aws elasticache describe-replication-groups --replication-group-id myreplgroup
```

ここでの出力は、クラスター内のプライマリノードの Address 属性に DNS 名、Port 属性にポート番号が表示されます。

```
 ...
"ReadEndpoint": {
"Port": 6379,
"Address": "myreplgroup-
111.1abc1d.1111.uuu1.cache.example.com"
}
...
```

MemoryDB for Redis をターゲットとして使用する場合は、次の AWS CLI コマンドを使用して Redis クラスターにエンドポイントアドレスを提供します。

```
aws memorydb describe-clusters --clusterid clusterid
```

SSL 接続の検証に使用する証明書バンドルのダウンロード

- コマンドラインで、次の wget コマンドを入力します。Wget は、インターネットからファイルをダウンロードする際に使用できる無料の GNU コマンドラインユーティリティツールです。

```
wget https://s3.aws-api-domain/rds-downloads/rds-combined-ca-bundle.pem
```

ここで、*aws-api-domain* は、指定された S3 バケットと rds-combined-ca-bundle.pem ファイルにアクセスするために必要な AWS リージョンの Amazon S3 ドメインとなります。

AWS DMS コンソールを使用してターゲットエンドポイントを作成するには

このエンドポイントは、既に実行中の Redis ターゲットのためのものです。

- コンソールで、ナビゲーションペインから [エンドポイント] を選択して、[エンドポイントの作成] をクリックします。次の表は、設定について説明しています。

使用するオプション	実行項目
エンドポイントタイプ	[ターゲット] エンドポイントタイプを選択する。
エンドポイント識別子	エンドポイントの名前を入力する。名前にはエンドポイントタイプを含める。例えば、 **my-redis-target** と入力する。
ターゲットエンジン	このエンドポイントに接続するデータベースエンジンタイプとして [Redis] を選択する。
クラスター名	Redis クラスターの DNS 名を入力する。
ポート	Redis クラスターのポート番号を入力する。
SSL セキュリティプロトコル	[プレーンテキスト] または [SSL 暗号化] のどちらかを選択する。 プレーンテキスト—このオプションの場合、エンドポイントとデータベース間のトラフィックに Transport Layer Security (TLS) 暗号化は提供されない。 SSL 暗号化—このオプションを選択した場合、SSL 認証機関 (CA) 証明書 ARN を入力してサーバーの証明書を確認し、暗号化された接続を確立する。 オンプレミスの Redis で DMS パブリック認証機関と Private Certificate Authority (CA) の両方をサポートする。ElastiCache for Redis の場合、DMS はパブリック CA のみをサポートする。

使用するオプション	実行項目
認証タイプ	Redis への接続中に実行する認証タイプを選択する。オプションには、[なし]、[認証ロール]、[認証トークン] がある。 認証ロールを選択した場合、[認証ユーザー名] と [認証パスワード] を入力する。 認証トークンを選択した場合に指定するのは、[認証パスワード] のみ。
レプリケーションインスタンス	[オプション]接続をテストする場合のみ、[レプリケーションの作成] ページで以前に入力したレプリケーションインスタンス名前を選択する。

エンドポイントのすべての情報を入力した後、AWS DMS は、データベース移行中に使用する Redis ターゲットエンドポイントを作成します。

移行タスクの作成とデータベース移行の開始については、「[Creating a task] (タスクの作成)」を参照してください。

ターゲットとしての Redis エンドポイント設定の指定

ターゲットエンドポイントを作成または変更するには、コンソール、`CreateEndpoint`、または `ModifyEndpoint` API オペレーションを使用します。

AWS DMS コンソールの Redis ターゲットでは、[エンドポイントの作成] または [エンドポイントの変更] コンソールページの [エンドポイント固有の設定] を指定します。

`CreateEndpoint` と `ModifyEndpoint` の API オペレーションでは、`RedisSettings` オプションのリクエストパラメータを指定します。次の例では、AWS CLI を使用して設定する方法を説明しています。

```
aws dms create-endpoint --endpoint-identifier my-redis-target
--endpoint-type target --engine-name redis --redis-settings
'{"ServerName":"sample-test-
sample.zz012zz.cluster.eee1.cache.bbbxxx.com","Port":6379,"AuthType":"auth-token",
 "SslSecurityProtocol":"ssl-encryption", "AuthPassword":"notanactualpassword"}'
```

```
{
    "Endpoint": {
        "EndpointIdentifier": "my-redis-target",
        "EndpointType": "TARGET",
        "EngineName": "redis",
        "EngineDisplayName": "Redis",
        "TransferFiles": false,
        "ReceiveTransferredFiles": false,
        "Status": "active",
        "KmsKeyId": "arn:aws:kms:us-east-1:999999999999:key/x-b188188x",
        "EndpointArn": "arn:aws:dms:us-
east-1:555555555555:endpoint:ABCDEFGHIJKLMONOPQRSTUVWXYZ",
        "SslMode": "none",
        "RedisSettings": {
            "ServerName": "sample-test-sample.zz012zz.cluster.eee1.cache.bbbxxx.com",
            "Port": 6379,
            "SslSecurityProtocol": "ssl-encryption",
            "AuthType": "auth-token"
        }
    }
}
```

`--redis-settings` パラメータ以降は次のとおりです。

- `ServerName`— (必須) `string` 型。同じ VPC に配置されたデータの移行先の Redis クラスターを指定する。
- `Port`— (必須) `number` 型。エンドポイントへのアクセスに使用されるポート値。
- `SslSecurityProtocol`— (オプション) 有効な値は `plaintext` と `ssl-encryption`。デフォルトは `ssl-encryption`。

  `plaintext` オプションは、エンドポイントとデータベース間のトラフィックに Transport Layer Security (TLS) 暗号化を提供しない。

  `ssl-encryption` を使用して、暗号化された接続を作成する。`ssl-encryption` ではサーバーの証明書を検証するための SSL 認証機関 (CA) ARN は必要ないが、必要に応じて `SslCaCertificateArn` 設定を使用して識別できる。認証機関 ARN が指定されていない場合、DMS は Amazon ルート CA を使用する。

  オンプレミスの Redis ターゲットを使用する場合、`SslCaCertificateArn` を使用してパブリック認証機関と Private Certificate Authority (CA) を DMS にインポートし、サーバー認証にその

ARN を指定する。ElastiCache for Redis をターゲットとして使用する場合、プライベート CA はサポートされない。

- `AuthType`–(必須) Redis に接続する際に実行する認証のタイプ。有効な値は、`none`、`auth-token`、`auth-role`。

  `auth-token` オプションには、「*`AuthPassword`*」を指定する必要がある。`auth-role` オプションには「*`AuthUserName`*」と「*`AuthPassword`*」を指定する必要がある。

AWS Database Migration Service のターゲットとしての Babelfish の使用

AWS Database Migration Service を使用して、ソースの Microsoft SQL Server データベースからターゲットの Babelfish にデータを移行できます。

Babelfish for Aurora PostgreSQL を使用すると、Amazon Aurora PostgreSQL 互換エディションの Microsoft SQL Server クライアントからのデータベース接続を受け入れる機能を使用できます。これにより、SQL Server 向けに構築されたアプリケーションは、従来の移行と比較してほとんどコードを変更することなく、データベースドライバーを変更せずに、Aurora PostgreSQL と直接連携できるようになります。

AWS DMS がターゲットとしてサポートする Babelfish のバージョンについては、「のターゲット AWS DMS」を参照してください。Aurora PostgreSQL 上の Babelfish の以前のバージョンの場合は、 Babelfish エンドポイントを使用する前にアップグレードする必要があります。

> **Note**
>
> データを Babelfish に移行するには、Aurora PostgreSQL ターゲットエンドポイントの使用をお勧めします。詳細については、「ターゲットとしての Babelfish for Aurora PostgreSQL の使用」を参照してください。

Babelfish をデータベースエンドポイントとして使用する方法の詳細については、「Amazon Aurora User Guide for Aurora」の「Babelfish for Aurora PostgreSQL」を参照してください。

Babelfishを AWS DMS のターゲットとして使用するための前提条件

AWS DMS で適切なータ型とテーブルメタデータを使用するには、データを移行する前にテーブルを作成する必要があります。移行を実行する前にターゲットにテーブルを作成しないと、AWS DMS は誤ったデータ型とアクセス権限を使用してテーブルを作成する可能性があります。例えば、AWS

DMS がタイムスタンプ列の代わりに binary (8) として作成すると、期待どおりのタイムスタンプと行のバージョン機能が提供されません。

移行前にテーブルを準備して作成するには

1. 一意の制約、プライマリキー、またはデフォルト制約を含む create table DDL ステートメントを実行します。

   外部キー制約、ビュー、ストアドプロシージャ、関数、トリガーなどのオブジェクトの DDL ステートメントは含めないでください。上記はソースデータベースを移行した後に適用できます。

2. テーブルのアイデンティティ列、計算列、または rowversion や timestamp などのデータ型を含む列を特定します。次に、移行タスクを実行する際の既知の問題に対処するために必要な変換ルールを作成します。詳細については、「変換ルールおよび変換アクション」を参照してください。

3. Babelfish がサポートしていないデータ型の列を特定します。次に、サポートされているデータ型を使用するようにターゲットテーブル内の影響を受ける列を変更するか、移行タスク中にこのような列を削除する変換ルールを作成します。詳細については、「変換ルールおよび変換アクション」を参照してください。

   次の表は、Babelfish でサポートされていないソースデータ型と、それに対して使用することが推奨されるターゲットでのデータ型を示しています。

ソースのデータ型	Babelfish の推奨データ型
HEIRARCHYID	NVARCHAR(250)
GEOMETRY	VARCHAR(最大)
GEOGRAPHY	VARCHAR(最大)

ソースの Aurora PostgreSQL Serverless V2 データベースの Aurora キャパシティユニット (ACU) レベルを設定するには

AWS DMS 移行タスクを実行する前に、最小 ACU 値を設定することで、タスクのパフォーマンスを向上できます。

- [Severless v2 キャパシティ設定] ウィンドウで、[最小 ACU] を **2**、または Aurora DB クラスターに適切なレベルに設定します。

詳細については、「Amazon Aurora ユーザーガイド」の「Aurora クラスターの Aurora Serverless v2 での容量範囲の選択」を参照してください。

AWS DMS 移行タスクを実行した後、ACU の最小値をソースの Aurora PostgreSQL Serverless V2 データベース向けに適切なレベルにリセットできます。

AWS Database Migration Service のターゲットとして Babelfish を使用する場合のセキュリティ要件

Babelfish のターゲットで AWS DMS を使用するためのセキュリティ要件は次のとおりです。

- データベースの作成に使用される管理者ユーザー名 (Admin ユーザー)
- PSQL ログインと、十分な SELECT、INSERT、UPDATE、DELETE、および REFERENCES アクセス権限を持つユーザー

Babelfish を AWS DMS のターゲットとして使用する場合のユーザー権限

> **Important**
>
> セキュリティ上の理由から、データ移行に使用するユーザーアカウントは、ターゲットとして使用する Babelfish データベースに登録されているユーザーである必要があります。

Babelfish ターゲットエンドポイントでは、AWS DMS 移行を実行するための最低限のユーザーのアクセス権限が必要です。

ログインと低特権の Transact-SQL (T-SQL) ユーザーを作成するには

1. サーバーに接続する際に使用するログイン名とパスワードを作成します。

```
CREATE LOGIN dms_user WITH PASSWORD = 'password';
GO
```

2. Babelfish クラスターの仮想データベースを作成します。

```
CREATE DATABASE my_database;
GO
```

3. ターゲットデータベースの T-SQL ユーザーを作成します。

```
USE my_database
GO
CREATE USER dms_user FOR LOGIN dms_user;
GO
```

4. Babelfish データベース内の各テーブルごとにテーブルに対するアクセス権限を付与します。

```
GRANT SELECT, DELETE, INSERT, REFERENCES, UPDATE ON [dbo].[Categories] TO dms_user;
```

Babelfish を AWS Database Migration Service のターゲットとして使用する場合の制限

Babelfish データベースを AWS DMS のターゲットとして使用する場合、次の制限が適用されます。

- テーブル作成モードについては、[何もしない] のみがサポートされています。
- ROWVERSION データ型には、移行タスク中にテーブルから列名を削除するテーブルマッピングルールが必要です。
- sql_variant データ型はサポートされていません。
- 完全 LOB モードはサポートされています。SQL Server をソースエンドポイントとして使用する場合、LOB をターゲットエンドポイントに移行するために SQL Server エンドポイント接続属性設定 `ForceFullLob=True` を設定する必要があります。
- レプリケーションタスクの設定には次の制限があります。

```
{
   "FullLoadSettings": {
      "TargetTablePrepMode": "DO_NOTHING",
      "CreatePkAfterFullLoad": false,
      }.

}
```

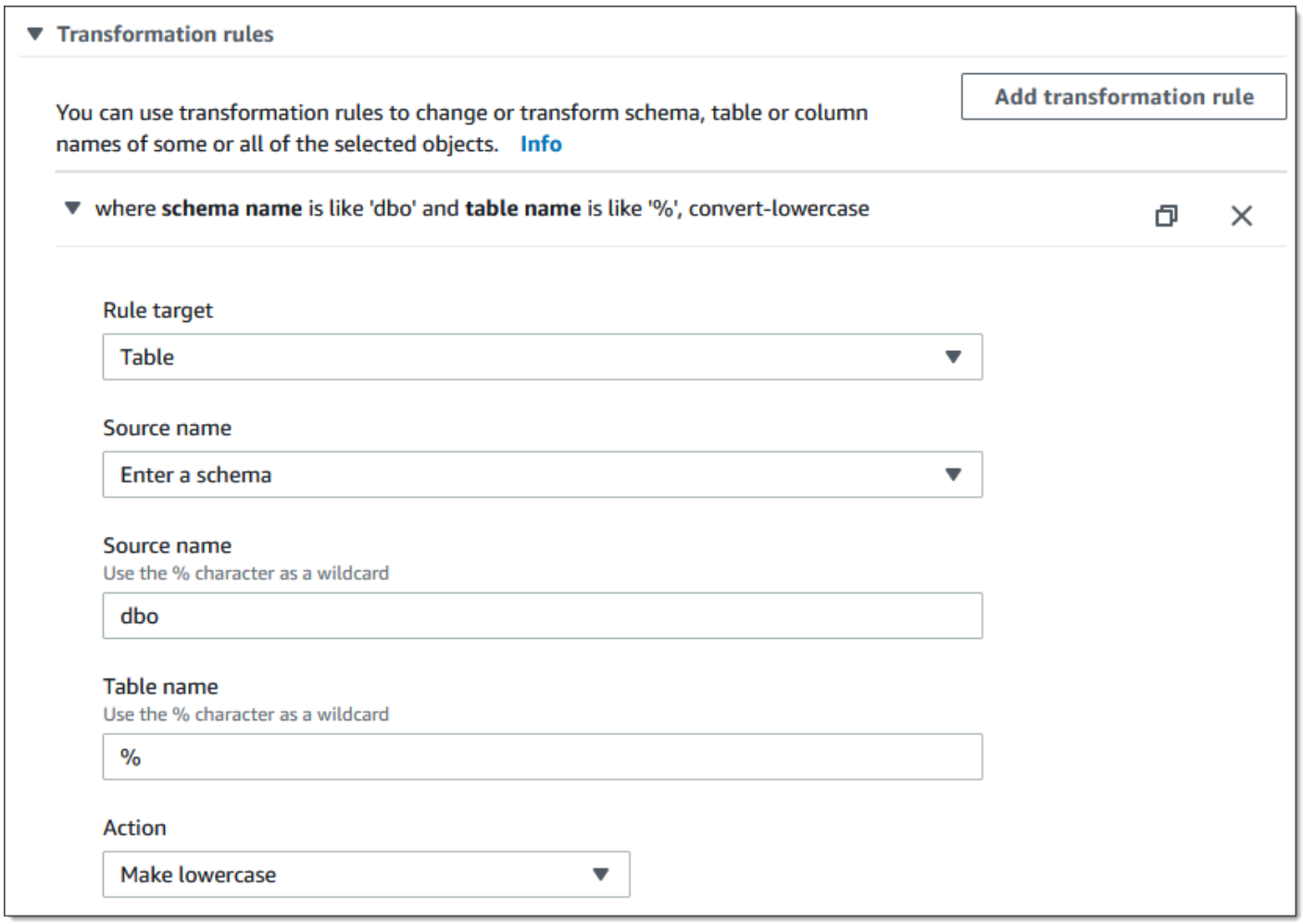
- Babelfish の TIME (7)、DATETIME2 (7)、DATETIMEOFFSET (7) のデータ型では、時刻の秒部分の精度値が 6 桁に制限されます。上記のデータ型を使用する際は、ターゲットテーブルの

精度値を 6 に設定することを検討します。Babelfish バージョン 2.2.0 以降では、TIME (7) と DATETIME2 (7) を使用すると、精度の 7 桁目は常にゼロになります。

- DO_NOTHING モードでは、DMS はテーブルが既に存在するかどうかを確認します。テーブルがターゲットスキーマに存在しない場合、DMS はソースのテーブル定義に基づいてテーブルを作成し、ユーザー定義のデータ型を基本データ型にマップします。
- Babelfish ターゲットへの AWS DMS 移行タスクは、 ROWVERSION または TIMESTAMP データ型を使用する列を持つテーブルをサポートしていません。転送プロセス中にテーブルから列名を削除するテーブルマッピングルールを使用できます。次の変換ルールの例では、ソースの `Actor` という名前のテーブルを変換して、ターゲットの `Actor` テーブル内の先頭文字が `col` であるすべての列を削除します。

```
{
  "rules": [{
  "rule-type": "selection",is
  "rule-id": "1",
  "rule-name": "1",
  "object-locator": {
   "schema-name": "test",
   "table-name": "%"
  },
  "rule-action": "include"
 }, {
  "rule-type": "transformation",
  "rule-id": "2",
  "rule-name": "2",
  "rule-action": "remove-column",
  "rule-target": "column",
  "object-locator": {
   "schema-name": "test",
   "table-name": "Actor",
   "column-name": "col%"
  }
 }]
 }
```

- アイデンティティ列または計算列を含むテーブルの場合、ターゲットテーブルで Categories などの大文字と小文字が混合した名前が使用されている場合、DMS タスクのためにテーブル名を小文字に変換する変換ルールアクションを作成する必要があります。次の例は、AWS DMS コンソールを使用して変換ルールアクション Make lowercase を作成する方法を説明しています。詳細については、「 変換ルールおよび変換アクション」を参照してください。



- Babelfish バージョン 2.2.0 以前のバージョンの場合、DMS では Babelfish ターゲットエンドポイントにレプリケートできる列の数が 20 列に制限されていました。Babelfish 2.2.0 では、この制限は 100 列に引き上げられています。ただし、Babelfish バージョン 2.4.0 以降では、レプリケートできる列の数がさらに増大しています。SQL Server データベースに対して次のコードサンプルを実行すると、長すぎるテーブルを特定できます。

```
USE myDB;
GO
DECLARE @Babelfish_version_string_limit INT = 8000; -- Use 380 for Babelfish versions
 before 2.2.0
WITH bfendpoint
AS (
SELECT
 [TABLE_SCHEMA]
      ,[TABLE_NAME]
   , COUNT( [COLUMN_NAME] ) AS NumberColumns
   , ( SUM( LEN( [COLUMN_NAME] ) + 3)
```

```
  + SUM( LEN( FORMAT(ORDINAL_POSITION, 'N0') ) + 3 )
     + LEN( TABLE_SCHEMA ) + 3
  + 12 -- INSERT INTO string
  + 12)  AS InsertIntoCommandLength -- values string
      , CASE WHEN ( SUM( LEN( [COLUMN_NAME] ) + 3)
  + SUM( LEN( FORMAT(ORDINAL_POSITION, 'N0') ) + 3 )
     + LEN( TABLE_SCHEMA ) + 3
  + 12 -- INSERT INTO string
  + 12)  -- values string
   >= @Babelfish_version_string_limit
   THEN 1
   ELSE 0
  END AS IsTooLong
FROM [INFORMATION_SCHEMA].[COLUMNS]
GROUP BY [TABLE_SCHEMA], [TABLE_NAME]
)
SELECT *
FROM bfendpoint
WHERE IsTooLong = 1
ORDER BY TABLE_SCHEMA, InsertIntoCommandLength DESC, TABLE_NAME
;
```

ターゲットの Babelfish のデータ型

次の表は、AWS DMS の使用時にサポートされるターゲットの Babelfish のデータ型と、AWS DMS データ型からのデフォルトのマッピングを説明しています。

AWS DMS のデータ型の詳細については、「AWS Database Migration Service のデータ型」を参照してください。

AWS DMS のデータ型	Babelfish のデータ型
BOOLEAN	TINYINT
BYTES	VARBINARY(長さ)
DATE	DATE
TIME	TIME
INT1	SMALLINT

AWS DMS のデータ型	Babelfish のデータ型
INT2	SMALLINT
INT4	INT
INT8	BIGINT
NUMERIC	NUMERIC(p,s)
REAL4	REAL
REAL8	FLOAT
STRING	列が日付列または時刻列の場合、次を実行する。 • SQL Server 2008 以降の場合、DATETIME2 を使用する。 • それ以前のバージョンでは、位取りが 3 以下の場合は DATETIME を使用する。その他すべての場合は、VARCHAR (37) を使用する。 列が日付または時刻列ではない場合、VARCHAR (長さ) を使用する。
UINT1	TINYINT
UINT2	SMALLINT
UINT4	INT
UINT8	BIGINT
WSTRING	NVARCHAR(長さ)

AWS DMS のデータ型	Babelfish のデータ型
BLOB	VARBINARY(最大) DMS でこのデータ型を使用するには、特定のタスク用に BLOB の使用を有効にする必要がある。DMS は、プライマリキーがあるテーブルでのみ BLOB データ型をサポートする。
CLOB	VARCHAR(最大) DMS でこのデータ型を使用するには、特定のタスクで CLOB の使用を有効にする必要がある。
NCLOB	NVARCHAR(最大) DMS でこのデータ型を使用するには、特定のタスクで NCLOB の使用を有効にする必要がある。CDC 中に、DMS はプライマリキーがあるテーブルでのみ NCLOB データ型をサポートする。

Amazon Timestream を AWS Database Migration Service のターゲットとして使用する

AWS Database Migration Service を使用して、ソースデータベースから Amazon Timestream ターゲットエンドポイントにデータを移行できます。フルロードおよび CDC データ移行がサポートされます。

Amazon Timestream は、大量のデータインジェスト用に構築された、高速でスケーラブルなサーバーレスの時系列データベースサービスです。時系列データは、一定の間隔で収集された一連のデータポイントで、時間の経過とともに変化するイベントの測定に使用します。IoT アプリケーション、DevOps アプリケーション、分析アプリケーションからメトリクスを収集、保存、分析するために使用されます。Timestream にデータを保存すると、データの傾向やパターンをほぼリアルタイムで視覚化して特定できます。Amazon Timestream の詳細については、「Amazon Timestream デベロッパーガイド」の「Amazon Timestream とは」を参照してください。

トピック

Amazon Timestream を AWS Database Migration Service のターゲットとして使用する場合の前提条件

Amazon Timestream を AWS DMS のターゲットとして設定する前に、IAM ロールを必ず作成してください。このロールにより、AWS DMS は Amazon Timestream に移行されるデータにアクセスできるようになります。Timestream への移行に使用するロールの最低限のアクセス許可は、次の IAM ポリシーに示されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDescribeEndpoints",
            "Effect": "Allow",
            "Action": [
                "timestream:DescribeEndpoints"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "timestream:ListTables",
                "timestream:DescribeDatabase"
            ],
            "Resource": "arn:aws:timestream:region:account_id:database/DATABASE_NAME"
        },
```

```
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "timestream:DeleteTable",
                "timestream:WriteRecords",
                "timestream:UpdateTable",
                "timestream:CreateTable"
            ],
            "Resource": "arn:aws:timestream:region:account_id:database/DATABASE_NAME/
table/TABLE_NAME"
        }
    ]
}
```

すべてのテーブルを移行する場合は、上の例の *TABLE_NAME* として * を使用します。

Timestream をターゲットとして使用する場合は、次の点に注意してください。

- タイムスタンプが 1 年を超える履歴データを取り込む場合は、AWS DMS を使用してデータをカンマ区切り値 (csv) 形式で Amazon S3 に書き込むことをお勧めします。次に、Timestream のバッチロードを使用してデータを Timestream に取り込みます。詳細については、「Amazon Timestream デベロッパーガイド」の「Timestream でのバッチロードの使用」を参照してください。https://docs.aws.amazon.com/timestream/latest/developerguide/what-is-timestream.html
- 1 年未満のデータをフルロードで移行する場合は、Timestream テーブルのメモリストア保持期間を最も古いタイムスタンプ以上に設定することをお勧めします。次に、移行が完了したら、テーブルのメモリストア保持期間を編集して希望の値に変更します。例えば、データの最も古いタイムスタンプが 2 か月前である場合、このデータを移行するには、次の操作を行います。
  - Timestream ターゲットテーブルのメモリストア保持期間を 2 か月に設定します。
  - AWS DMS を使用してデータ移行を開始します。
  - データ移行が完了したら、ターゲット Timestream テーブルの保持期間を希望の値に変更します。

  以下のページの情報を使用して、移行前にメモリストアのコストを見積もることをお勧めします。
  - Amazon Timestream の料金
  - AWS 料金見積りツール
- CDC データ移行では、取り込まれたデータがメモリストア保持期間の範囲内に収まるように、ターゲットテーブルのメモリストア保持期間を設定することをお勧めします。詳細については、

「Amazon Timestream デベロッパーガイド」の「書き込みのベストプラクティス」を参照してください。https://docs.aws.amazon.com/timestream/latest/developerguide/what-is-timestream.html

マルチスレッド全ロードタスク設定

データ転送を速くするために、AWS DMS は、これらのタスク設定を使用した Timestream ターゲットエンドポイントへのマルチスレッド全ロード移行タスクをサポートしています。

- `MaxFullLoadSubTasks` - このオプションを使用して、並列ロードするソーステーブルの最大数を指定します。DMS は、専用のサブタスクを使用して、対応する Amazon Timestream ターゲットテーブルに各テーブルをロードします。デフォルトは 8、最大値は 49 です。
- `ParallelLoadThreads` - このオプションを使用して、AWS DMS が各テーブルを Amazon Timestream ターゲットテーブルにロードするために使用するスレッド数を指定します。Timestream ターゲットの最大値は 32 です。この上限を増やすよう依頼できます。
- `ParallelLoadBufferSize` – このオプションを使用して、Amazon Timestream ターゲットにデータをロードするために並列ロードスレッドで使用する、バッファ内に保存するレコードの最大数を指定します。デフォルト値は 50 です。最大値は 1000 です。この設定は `ParallelLoadThreads` で使用します。`ParallelLoadBufferSize` は、複数のスレッドがある場合にのみ有効です。
- `ParallelLoadQueuesPerThread` - このオプションを使用して、各同時スレッドがキューからデータレコードを取り出し、ターゲットのバッチロードを生成するためにアクセスするキューの数を指定します。デフォルト は 1 です。ただし、さまざまなペイロードサイズの Amazon Timestream ターゲットの場合、有効な範囲は 1 スレッドあたり 5～512 キューです。

マルチスレッド CDC ロードタスクの設定

CDC のパフォーマンスを向上させるため、AWS DMS では次のタスク設定をサポートしています。

- `ParallelApplyThreads` – データレコードを Timestream ターゲットエンドポイントにプッシュするために CDC ロード中に AWS DMS で使用する同時スレッドの数を指定します。デフォルト値は 0、最大値は 32 です。
- `ParallelApplyBufferSize` – 同時スレッドが CDC ロード中に Timestream ターゲットエンドポイントにプッシュするために、各バッファキューに保存するレコードの最大数を指定します。デフォルト値は 100 で、最大値は 1,000 です。このオプションは、`ParallelApplyThreads` で複数のスレッドを指定する場合に使用します。

- `ParallelApplyQueuesPerThread` – 各スレッドがキューからデータレコードを取り出し、CDC 中に Timestream エンドポイントのバッチロードを生成するためにアクセスするキューの数を指定します。デフォルト値は 1、最大値は 512 です。

Timestream を AWS DMS のターゲットとして使用する場合のエンドポイント設定

エンドポイント設定を使用して、追加の接続属性を使用する場合と同じように、Timestream ターゲットデータベースを設定できます。ターゲットエンドポイントを作成する際に、AWS DMS コンソールを使用するか、AWS CLI で `--timestream-settings '{"EndpointSetting": "value", ...}'` の JSON 構文を指定して `create-endpoint` コマンドを使用して設定を指定します。

次の表は、ターゲットとしての Timestream で使用できるエンドポイント設定を示しています。

名前	説明
`MemoryDuration`	この属性を設定して保持期間を指定し、移行したデータを Timestream のメモリストアに保存します。時間は時間単位で測定されます。Timestream のメモリストアは、高い取り込みスループットと高速アクセスを実現するように最適化されています。 デフォルト値: 24 (時間) 有効な値: 1 ~ 8,736 (1 時間 ~ 12 か月を時間単位で測定) 例: `--timestream-settings '{"MemoryDuration": 20}'`
`DatabaseName`	この属性を設定して、ターゲット Timestream データベース名を指定します。 タイプ: 文字列 例: `--timestream-settings '{"DatabaseName": " db_name"}'`
`TableName`	この属性を設定して、ターゲット Timestream テーブル名を指定します。

名前	説明
	タイプ: 文字列 例: `--timestream-settings '{"TableName": "table_name "}'`
`MagneticDuration`	この属性を設定して、Timestream テーブルに適用されるマグネティック持続時間を日単位で指定します。これは取り込まれたデータの保持期間です。Timestream は、保存期間を超えるタイムスタンプをすべて削除します。詳細については、「Amazon Timestream デベロッパーガイド」の「ストレージ」をご覧ください。https://docs.aws.amazon.com/timestream/latest/developerguide/ 例: `--timestream-settings '{"MagneticDuration": "3"}'`
`CdcInsertsAndUpdates`	この属性を `true` に設定すると、AWS DMS が挿入と更新のみを適用し、削除を適用しないように指定できます。Timestream ではレコードの削除を許可しないため、この値は `false` となり、AWS DMS は Timestream データベース内の対応するレコードを削除せずに NULL にします。詳細については、「制限事項」を参照してください。 デフォルト値: `false` 例: `--timestream-settings '{"CdcInsertsAndUpdates": "true"}'`

名前	説明
EnableMagneticStoreWrites	この属性を true に設定し、マグネティックストアの書き込みを有効にします。この値が false の場合、AWS DMS はデフォルトではマグネティックストアの書き込みを許可しないため、ターゲットテーブルのメモリストア保持期間より古いタイムスタンプのレコードは書き込まれません。詳細については、「Amazon Timestream デベロッパーガイド」の「書き込みのベストプラクティス」を参照してください。https://docs.aws.amazon.com/timestream/latest/developerguide/ デフォルト値: false 例: --timestream-settings '{"EnableMagneticStoreWrites": "true"}'

Amazon Timestream ターゲットエンドポイントの作成と変更

IAM ロールを作成して最小限のアクセス許可を設定すると、AWS DMS コンソールを使用するか、AWS CLI の create-endpoint コマンドを --timestream-settings '{"*EndpointSetting*": *"value"*, *...*}' JSON 構文で使用して、Amazon Timestream ターゲットエンドポイントを作成できます。

以下の例は、AWS CLI を使用して Timestream ターゲットエンドポイントを作成および変更する方法を示しています。

Timestream ターゲットエンドポイントの作成コマンド

```
aws dms create-endpoint –endpoint-identifier timestream-target-demo
--endpoint-type target –engine-name timestream
--service-access-role-arn arn:aws:iam::123456789012:role/my-role
--timestream-settings
{
    "MemoryDuration": 20,
    "DatabaseName":"db_name",
    "MagneticDuration": 3,
    "CdcInsertsAndUpdates": true,
    "EnableMagneticStoreWrites": true,
```

```
}
```

Timestream ターゲットエンドポイントの変更コマンド

```
aws dms modify-endpoint –endpoint-identifier timestream-target-demo
--endpoint-type target –engine-name timestream
--service-access-role-arn arn:aws:iam::123456789012:role/my-role
--timestream-settings
{
    "MemoryDuration": 20,
    "MagneticDuration": 3,
}
```

データを Timestream トピックに移行するためのオブジェクトマッピングの使用

AWS DMS は、テーブルマッピングルールを使用して、ソースのデータをターゲット Timestream トピックにマッピングします。ターゲットトピックにデータをマッピングするために、オブジェクトマッピングと呼ばれるテーブルマッピングルールのタイプを使用します。オブジェクトマッピングを使用して、ソースのデータレコードをどのように Timestream トピックに発行されたデータレコードにマッピングするかを定義します。

Timestream トピックには、パーティションキー以外にプリセット構造はありません。

> **Note**
>
> オブジェクトマッピングは必ずしも使用する必要はありません。通常のテーブルマッピングは、さまざまな変換に使用できます。ただし、パーティションキータイプは次のデフォルト動作に従います。
>
> - プライマリキーはフルロードのパーティションキーとして使用されます。
> - 並行適用タスク設定を使用しない場合は、`schema.table` が CDC のパーティションキーとして使用されます。
> - 並列適用タスク設定を使用する場合、プライマリキーは CDC のパーティションキーとして使用されます。

オブジェクトマッピングルールを作成するには、`object-mapping` として `rule-type` を指定します。このルールが、使用したいオブジェクトマッピングのタイプを指定します。ルールの構造は次のとおりです。

```
{
    "rules": [
        {
            "rule-type": "object-mapping",
            "rule-id": "id",
            "rule-name": "name",
            "rule-action": "valid object-mapping rule action",
            "object-locator": {
                "schema-name": "case-sensitive schema name",
                "table-name": ""
            }
        }
    ]
}
```

```
{
    "rules": [
        {
            "rule-type": "object-mapping",
            "rule-id": "1",
            "rule-name": "timestream-map",
            "rule-action": "map-record-to-record",
            "target-table-name": "tablename",
            "object-locator": {
                "schema-name": "",
                "table-name": ""
            },
            "mapping-parameters": {
                "timestream-dimensions": [
                    "column_name1",
                     "column_name2"
                ],
                "timestream-timestamp-name": "time_column_name",
                "timestream-multi-measure-name": "column_name1or2",
                "timestream-hash-measure-name":  true or false,
                "timestream-memory-duration": x,
                "timestream-magnetic-duration": y
            }
        }
    ]
}
```

AWS DMS では現在、rule-action パラメータに対する有効な値として map-record-to-record および map-record-to-document のみがサポートされています。map-record-to-record および map-record-to-document の値は、exclude-columns 属性リストの一部として除外されてないものとして AWS DMS がデフォルトで記録するものを指定します。これらの値は、どのような方法でも属性マッピングに影響しません。

リレーショナルデータベースから Timestream トピックに移行する場合は map-record-to-record を使用します。このルールタイプでは、Timestream トピックのパーティションキーとしてリレーショナルデータベースの taskResourceId.schemaName.tableName 値を使用し、ソースデータベース内の各列の属性を作成します。map-record-to-record を使用する場合、exclude-columns 属性リストに示されていないソーステーブル内の列について、AWS DMS はターゲットトピックに対応する属性を作成します。この対応する属性は、そのソース列が属性マッピングで使用されているかどうかにかかわらず作成されます。

map-record-to-record を理解するための 1 つの方法は、実際の動作を確認することです。この例では、次の構造とデータを含むリレーショナルデータベースのテーブルの行から始めると想定してください。

FirstName	LastName	StoreId	HomeAddress	HomePhone	WorkAddress	WorkPhone	DateofBirth
Randy	Marsh	5	221B Baker Street	1234567890	31 Spooner Street, Quahog	9876543210	02/29/1988

この情報を Test という名前のスキーマから Timestream トピックに移行するには、データをターゲットトピックにマッピングするルールを作成します。以下のルールはマッピングを示しています。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "rule-action": "include",
```

```
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            }
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",
            "rule-name": "DefaultMapToTimestream",
            "rule-action": "map-record-to-record",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Customers"
            }
        }
    ]
}
```

Timestream トピックとパーティションキー (この例では `taskResourceId.schemaName.tableName`) を指定した場合、Timestream ターゲットトピックのサンプルデータを使用した結果のレコード形式は次のとおりです。

```
  {
     "FirstName": "Randy",
     "LastName": "Marsh",
     "StoreId":  "5",
     "HomeAddress": "221B Baker Street",
     "HomePhone": "1234567890",
     "WorkAddress": "31 Spooner Street, Quahog",
     "WorkPhone": "9876543210",
     "DateOfBirth": "02/29/1988"
  }
```

Amazon Timestream を AWS Database Migration Service のターゲットとして使用する場合の制限

Amazon Timestream をターゲットとして使用する場合は、以下の制限が適用されます。

- ディメンションとタイムスタンプ: Timestream はソースデータ内のディメンションとタイムスタンプを複合プライマリキーのように使用します。また、これらの値を更新することを許可しません。つまり、ソースデータベース内のレコードのタイムスタンプまたはディメンションを変更す

ると、Timestream データベースは新しいレコードを作成しようとします。そのため、レコードのディメンションまたはタイムスタンプを別の既存のレコードと一致するように変更すると、AWS DMS は、新しいレコードを作成したり、以前の対応するレコードを更新したりせずに、別のレコードの値を更新します。

- DDL コマンド: AWS DMS の現在のリリースでは `CREATE TABLE` および `DROP TABLE` DDL コマンドのみをサポートしています。
- レコードの制限: Timestream には、レコードサイズやメジャーサイズなど、レコードに関する制限があります。詳細については、「Amazon Timestream デベロッパーガイド」の「クォータ」を参照してください。https://docs.aws.amazon.com/
- レコードと NULL 値の削除: Timestream はレコードの削除をサポートしていません。ソースから削除されたレコードの移行をサポートするために、AWS DMS は、Timestream ターゲットデータベースのレコード内の対応するフィールドをクリアします。AWS DMS は、対応するターゲットレコード内のフィールドの値を、数値フィールドの場合は 0、テキストフィールドの場合は null、ブールフィールドの場合は false に変更します。
- ターゲットとしての Timestream は、リレーショナルデータベース (RDBMS) 以外のソースをサポートしていません。
- AWS DMS は、以下のリージョンでのみ、Timestream をターゲットとしてサポートしています。
  - 米国東部 (バージニア北部)
  - 米国東部 (オハイオ)
  - 米国西部 (オレゴン)
  - 欧州 (アイルランド)
  - 欧州 (フランクフルト)
  - アジアパシフィック (シドニー)
  - アジアパシフィック (東京)
- ターゲットとしての Timestream は、`TargetTablePrepMode` を `TRUNCATE_BEFORE_LOAD` に設定することをサポートしていません。この設定で `DROP_AND_CREATE` を使用しないことをお勧めします。

AWS DMS のターゲットとしての Amazon RDS for Db2 および IBM Db2 LUW の使用

AWS Database Migration Service (AWS DMS) を使用して、Db2 LUW データベースから Amazon RDS for Db2 またはオンプレミスの Db2 データベースにデータを移行できます。

AWS DMS がターゲットとしてサポートする Db2 LUW のバージョンについては、「のターゲット AWS DMS」を参照してください。

Secure Sockets Layer (SSL) を使用して、Db2 LUW エンドポイントとレプリケーションインスタンスとの接続を暗号化できます。Db2 LUW エンドポイントで SSL を使用する方法の詳細については、「での SSL の使用 AWS Database Migration Service」を参照してください。

AWS DMS のターゲットとして Db2 LUW を使用する場合の制限

AWS DMS のターゲットとして Db2 LUW データベースを使用する場合は、以下の制限が適用されます。Db2 LUW をソースとして使用する場合の制限については、「Db2 LUW をソースとして使用する場合の制限事項 AWS DMS」を参照してください。

- Db2 LUW または Db2 for z/OS がソースである場合、AWS DMS は Db2 LUW のみをターゲットとしてサポートします。
- Db2 LUW をターゲットとして使用する場合、フル LOB モードのレプリケーションはサポートされません。
- Db2 LUW をターゲットとして使用する場合、フルロードフェーズでは XML データ型がサポートされません。これは IBM dbload ユーティリティの制限です。詳細については、IBM Informix Serversドキュメントの「dbload ユーティリティ」を参照してください。
- AWS DMS は、二重引用符 (") に対応する値を持つ BLOB フィールドを切り捨てます。これは IBM dbload ユーティリティの制限です。

AWS DMS のソースとして Db2 LUW を使用する場合のエンドポイント設定

エンドポイント設定を使用して、追加の接続属性を使用する場合と同じように、Db2 LUW データベースを設定できます。ターゲットエンドポイントを作成する際に、AWS DMS コンソールを使用するか、AWS CLI で `--ibm-db2-settings '{"EndpointSetting": "value", ...}'` の JSON 構文を指定して `create-endpoint` コマンドを使用して設定を指定します。

次の表は、ターゲットとしての Db2 LUW で使用できるエンドポイント設定を示しています。

名前	説明
`KeepCsvFiles`	true の場合、AWS DMS は、データのレプリケーションに使用した .csv ファイルを Db2 LUW ターゲットに保存します。DMS は、これらのファイルを分析とトラブルシューティングに使用します。

名前	説明
LoadTimeout	Db2 ターゲットに対して DMS が実行する操作を AWS DMS でタイムアウトするまでの時間 (ミリ秒単位)。デフォルト値は 1200 (20 分) です。
MaxFileSize	Db2 LUW へのデータ転送に使用する .csv ファイルの最大サイズ (KB 単位) を指定します。
WriteBufferSize	DMS レプリケーションインスタンスで .csv ファイルをローカルディスクに生成するときに使用するメモリ内ファイル書き込みバッファのサイズ (KB 単位)。デフォルト値は 1024 (1 MB) です。

AWS DMS ソースエンドポイントとターゲットエンドポイントとしての VPC エンドポイントの設定

AWS DMS は、Amazon Virtual Private Cloud (VPC) エンドポイントをソースとターゲットとしてサポートします。AWSDMS は、ソースデータベースとターゲットデータベースへの明示的に定義されたルートが AWS DMS VPC で定義されている限り、Amazon VPC エンドポイントを使用して任意の AWS ソースデータベースまたはターゲットデータベースに接続できます。

Amazon VPC エンドポイントがサポートされるため、追加のネットワーク設定やセットアップを行わずに、AWS DMS は、すべてのレプリケーションタスクのエンドツーエンドのネットワークセキュリティを簡単に管理できます。すべてのソースエンドポイントとターゲットエンドポイントに VPC エンドポイントを使用すると、すべてのトラフィックを確実に VPC 内に維持し、コントロールできます。AWS DMS バージョン 3.4.7 以降にアップグレードするには、VPC エンドポイントを使用するか、次の Amazon Web Services と連携するすべてのソースエンドポイントとターゲットエンドポイントへのパブリックルートを使用するように AWS DMS を設定する必要があります。

- Amazon S3
- Amazon Kinesis
- AWS Secrets Manager
- Amazon DynamoDB
- Amazon Redshift
- Amazon OpenSearch Service

次の説明のとおり、バージョン 3.4.7 以降の AWS DMS をサポートするには、VPC エンドポイントが必要になる場合があります。

AWS DMS バージョン 3.4.7 以降への移行で影響を受けるのはケースとは

上記の単一または複数の AWS DMS エンドポイントを使用していて、エンドポイントがパブリックにルーティング可能でないか、VPC エンドポイントがまだ関連付けられていない場合は、影響を受けます。

AWS DMS バージョン 3.4.7 以降に移行しても影響を受けないケースとは

次の場合は影響を受けません。

- 上記の AWS DMS エンドポイントを 単独または複数で使用していない。
- 上記のエンドポイントのいずれかを使用していており、エンドポイントがパブリックにルーティング可能である。
- 上記のエンドポイントのいずれかを使用しており、VPC エンドポイントに関連付けられている。

AWS DMS バージョン 3.4.7 以降への移行の準備

上記のエンドポイントのいずれかを使用している場合、AWS DMS タスクが失敗するのを回避するには、AWS DMS をバージョン 3.4.7 以降にアップグレードする前に、次のいずれかの手順を実行します。

- 影響を受ける AWS DMS エンドポイントをパブリックにルーティングできるようにします。例えば、AWS DMS レプリケーションインスタンスで既に使用されている VPC にインターネットゲートウェイ (IGW) ルートを追加して、ソースとターゲットのすべてのエンドポイントをパブリックにルーティングできるようにします。
- 次の説明のとおり、AWS DMS が使用するすべてのソースエンドポイントとターゲットエンドポイントにアクセスする VPC エンドポイントを作成します。

AWS DMS のソースエンドポイントとターゲットエンドポイントに使用する既存の VPC エンドポイントについては、必ず XML ポリシードキュメントに `dms-vpc-role` 準拠する信頼ポリシーを使用します。このポリシーの詳細については、「AWS CLI および AWS DMS API で使用する IAM ロールの作成」を参照してください。

上記以外には、レプリケーションインスタンスが配置された VPC に VPC エンドポイントを追加して、レプリケーションインスタンスを VPC エンドポイントとして設定する方法があります。パブ

リックエンドポイントなしでレプリケーションインスタンスを設定した場合、パブリックにアクセス可能な VPC エンドポイントをレプリケーションインスタンスが配置された VPC に追加すると、パブリックにアクセスできるようになります。レプリケーションインスタンスを VPC エンドポイントと明確に関連付けるための作業は特にありません。

> **Note**
>
> サービスごとに固有の VPC エンドポイント設定がある場合があります。例えば、AWS Secrets Manager を使用する場合、ルーティングテーブルを調整する必要は通常ありません。各サービスの具体的な要件を必ず確認します。

レプリケーションインスタンスが配置された VPC に VPC エンドポイントを作成する

1. AWS Management Console にサインインして、https://console.aws.amazon.com/vpc/ で Amazon VPC コンソールを開きます。
2. VPC コンソールのメニューバーで、AWS DMS レプリケーションインスタンスと同じ AWS リージョン を選択します。
3. VPC ナビゲーションペインで[エンドポイント] を選択します。
4. [エンドポイント] を選択して、[エンドポイントを作成] をクリックします。
5. 必要に応じて名前タグを指定できます。例えば、**`my-endpoint-DynamoDB-01`** と指定します。
6. S3 または DynamoDB 専用の [サービス] で、[タイプ] が [ゲートウェイ] に設定された [サービス名] を選択します。
7. [VPC] で、AWS DMS レプリケーションインスタンスと同じ VPC を選択してエンドポイントを作成します。
8. [ルートテーブル] で、使用できるすべての [ルートテーブル ID] 値を選択します。
9. アクセスコントロールを指定するには、[ポリシー] で [フルアクセス] を選択します。ポリシー作成ツールを使用して独自のアクセス制御を指定する場合は、[カスタム] を選択します。いずれの場合も、JSON ポリシードキュメント `dms-vpc-role` に準拠する信頼ポリシーを使用します。このポリシーの詳細については、「AWS CLI および AWS DMS API で使用する IAM ロールの作成」を参照してください。
10. [エンドポイント] で、新たに作成した VPC エンドポイントの [ステータス] が [利用可能] であることを確認します。

AWS DMS レプリケーションインスタンスのための VPC エンドポイントの設定の詳細については、「データベース移行のネットワーク設定」を参照してください。AWS サービスにアクセスするための一般的なインターフェイス VPC エンドポイントの作成の詳細については、「AWS PrivateLink ガイド」の「インターフェイス VPC エンドポイントを使用して AWS のサービスにアクセスする」を参照してください。VPC エンドポイントの AWS DMS リージョンの可用性については、「AWS リージョン表」を参照してください。

AWS DMS によりサポートされている DDL ステートメント

データ移行プロセス中は、ソースデータベース上でデータ定義言語 (DDL) ステートメントを実行できます。これらのステートメントはレプリケーションサーバー上で、ターゲットデータベースにレプリケートされます。

サポートされている DDL ステートメントは以下のとおりです。

- Create table
- Drop table
- Rename table
- Truncate table
- Add column
- Drop column
- Rename column
- Change column data type

DMS は一部のソースエンジンタイプでサポートされている DDL ステートメントをすべてキャプチャするわけではありません。また、DMS は DDL ステートメントを特定のターゲットエンジンに適用するときに DDL ステートメントを異なる方法で処理します。特定のソースでサポートされている DDL ステートメントと、ターゲットに適用する方法の詳細については、ソースエンドポイントとターゲットエンドポイントの特定のドキュメンテーショントピックをご参照ください。

タスク設定を使用して、変更データキャプチャ (CDC) 中に DMS が DDL の動作を処理する方法を設定できます。詳細については、「変更処理の DDL 処理のタスク設定」を参照してください。

AWS DMS タスクの使用

AWS Database Migration Service (AWS DMS) タスクは、すべての処理が行われる場所です。ログ記録要件、制御テーブルデータ、エラー処理など、移行と特別な処理に使用するテーブル (またはビュー) とスキーマを指定します。

タスクは、3 つの主なフェーズで構成できます。

- 既存データの移行 (フルロード)
- キャッシュされた変更の適用
- 継続的なレプリケーション (変更データキャプチャ)

AWS DMS 移行タスクがデータを移行する方法の詳細と概要については、「の概要 AWS DMS」を参照してください。

移行タスクを作成するとき、いくつかのことを知っておく必要があります。

- 移行タスクを作成する前に、ソースエンドポイント、ターゲットエンドポイント、およびレプリケーションインスタンスを作成していることを確認します。
- 移行タスクを調整するために多くのタスク設定を指定できます。それらは、AWS Management Console、 AWS Command Line Interface (AWS CLI)、または AWS DMS API を使用して設定できます。これらの設定には、移行エラーの処理方法、エラーのログ記録、および制御テーブル情報を指定することが含まれます。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。
- タスクを作成した後、直ちに実行できます。必要なメタデータ定義を含むターゲットテーブルが自動的に作成されてロードされるため、継続的なレプリケーションを指定できます。
- デフォルトでは、タスクを作成するとすぐに AWS DMS によりタスクが開始されます。ただし、状況によっては、タスクの開始を延期できます。たとえば、AWS CLI を使用するとき、タスクを作成するプロセスと、トリガーイベントに基づいてタスクを開始する別のプロセスが存在する場合があります。必要に応じて、タスクの開始を延期できます。
- AWS CLI コンソール、AWS DMS、または API を使用して、タスクのモニタリング、停止、再開を行うことができます。AWS DMS APIを使用してタスクを停止する方法については、AWS DMSAPI リファレンスの「StopReplicationTask」をご参照ください。

AWS DMS タスクを操作するときに実行できるアクションを以下に示します。

タスク	関連資料
[Creating a task] (タスクの作成) タスクを作成するときに、ソース、ターゲット、およびレプリケーションインスタンスを、移行設定とともに作成します。	[Creating a task] (タスクの作成)
[Creating an ongoing replication task] (継続的レプリケーション タスク作成) ソースとターゲット間で、継続的なレプリケーションを提供するようにタスクをセットアップできます。	AWS DMS を使用した継続的なレプリケーションのタスクの作成
[Applying task settings] (タスク設定の適用) 各タスクには、データベース移行の必要に応じて設定できる設定があります。これらの設定は JSON ファイルで作成します。その一部の設定は AWS DMS コンソールで指定できます。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。	AWS Database Migration Service タスクのタスク設定の指定
[Using table mapping] (テーブルマッピングの使用) テーブル マッピングでは、いくつかのタイプのルールを使用して、テーブルの追加タスク設定を指定します。これらのルールを使用すると、データソース、ソース スキーマ、	選択ルール 選択ルールと選択アクション 変換ルール 変換ルールおよび変換アクション テーブル設定ルール

タスク	関連資料
テーブルとビュー、データ、タスク中に発生するテーブルおよびデータの変換、これらのテーブルおよび列をソースからターゲットに移行する方法の設定を指定できます。	テーブルとコレクション設定のルールとオペレーション
[Running premigration task assessments] (移行前のタスク評価の実行) 移行中に問題を引き起こす可能性のあるサポートされているソースおよびターゲットデータベースに関する問題を示す移行前タスク評価を有効にして実行することができます。これには、サポートされていないデータ型および、インデックスとプライマリ キーの不一致、その他の競合するタスク設定などの問題が含まれます。これらの移行前評価は、タスクを実行する前に実行され、移行中に発生する可能性のある問題を特定します。	タスクの移行前評価の有効化と操作
[Data validation] (データ検証) データ検証は、AWS DMS でターゲットデータストア上のデータを、ソースデータストアからのデータと比較するために使用できるタスク設定です。	AWS DMS データ検証.
[Modifying a task] (タスクの変更) タスクが停止した際に、そのタスクの設定を変更できます。	[Modifying a task] (タスクの変更)

タスク	関連資料
[Moving a task] (タスクの移動) タスクが停止すると、そのタスクを別のレプリケーション インスタンスに移動できます。	タスクの移動
タスク実行中のテーブルの再ロード タスク実行中にエラーが発生した場合には、タスク実行中にテーブルを再ロードできます。	タスク実行中のテーブルの再ロード
フィルターの適用 ソースフィルタを使用すると、ソースからターゲットに転送されるレコードの数とタイプを制限できます。例えば、本社を拠点とする従業員だけがターゲットデータベースに移行されるように指定できます。データの列にフィルタを適用します。	ソースフィルタの使用
[Monitoring a task] (タスクのモニタリング) タスクのパフォーマンスとそのタスクが使用するテーブルに関する情報を取得するためには、複数の方法があります。	AWS DMS タスクのモニタリング
タスクログの管理 AWS DMS API または AWS CLI を使用してタスクログを表示および削除することができます。	AWS DMS タスクログの表示と管理

トピック

[Creating a task] (タスクの作成)

AWS DMS 移行タスクを作成するには、次の手順を実行します。

- 移行タスクを作成する前に、ソースエンドポイント、ターゲットエンドポイント、およびレプリケーションインスタンスを作成します。
- 移行方法を選択します。
  - [Migrating data to the target database] (データをターゲットデータベースに移行する) – このプロセスでは、ターゲットデータベースにファイルまたはテーブルを作成し、ターゲットで必要なメタデータを自動的に定義します。また、ソースのデータをテーブルに入力します。テーブルのデータは、効率を高めるために並列でロードされます。このプロセスは、 の既存のデータ移行オプションであり AWS Management Console 、 API `Full Load`で呼び出されます。
  - [Capturing changes during migration] (移行中に変更をキャプチャする) – このプロセスでは、データがソースからターゲットに移行されているときに発生した変更をソースデータベースにキャプチャします。最初にリクエストされたデータの移行が完了すると、変更データキャプチャ (CDC) プロセスがキャプチャした変更をターゲットデータベースに適用します。変更は、1 つのコミットされたトランザクションユニットとしてキャプチャおよび適用され、複数の異なるターゲットテーブルを 1 つのソースコミットとして更新できます。このアプローチでは、ターゲットデータベースにおけるトランザクションの完全性が保証されます。このプロセスは、コンソールでは [Migrate existing data and replicate ongoing changes] (既存データの移行、継続的変更のレプリケーション) オプションとなっており、API では `full-load-and-cdc` と呼ばれています。

  - [Replicating only data changes on the source database] (データ変更のみソースデータベースにレプリケートする) – このプロセスでは、ソースデータベース管理システム (DBMS) の復旧ログファイルを読み取り、各トランザクションのエントリをまとめます。場合によっては、妥当な時間内にターゲットに変更を適用 AWS DMS できないことがあります (ターゲットにアクセスできない場合など) 。このような場合、 は必要な期間、レプリケーションサーバー上の変更を AWS DMS バッファリングします。DBMS ログを再読み取りしないため、長時間かかる可能性があります。このプロセスは、 AWS DMS コンソールでは [Replicate data changes only] (データ変更のみをレプリケートする) オプションを指します。
- ソースでラージバイナリオブジェクト (LOB) を処理する方法を決定します。詳細については、「AWS DMS タスクでのソースデータベースの LOB サポートの設定」をご参照ください。
- 移行タスクの設定を指定します。これには、ログ記録の設定、移行の制御テーブルに書き込まれるデータ、エラーの処理方法、およびその他の設定が含まれます。タスク設定の詳細については、「AWS Database Migration Service タスクのタスク設定の指定」をご参照ください。
- テーブルマッピングを設定して、選択するルールを定義し、移行するデータをフィルタします。テーブルマッピングの詳細については、「テーブルマッピングを使用して、タスクの設定を指定する」をご参照ください。マッピングを指定する前に、ソースデータベースとターゲットデータベースのデータ型マッピングのドキュメントセクションを確認してください。
- タスクを実行する前に、移行前タスク評価を有効にして実行します。移行前評価の詳細については、「タスクの移行前評価の有効化と操作」をご参照ください。
- データを移行するタスクに必要な補足データを指定します。詳細については、「タスク設定の補足データを指定する」をご参照ください。

[タスクの作成] ページでタスクへの情報の指定が完了するとすぐにタスクを開始することができます。または、後でダッシュボードページからタスクを開始することもできます。

次の手順では、レプリケーションインスタンス情報とエンドポイントがすでに指定されていることを前提としています。統合の設定の詳細については、「ソースおよびターゲットエンドポイントの作成」をご参照ください。

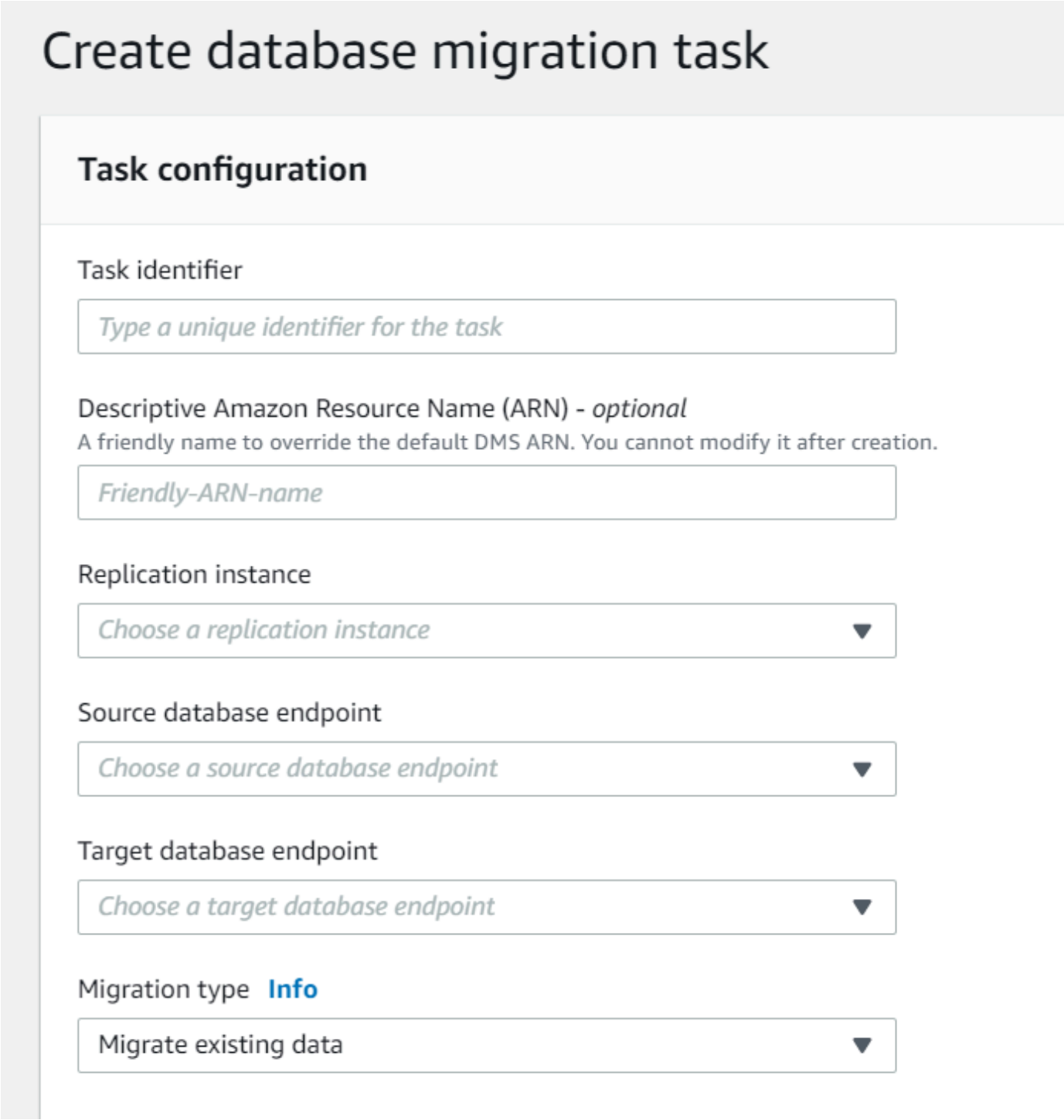
移行タスクを作成するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。

   AWS Identity and Access Management (IAM) ユーザーとしてサインインしている場合は、 にアクセスするための適切なアクセス許可があることを確認してください AWS DMS。必要なアク

セス権限の詳細については、「AWS DMSの使用に必要な IAM アクセス許可」をご参照ください。

2. ナビゲーションペインで、[データベース移行タスク] を選択し、[タスクの作成] をクリックします。
3. [データ移行タスクの作成] ページの [タスクの設定] セクションで、タスクのオプションを指定します。次の表で設定について説明します。



使用するオプション	この操作を行います
タスク識別子	タスクの名前を入力します。

使用するオプション	この操作を行います
説明的な Amazon リソースネーム (ARN) - オプション	デフォルトの AWS DMS ARN を上書きするわかりやすい名前。タスク作成後は名前の変更はできない。
レプリケーションインスタンス	使用するレプリケーションインスタンスを表示します。
ソースデータベースエンドポイント	使用するソースエンドポイントを表示します。
ターゲットデータベースエンドポイント	使用するターゲットエンドポイントを表示します。
移行タイプ	使用する移行方法を選択します。既存のデータのみをターゲットデータベースへ移行するか、移行したデータに加えて継続的な変更もターゲットデータベースに送信するかを選択できます。

4. [タスク設定] セクションでは、タスクの編集、ターゲットテーブル作成モード、タスク停止、LOB 設定、検証、ログ記録の値を指定します。

使用するオプション	この操作を行います
編集モード	タスク設定を指定するためにウィザードを使用するか JSON エディターを使用するかを選択する。ウィザードを選択すると、次のオプションが表示される。
ソーストランザクションの CDC 開始モード	この設定は、前のセクションの[移行タイプ]で [データ変更のみをレプリケート] を選択した場合にのみ表示される。 カスタム CDC 開始モードを無効にする – このオプションを選択すると、次の [作成時に自動的に行う] オプションを使用してタスクを自動的に開始するか、コンソールを使用して手動でタスクを開始できる。 カスタム CDC 開始モードを有効にする – このオプションを選択すると、変更の処理を開始するカスタム UTC 開始時刻を指定できる。

使用するオプション	この操作を行います
ターゲットテーブル作成モード	この設定は、前のセクションの [移行タイプ] で、[既存のデータを移行する] または [既存のデータを移行して継続的な変更をレプリケートする] を選択した場合にのみ表示される。 何もしない – 何もしないモードでは、 はターゲットテーブルがターゲットに事前に作成されていることを AWS DMS 前提としています。テーブルが空でない場合、データ移行中に競合が発生し、DMS タスクエラーが発生する可能性がある。ターゲットテーブル不在の場合、DMS で自動的にテーブルが作成されます。テーブルの構造はそのまま変更されず、既存のデータはテーブル内に残ります。[何もしない] モードは、CDC のみのタスクに適切な選択肢です。この場合、ターゲットテーブルはソースからバックフィルされ、ソースとターゲットの同期を維持するために継続的なレプリケーションが適用されます。テーブルを事前に作成するには、 AWS Schema Conversion Tool (AWS SCT) を使用できる。詳細については、「 のインストール AWS SCT」を参照してください。 [Drop tables on target](ターゲット上のテーブルを削除]) – ターゲット上のテーブルを削除 モードの場合、AWS DMS はターゲットテーブルを削除し、これらを移行の開始前に再作成します。このアプローチにより、移行の開始時にターゲットテーブルが空になります。 は、テーブル、プライマリキー、場合によっては一意のインデックスなどのデータを効率的に移行するために必要なオブジェクトのみ AWS DMS を作成します。 AWS DMS は、セカンダリインデックス、非プライマリキー制約、または列データのデフォルトを作成しません。全ロードと CDC または CDC のみのタスクを実行する場合は、この時点で移行を一時停止することをお勧めします。次に、更新および削除ステート

使用するオプション	この操作を行います
	メントのフィルタリングをサポートするセカンダリインデックスを作成します。 [ターゲット上のテーブルを削除] モードを使用する場合は、必要に応じてターゲットデータベースで一部の設定を行います。例えば、Oracle ターゲットの場合、セキュリティ上の理由から、 AWS DMS はスキーマ (データベースユーザー) を作成できません。この場合、移行の開始時に がテーブルを作成 AWS DMS できるように、スキーマユーザーを事前に作成します。他のほとんどのターゲットタイプでは、 はスキーマと関連するすべてのテーブルを適切な設定パラメータで AWS DMS 作成します。 切り捨て – 切り捨てモードでは、 は移行を開始する前にすべてのターゲットテーブル AWS DMS を切り捨てます。ターゲットテーブル不在の場合、DMS で自動的にテーブルが作成されます。テーブル構造はそのまま維持されますが、ターゲットではテーブルが切り捨てられます。切り捨てモードは全ロードまたは全ロード + CDC の移行に適しています。この場合、移行の開始前にターゲットスキーマは作成済みとします。テーブルを事前に作成するには、 AWS SCTを使用できます。詳細については、「 のインストール AWS SCT」を参照してください。 **Note** ターゲットが MongoDB の場合、[切り捨て] モードではターゲットでテーブルが切り捨てられない。代わりに、コレクションを削除し、すべてのインデックスを失います。ターゲットが MongoDB の場合は [切り捨て] モードの使用は避ける。

使用するオプション	この操作を行います
全ロードの完了後にタスクを停止する	この設定は、前のセクションの [移行タイプ] で [既存のデータを移行して継続的な変更をレプリケートする] を選択した場合にのみ表示される。 [Don't stop] (停止しない) - タスクを停止せず、キャッシュされた変更をすぐに適用したら、そのまま続行します。 キャッシュされた変更の適用前に停止する – キャッシュした変更を適用する前にタスクを停止する。この方法を使用して、変更の適用を高速化する可能性があるセカンダリインデックスを追加できます。 キャッシュされた変更の適用後に停止 – キャッシュした変更が適用された後にタスクを停止する。トランザクショナルな適用を使用する場合、この方法を使用して、外部キーを追加できます。
[Include LOB columns in replication] (レプリケーションに LOB 列を含める)	[Don't include LOB columns] (LOB 列を含めない) – LOB 列は移行対象から除外されます。 フル LOB モード – size に関係なく、完全な LOBs を移行します。 は LOBs チャンクサイズパラメータによって制御されるチャンクで LOB を分割して AWS DMS 移行します。このモードは制限付き LOB モードを使用するよりも低速です。 [Limited LOB mode] (制限付き LOB モード) – LOB を [Max LOB size] (最大 LOB サイズ) パラメータの値まで切り詰めます。このモードは完全 LOB モードを使用するよりも高速です。
最大 LOB サイズ (KB)	[制限付き LOB モード] では、[最大 LOB サイズ] の設定を超える LOB 列は指定した [最大 LOB サイズ] まで切り捨てられます。

使用するオプション	この操作を行います
検証の有効化	データの検証を有効にして、ソースからターゲットにデータが正確に移行されることを確認します。詳細については、「AWS DMS データ検証」を参照してください。
CloudWatch ログを有効にする	Amazon による ログ記録を有効にします CloudWatch。

5. [移行前評価] セクションで、移行前評価を実行するかを選択します。移行前評価は、データベース移行タスクを開始する前に潜在的な移行の問題について警告を作成します。詳細については、「移行前評価の有効化と操作」を参照してください。
6. [移行タスクのスタートアップ設定] セクションで、タスクの作成後に自動的に開始するかを指定します。
7. [タグ] セクションで、タスクを整理するために必要なタグを指定します。タグを使用すると、IAM ロールとポリシーを管理して、DMS コストを追跡できます。詳細については、「リソースのタグ付け」を参照してください。
8. タスクの設定を完了した後、[タスクの作成] を選択します。

AWS Database Migration Service タスクのタスク設定の指定

各タスクには、データベース移行の必要に応じて設定できる設定があります。これらの設定は JSON ファイルで作成するか、一部の設定で AWS DMS コンソールを使用して指定できます。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。

以下に示すように、タスク設定には、いくつかの主要なタイプがあります。

トピック

- タスク設定例
- ターゲットメタデータのタスク設定
- 全ロードタスク設定
- Time Travel タスクの設定
- ロギングタスク設定
- 制御テーブルタスク設定

- ストリームバッファタスク設定
- 変更処理のチューニング設定
- データ検証タスクの設定
- 変更処理の DDL 処理のタスク設定
- 文字置換タスクの設定
- 前イメージタスク設定前
- エラー処理タスクの設定
- タスク設定の保存

タスク設定	関連資料
タスク評価レポートの作成 移行中に問題を発生させる可能性のある、サポートされていないデータ型を示すタスク評価レポートを作成できます。タスクを実行する前にタスクでこのレポートを実行して、潜在的な問題を見つけることができます。	タスクの移行前評価の有効化と操作
[Creating a task] (タスクの作成) タスクを作成するときに、ソース、ターゲット、およびレプリケーションインスタンスを、移行設定とともに作成します。	[Creating a task] (タスクの作成)
[Creating an ongoing replication task] (継続的レプリケーション タスク作成) ソースとターゲット間で、継続的なレプリケーションを提供するようにタスクをセットアップできます。	AWS DMS を使用した継続的なレプリケーションのタスクの作成

タスク設定	関連資料
[Applying task settings] (タスク設定の適用) 各タスクには、データベース移行の必要に応じて設定できる設定があります。これらの設定は JSON ファイルで作成するか、一部の設定で AWS DMS コンソールを使用して指定できます。	AWS Database Migration Service タスクのタスク設定の指定
[Data validation] (データ検証) データ検証を使用して、ターゲットデータストア上のデータとソースデータストアのデータ AWS DMS を比較します。	AWS DMS データ検証
[Modifying a task] (タスクの変更) タスクが停止した際に、そのタスクの設定を変更できます。	[Modifying a task] (タスクの変更)
タスク実行中のテーブルの再ロード タスク実行中にエラーが発生した場合には、タスク実行中にテーブルを再ロードできます。	タスク実行中のテーブルの再ロード
[Using table mapping] (テーブルマッピングの使用) テーブルマッピングは、データソース、ソーススキーマ、データ、およびタスク実行中に必要なすべての変換のタスク設定を指定するための複数のルールタイプを使用します。	選択ルール 選択ルールと選択アクション 変換ルール 変換ルールおよび変換アクション

タスク設定	関連資料
フィルターの適用 ソースフィルタを使用すると、ソースからターゲットに転送されるレコードの数とタイプを制限できます。例えば、本社を拠点とする従業員だけがターゲットデータベースに移行されるように指定できます。データの列にフィルタを適用します。	ソースフィルタの使用
[Monitoring a task] (タスクのモニタリング) タスクのパフォーマンスとそのタスクが使用するテーブルに関する情報を取得するためには、複数の方法があります。	AWS DMS タスクのモニタリング
タスクログの管理 AWS DMS API または を使用して、タスクログを表示および削除できます AWS CLI。	AWS DMS タスクログの表示と管理

タスク設定例

AWS Management Console または のいずれかを使用して AWS CLI 、レプリケーションタスクを作成できます。を使用する場合は AWS CLI、JSON ファイルを作成し、タスクオペレーションの ReplicationTaskSettingsパラメータとして JSON ファイルの file:// URI を指定してCreateReplication、タスク設定を行います。

次の例は、 を使用して CreateReplicationTaskオペレーションを AWS CLI 呼び出す方法を示しています。

```
aws dms create-replication-task \
--replication-task-identifier MyTask \
```

```
--source-endpoint-arn arn:aws:dms:us-
west-2:123456789012:endpoint:ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890ABC \
--target-endpoint-arn arn:aws:dms:us-
west-2:123456789012:endpoint:ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890ABC \
--replication-instance-arn arn:aws:dms:us-
west-2:123456789012:rep:ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890ABC \
--migration-type cdc \
--table-mappings file://tablemappings.json \
--replication-task-settings file://settings.json
```

前の例では、tablemappings.jsonというテーブルマッピングファイルを使用しています。テーブルマッピング例については、「テーブルマッピングを使用して、タスクの設定を指定する」をご参照ください。

タスク設定の JSON ファイルは次のようになります。

```
{
  "TargetMetadata": {
    "TargetSchema": "",
    "SupportLobs": true,
    "FullLobMode": false,
    "LobChunkSize": 64,
    "LimitedSizeLobMode": true,
    "LobMaxSize": 32,
    "InlineLobMaxSize": 0,
    "LoadMaxFileSize": 0,
    "ParallelLoadThreads": 0,
    "ParallelLoadBufferSize":0,
    "ParallelLoadQueuesPerThread": 1,
    "ParallelApplyThreads": 0,
    "ParallelApplyBufferSize": 100,
    "ParallelApplyQueuesPerThread": 1,
    "BatchApplyEnabled": false,
    "TaskRecoveryTableEnabled": false
  },
  "FullLoadSettings": {
    "TargetTablePrepMode": "DO_NOTHING",
    "CreatePkAfterFullLoad": false,
    "StopTaskCachedChangesApplied": false,
    "StopTaskCachedChangesNotApplied": false,
    "MaxFullLoadSubTasks": 8,
```

```
    "TransactionConsistencyTimeout": 600,
    "CommitRate": 10000
  },
    "TTSettings" : {
    "EnableTT" : true,
    "TTS3Settings": {
        "EncryptionMode": "SSE_KMS",
        "ServerSideEncryptionKmsKeyId": "arn:aws:kms:us-west-2:112233445566:key/
myKMSKey",
        "ServiceAccessRoleArn": "arn:aws:iam::112233445566:role/dms-tt-s3-access-role",
        "BucketName": "myttbucket",
        "BucketFolder": "myttfolder",
        "EnableDeletingFromS3OnTaskDelete": false
      },
    "TTRecordSettings": {
        "EnableRawData" : true,
        "OperationsToLog": "DELETE,UPDATE",
        "MaxRecordSize": 64
      }
  },
  "Logging": {
    "EnableLogging": false
  },
  "ControlTablesSettings": {
    "ControlSchema":"",
    "HistoryTimeslotInMinutes":5,
    "HistoryTableEnabled": false,
    "SuspendedTablesTableEnabled": false,
    "StatusTableEnabled": false
  },
  "StreamBufferSettings": {
    "StreamBufferCount": 3,
    "StreamBufferSizeInMB": 8
  },
  "ChangeProcessingTuning": {
    "BatchApplyPreserveTransaction": true,
    "BatchApplyTimeoutMin": 1,
    "BatchApplyTimeoutMax": 30,
    "BatchApplyMemoryLimit": 500,
    "BatchSplitSize": 0,
    "MinTransactionSize": 1000,
    "CommitTimeout": 1,
    "MemoryLimitTotal": 1024,
    "MemoryKeepTime": 60,
```

```
    "StatementCacheSize": 50
  },
  "ChangeProcessingDdlHandlingPolicy": {
    "HandleSourceTableDropped": true,
    "HandleSourceTableTruncated": true,
    "HandleSourceTableAltered": true
  },
  "LoopbackPreventionSettings": {
    "EnableLoopbackPrevention": true,
    "SourceSchema": "LOOP-DATA",
    "TargetSchema": "loop-data"
  },

  "CharacterSetSettings": {
    "CharacterReplacements": [ {
        "SourceCharacterCodePoint": 35,
        "TargetCharacterCodePoint": 52
      }, {
        "SourceCharacterCodePoint": 37,
        "TargetCharacterCodePoint": 103
      }
    ],
    "CharacterSetSupport": {
      "CharacterSet": "UTF16_PlatformEndian",
      "ReplaceWithCharacterCodePoint": 0
    }
  },
  "BeforeImageSettings": {
    "EnableBeforeImage": false,
    "FieldName": "",
    "ColumnFilter": "pk-only"
  },
  "ErrorBehavior": {
    "DataErrorPolicy": "LOG_ERROR",
    "DataTruncationErrorPolicy":"LOG_ERROR",
    "DataErrorEscalationPolicy":"SUSPEND_TABLE",
    "DataErrorEscalationCount": 50,
    "TableErrorPolicy":"SUSPEND_TABLE",
    "TableErrorEscalationPolicy":"STOP_TASK",
    "TableErrorEscalationCount": 50,
    "RecoverableErrorCount": 0,
    "RecoverableErrorInterval": 5,
    "RecoverableErrorThrottling": true,
    "RecoverableErrorThrottlingMax": 1800,
```

```
    "ApplyErrorDeletePolicy":"IGNORE_RECORD",
    "ApplyErrorInsertPolicy":"LOG_ERROR",
    "ApplyErrorUpdatePolicy":"LOG_ERROR",
    "ApplyErrorEscalationPolicy":"LOG_ERROR",
    "ApplyErrorEscalationCount": 0,
    "FullLoadIgnoreConflicts": true
  },
  "ValidationSettings": {
    "EnableValidation": false,
    "ValidationMode": "ROW_LEVEL",
    "ThreadCount": 5,
    "PartitionSize": 10000,
    "FailureMaxCount": 1000,
    "RecordFailureDelayInMinutes": 5,
    "RecordSuspendDelayInMinutes": 30,
    "MaxKeyColumnSize": 8096,
    "TableFailureMaxCount": 10000,
    "ValidationOnly": false,
    "HandleCollationDiff": false,
    "RecordFailureDelayLimitInMinutes": 1,
    "SkipLobColumns": false,
    "ValidationPartialLobSize": 0,
    "ValidationQueryCdcDelaySeconds": 0
  }
}
```

ターゲットメタデータのタスク設定

ターゲットメタデータ設定には、以下のものが含まれます。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。

- TargetSchema - ターゲットテーブルスキーマ名。このメタデータオプションが空の場合、ソーステーブルのスキーマが使用されます。 AWS DMS は、ソーススキーマが定義されていない場合、ターゲットデータベースの所有者プレフィックスをすべてのテーブルに自動的に追加します。このオプションは、MySQL 型のターゲットエンドポイントでは空のままにする必要があります。データマッピングでスキーマの名前を変更すると、この設定よりも優先されます。
- LOB 設定 - ラージ オブジェクト (LOB) の管理方法を決定する設定。SupportLobs=true と設定した場合、次のいずれかを true に設定する必要があります。
  - FullLobMode – このオプションを true に設定した場合、 LobChunkSize オプションの値を入力します。ターゲットにデータをレプリケートするときに使用する LOB チャンクサイズ

をキロバイト単位で入力します。FullLobMode オプションは、LOB のサイズが大きい場合に最適ですが、ロードが遅くなる傾向になります。LobChunkSize の推奨値は 64 キロバイトです。LobChunkSize の値を 64 キロバイト以上に設定すると、タスクが失敗する可能性があります。

- InlineLobMaxSize – この値は、全ロード中にインラインで転送LOBs AWS DMS を決定します。小さな LOB は、ソーステーブルから探すよりも転送する方が効率的です。全ロード中に、 はすべての LOBs AWS DMS をチェックし、 より小さい LOBs のインライン転送を実行しますInlineLobMaxSize。 は、 InlineLobMaxSizeの より大きいすべての LOBs AWS DMS を転送しますFullLobMode。InlineLobMaxSize のデフォルト値は 0 で、範囲は 1 ~ 102,400 キロバイト (100 MB) です。ほとんどの LOB が InlineLobMaxSize で設定した値よりも小さいことがわかっている場合のみ、InlineLobMaxSize に値を指定します。
- LimitedSizeLobMode – このオプションを true に設定した場合、 LobMaxSize オプションの値を入力します。個々の LOB の最大サイズをキロバイト単位で入力します。LobMaxSize の最大推奨値は 102,400 キロバイト (100 MB) です。

これらのタスクの LOB 設定を使用する条件の詳細については、「AWS DMS タスクでのソースデータベースの LOB サポートの設定」をご参照ください。また、個々のテーブルの LOB の管理を制御できます。詳細については、「テーブルとコレクション設定のルールとオペレーション」を参照してください。

- LoadMaxFileSize — データのロードにカンマ区切り値 (.csv) ファイルの使用をサポートする MySQL、PostgreSQL、Amazon Redshift などの CSV ベースのターゲット エンドポイントのオプション。LoadMaxFileSizeは、.csv ファイルなどの保存されたアンロードデータのディスク上の最大サイズを定義します。このオプションは、ターゲットのエンドポイント接続属性 maxFileSize より優先されます。0 (このオプションが接続属性を上書きしないことを示します) から 100,000 KB までの値を指定できます。
- BatchApplyEnabled - 各トランザクションを個別に適用するか、変更をバッチでコミットするかを決定します。デフォルト値は、falseです。

  BatchApplyEnabled を true に設定した場合、DMS は ソーステーブルにプライマリキー (PK) または一意のキー (UK) を必要とします。ソーステーブルに PK または UK がない場合、バッチ挿入のみが適用され、バッチ更新と削除は適用されません。

  BatchApplyEnabled が true に設定されていて、ターゲットテーブルに一意の制約とプライマリキーがある場合、 AWS DMS はエラーメッセージを生成します。BatchApplyEnabled が true に設定されている場合、一意の制約とプライマリキーの両方を持つターゲットテーブルはサポートされません。

BatchApplyEnabled が true に設定され、デフォルトのエラー処理ポリシーを持つテーブルからデータエラー AWS DMS が発生すると、 AWS DMS タスクは残りのテーブルのバッチモードから one-by-one モードに切り替わります。この動作を変更するには、"SUSPEND_TABLE" の次のポリシーに対するアクションを タスク設定 JSON ファイルの "ErrorBehavior" グループ プロパティに設定できます:

- DataErrorPolicy
- ApplyErrorDeletePolicy
- ApplyErrorInsertPolicy
- ApplyErrorUpdatePolicy

"ErrorBehavior" のグループ プロパティの詳細については、AWS Database Migration Service タスクのタスク設定の指定 のタスク設定 JSON ファイル例をご参照ください。これらのポリシーを に設定すると"SUSPEND_TABLE"、 AWS DMS タスクは、ポリシーを発生させたテーブルのデータエラーを停止し、すべてのテーブルに対してバッチモードで続行します。

BatchApplyEnabled パラメータを BatchApplyPreserveTransaction パラメータと併用することはできません。BatchApplyEnabled を true に設定すると、BatchApplyPreserveTransaction パラメータがトランザクションの整合性を確認します。

BatchApplyPreserveTransaction を true に設定すると、トランザクションの整合性が保持され、バッチにはソースのトランザクション内のすべての変更が確実に含まれます。

BatchApplyPreserveTransaction を false に設定すると、パフォーマンスを向上させるためにトランザクションの整合性が一時的に失われることがあります。

BatchApplyPreserveTransaction パラメータは、Oracle ターゲットエンドポイントにのみ適用され、BatchApplyEnabled パラメータが true に設定されている場合に限り、適切に機能します。

LOB 列がレプリケーションに含まれる場合、制限された LOB モードのみで BatchApplyEnabled を使用できます。

変更データ キャプチャ (CDC) のロードでこれらの設定を使用する方法の詳細については、「変更処理のチューニング設定」をご参照ください。

- MaxFullLoadSubTasks – 並列にロードするテーブルの最大数を指定します。デフォルトは 8、最大値は 49 です。

- ParallelLoadThreads – AWS DMS が各テーブルをターゲットデータベースにロードするために使用するスレッドの数を指定します。このパラメータには、非 RDBMS ターゲットの最大値があります。DynamoDB ターゲットの最大数は 200 です。Amazon Kinesis Data Streams、Apache Kafka、または Amazon OpenSearch Service ターゲットの最大値は 32 です。この上限を引き上げるように要請できます。ParallelLoadThreads はフルロードタスクに適用します。個々のテーブルの並列ロードを設定する詳細については、「テーブルとコレクション設定のルールとオペレーション」をご参照ください。

  この設定は、次のエンドポイントエンジンタイプに適用されます。

  - DynamoDB
  - Amazon Kinesis Data Streams
  - Amazon MSK
  - Amazon OpenSearch サービス
  - Amazon Redshift

  AWS DMS は、追加の接続属性として MySQL ParallelLoadThreadsの をサポートします。ParallelLoadThreadsはタスク設定として MySQL には適用されません。

- ParallelLoadBufferSize – 並列ロードスレッドがターゲットにデータをロードするために使用するバッファに保存するレコードの最大数を指定します。デフォルト値は 50 です。最大値は 1000 です。この設定は現在、DynamoDB 、Kinesis、Apache Kafka、または OpenSearch がターゲットの場合にのみ有効です。このパラメータを ParallelLoadThreads で使用します。1 つ以上のスレッドがある場合にのみ ParallelLoadBufferSize が有効になります。個々のテーブルの並列ロードを設定する詳細については、「テーブルとコレクション設定のルールとオペレーション」をご参照ください。

- ParallelLoadQueuesPerThread - 各同時スレッドがアクセスするキューの数を指定して、データレコードをキューから取り出し、ターゲットのバッチロードを生成します。デフォルト は 1 です。この設定は、現在、Kinesis または Apache Kafka がターゲットの場合にのみ有効です。

- ParallelApplyThreads — CDC ロード中に AWS DMS がデータレコードを Amazon DocumentDB 、Kinesis、Amazon MSK、 OpenSearchまたは Amazon Redshift ターゲットエンドポイントにプッシュするために使用する同時スレッドの数を指定します。デフォルト値は 0 です。

  この設定は CDC にのみ適用されます。この設定はフルロードには適用されません。

  この設定は、次のエンドポイントエンジンタイプに適用されます。

- Amazon DocumentDB (MongoDB 互換性)
- Amazon Kinesis Data Streams
- Amazon Managed Streaming for Apache Kafka
- Amazon OpenSearch サービス
- Amazon Redshift

- `ParallelApplyBufferSize` – CDC ロード中に Amazon DocumentDB 、Kinesis、Amazon MSK、 OpenSearchまたは Amazon Redshift ターゲットエンドポイントにプッシュする同時スレッドの各バッファキューに保存するレコードの最大数を指定します。デフォルト値は 100 です。最大値は 1000 です。このオプションは、`ParallelApplyThreads` で複数のスレッドを指定する場合に使用します。
- `ParallelApplyQueuesPerThread` – CDC 中に各スレッドがキューからデータレコードを取得し、Amazon DocumentDB 、Kinesis、Amazon MSK、または OpenSearchエンドポイントのバッチロードを生成するためにアクセスするキューの数を指定します。デフォルト値は 1 です。

全ロードタスク設定

全ロード設定には、以下のものが含まれます。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。

- 全ロードセットアップ時にターゲットのロードを処理する方法を指定するには、`TargetTablePrepMode` オプションに次のいずれかの値を指定します。
  - `DO_NOTHING` - 既存のターゲットテーブルのデータとメタデータには影響しません。
  - `DROP_AND_CREATE` - 既存のテーブルが削除され、新しいテーブルがその場所に作成されます。
  - `TRUNCATE_BEFORE_LOAD` - テーブルメタデータに影響を与えずにデータが切り捨てられます。
- 全ロードが完了するまでプライマリ キーや一意のインデックスの作成を遅らせるには、`CreatePkAfterFullLoad` オプションを `true` に設定します。
- 全ロードタスクと CDC が有効なタスクの場合、次の `Stop task after full load completes` のオプションを設定できます。
  - `StopTaskCachedChangesApplied` – このオプションを `true` に設定し、全ロードが完了してキャッシュされた変更が適用された後にタスクを停止します。
  - `StopTaskCachedChangesNotApplied` – キャッシュされた変更が適用される前にタスクを停止するには、このオプションを `true` に設定します。
- 並行してロードするテーブルの最大数を指定するには、`MaxFullLoadSubTasks` オプションを設定します。デフォルトは 8、最大値は 49 です。

- データレコードをターゲットエンドポイントにプッシュするために DMS がフルロードプロセス中に使用する同時スレッドの数を指定するには、ParallelLoadThreads を設定します。デフォルト値はゼロ (0) です。

> **Important**
>
> MaxFullLoadSubTasks は、並行してロードするテーブルまたはテーブルのセグメント数を制御します。ParallelLoadThreads は、ロードを並行して実行するために移行タスクが使用するスレッド数を制御します。上記の設定は乗算です。そのため、フルロードタスクで使用するスレッドの合計数は、ほぼ ParallelLoadThreads 値と MaxFullLoadSubTasks 値を乗算した値になります (ParallelLoadThreads * MaxFullLoadSubtasks))
> 多数のフルロードサブタスクがあり、多数の並列ロードスレッドがあるタスクを作成すると、タスクがメモリを大量に消費してエラーとなる可能性があります。

- 全ロードオペレーションを開始する前にトランザクションが閉じるのを AWS DMS 待機する秒数を設定できます。これを行うには、タスクの開始時にトランザクションが開いている場合は、TransactionConsistencyTimeout オプションを設定します。デフォルト値は 600 (10 分) です。 は、開いているトランザクションがある場合でも、タイムアウト値に達した後に全ロード AWS DMS を開始します。 full-load-only タスクは 10 分間待機せず、すぐに開始されます。
- 一括転送可能なレコードの最大数を指定するには、CommitRate オプションを設定します。デフォルト値は 10000 で、最大値は 50000 です。

Time Travel タスクの設定

レプリケーションタスクをログに記録してデバッグするには、 AWS DMS Time Travel を使用できます。この方法の場合、Amazon S3 を使用してログを保存し、暗号化キーを使用して暗号化します。Time Travel S3 バケットにアクセスできる場合にのみ、日時フィルターを使用して S3 ログを取得して、必要に応じてログを表示、ダウンロード、難読化できます。これにより、安全に「時間を遡って」データベースのアクティビティを調査できます。Time Travel は CloudWatch ログ記録とは独立して動作します。 CloudWatch ログ記録の詳細については、「」を参照してください[ロギングタスク設定](#)。

Time Travel は、 AWS DMSがサポートする Oracle、Microsoft SQL Server、PostgreSQL ソースエンドポイント、および AWS DMSがサポートする PostgreSQL と MySQL ターゲットエンドポイントを持つすべての AWS リージョンで使用できます。Time Travel は、フルロードと変更データキャプチャ (CDC) タスクと CDC のみのタスクでのみ有効にできます。Time Travel を有効にしたり、既存

の Time Travel 設定を変更したりするには、レプリケーションタスクが停止していることを確認します。

Time Travel 設定には、次の TTSettings プロパティがあります。

- EnableTT – このオプションを true に設定すると、タスクの Time Travel ログ記録が有効になります。デフォルト値は false です。

  タイプ: ブール

  必須：いいえ

- EncryptionMode – データとログを保存するために S3 バケットで使用されるサーバー側の暗号化のタイプ。"SSE_S3" (デフォルト) または "SSE_KMS" のいずれかを指定できます。

  EncryptionMode を "SSE_KMS" から "SSE_S3" に変更することはできても、その逆の変更はできません。

  タイプ: 文字列

  必須: いいえ

- ServerSideEncryptionKmsKeyId – "SSE_KMS"に を指定する場合はEncryptionMode、カスタムマネージド AWS KMS キーの ID を指定します。使用するキーに、 AWS Identity and Access Management (IAM) ユーザーのアクセス許可を有効にし、キーの使用を許可するポリシーがアタッチされていることを確認します。

  "SSE_KMS" オプションでは、独自のカスタマーマネージド 対称 KMS キーのみがサポートされます。

  型: 文字列

  必須: EncryptionMode が "SSE_KMS" に設定されている場合のみ

- ServiceAccessRoleArn – IAM ロールにアクセスするためにサービスが使用する Amazon リソースネーム (ARN)。ロール名は、dms-tt-s3-access-role に設定します。これは、 が S3 バケットからオブジェクト AWS DMS を書き込んだり読み取ったりできるようにする必須の設定です。

  型: 文字列

  必須: Time Travel がオンになっている場合

このロールのポリシーの例は次のとおりです。

```
{
 "Version": "2012-10-17",
 "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::S3bucketName*",
                "arn:aws:kms:us-east-1:112233445566:key/1234a1a1-1m2m-1z2z-
d1d2-12dmstt1234"
            ]
        }
    ]
}
```

このロールの信頼ポリシーの例は次のとおりです。

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

- BucketName – Time Travel ログを保存する S3 バケットの名前。Time Travel ログを有効にする前に、この S3 バケットが作成されていることを確認します。

  型: 文字列

  必須: Time Travel がオンになっている場合

- BucketFolder – S3 バケット内のフォルダ名を設定するためのオプションのパラメータ。のパラメータを指定すると、DMS は "/*BucketName*/*BucketFolder*/*taskARN*/*YYYY*/*MM*/*DD*/*hh*" のパスにTime Travel ログを作成します。このパラメータを指定しない場合、 はデフォルトパスを として AWS DMS 作成します"/*BucketName*/dms-time-travel-logs/*taskARN*/*YYYY*/*MM*/*DD*/*hh*。

  タイプ: 文字列

  必須: いいえ

- EnableDeletingFromS3OnTaskDelete – このオプションが に設定されている場合true、タスク AWS DMS が削除されると、 は S3 から Time Travel ログを削除します。デフォルト値は false です。

  タイプ: 文字列

  必須: いいえ

- EnableRawData – このオプションを true に設定すると、Time Travel ログのデータ操作言語 (DML) の raw データがTime Travel ログの raw_data 列の下に表示されます。詳細については、「Time Travel ログの使用」を参照してください。デフォルト値は、falseです。このオプションを false に設定すると、DML タイプのみがキャプチャされます。

  タイプ: 文字列

  必須: いいえ

- RawDataFormat – AWS DMS バージョン 3.5.0 以降では、 EnableRawDataが に設定されている場合true。このプロパティは、Time Travel ログ内の DML の raw データの形式を指定し、次のとおり表示します。

  - "TEXT" – CDC 中に Raw フィールドとしてキャプチャした DML イベントの、解析され、読み取り可能な列名と値。
  - "HEX" – CDC 中に DML イベントでキャプチャした列名と値の元の16 進数値。

  このプロパティは Oracle と Microsoft SQL Server データベースソースに適用されます。

タイプ: 文字列

必須: いいえ

- `OperationsToLog` – Time Travel ログに記録する DML オペレーションのタイプを指定します。以下のいずれかを指定できます。
  - `"INSERT"`
  - `"UPDATE"`
  - `"DELETE"`
  - `"COMMIT"`
  - `"ROLLBACK"`
  - `"ALL"`

  デフォルトは `"ALL"` です。

  タイプ: 文字列

  必須: いいえ
- `MaxRecordSize` – 各行に記録される Time Travel ログのレコードの最大サイズを指定します。このプロパティを使用して、特に使用頻度の高いテーブルの Time Travel ログの増大を制御します。デフォルトは 64 KB です。

  タイプ: 整数

  必須：いいえ

Time Travel ログを有効にして使用する方法の詳細については、次のトピックを参照してください。

トピック

- タスクの Time Travel ログを有効にする
- Time Travel ログの使用
- が Time Travel ログを S3 AWS DMS にアップロードする頻度

タスクの Time Travel ログを有効にする

前述の AWS DMS タスク設定を使用して、タスクの Time Travel を有効にできます。Time Travel を有効にする前に、レプリケーションタスクが停止していることを確認します。

を使用して Time Travel を有効にするには AWS CLI

1. DMS タスク設定 JSON ファイルを作成して、次のとおりの TTSettings セクションを追加します。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。

```
  .
  .
  .
     },
"TTSettings" : {
  "EnableTT" : true,
  "TTS3Settings": {
      "EncryptionMode": "SSE_KMS",
      "ServerSideEncryptionKmsKeyId": "arn:aws:kms:us-west-2:112233445566:key/
myKMSKey",
      "ServiceAccessRoleArn": "arn:aws:iam::112233445566:role/dms-tt-s3-access-
role",
      "BucketName": "myttbucket",
      "BucketFolder": "myttfolder",
      "EnableDeletingFromS3OnTaskDelete": false
    },
  "TTRecordSettings": {
      "EnableRawData" : true,
      "OperationsToLog": "DELETE,UPDATE",
      "MaxRecordSize": 64
    },
  .
  .
  .
```

2. 適切なタスクアクションで、--replication-task-settings オプションを使用してこの JSON ファイルを指定します。例えば、次の CLI コードフラグメントは、では、このTime Travel 設定ファイルを create-replication-task の一部で指定しています。

```
aws dms create-replication-task
--target-endpoint-arn arn:aws:dms:us-
east-1:112233445566:endpoint:ELS5O7YTYV452CAZR2EYBNQGILFHQIFVPWFRQAY \
--source-endpoint-arn arn:aws:dms:us-
east-1:112233445566:endpoint:HNX2BWIIN5ZYFF7F6UFFZVWTDFFSMTNOV2FTXZA \
```

```
--replication-instance-arn arn:aws:dms:us-
east-1:112233445566:rep:ERLHG2UA52EEJJKFYNYWRPCG6T7EPUAB5AWBUJQ \
--migration-type full-load-and-cdc --table-mappings 'file:///FilePath/
mappings.json' \
--replication-task-settings 'file:///FilePath/task-settings-tt-enabled.json' \
--replication-task-identifier test-task
                              .
                              .
                              .
```

この例での、Time Travel 設定ファイル名は `task-settings-tt-enabled.json` です。

同様に、このファイルを `modify-replication-task` アクションの一部として指定できます。

次のタスクアクションの Time Travel ログの特別な処理に注意します。

- `start-replication-task` – レプリケーションタスクを実行する際に、Time Travel に使用される S3 バケットにアクセスできない場合、タスクは `FAILED` としてマークされます。
- `stop-replication-task` – タスクが停止すると、 は、レプリケーションインスタンスで現在利用可能なすべての Time Travel ログを Time Travel に使用される S3 バケットに AWS DMS 直ちにプッシュします。

レプリケーションタスク実行中は、`EncryptionMode` 値を `"SSE_KMS"` から `"SSE_S3"` に変更できます。ただし、その逆の変更はできません。

実行中のタスクの Time Travel ログのサイズが 1 GB を超える場合、DMS はそのサイズに達してから 5 分以内にログを S3 にプッシュします。タスクの実行後、S3 バケットまたは KMS キーにアクセスできなくなると、DMS はこのバケットへのログのプッシュを停止します。ログが S3 バケットにプッシュされていない場合は、S3 と の AWS KMS アクセス許可を確認してください。DMS が上記のログを S3 にプッシュする頻度の詳細については、「が Time Travel ログを S3 AWS DMS にアップロードする頻度」を参照してください。

コンソールから既存のタスクの Time Travel を有効にするには、[タスク設定] の下にある JSON エディタオプションを使用して `TTSettings` セクションを追加します。

Time Travel ログの使用

Time Travel ログファイルは、次のフィールドを持つカンマ区切り値 (CSV) ファイルです。

```
log_timestamp
component
dms_source_code_location
transaction_id
event_id
event_timestamp
lsn/scn
primary_key
record_type
event_type
schema_name
table_name
statement
action
result
raw_data
```

Time Travel ログが S3 で利用できるようになったら、Amazon Athena などのツールを使用して直接アクセスし、クエリを実行できます。または、S3 から任意のファイルをダウンロードする場合と同様に、ログをダウンロードすることもできます。

mytable というテーブルのトランザクションがログ記録される Time Travel ログの例は次のとおりです。読みやすくするために、次のログには行末が追加されています。

```
"log_timestamp ","tt_record_type","dms_source_code_location ","transaction_id",
"event_id","event_timestamp","scn_lsn","primary_key","record_type","event_type",
"schema_name","table_name","statement","action","result","raw_data"
"2021-09-23T01:03:00:778230","SOURCE_CAPTURE","postgres_endpoint_wal_engine.c:00819",
"609284109","565612992","2021-09-23 01:03:00.765321+00","00000E9C/D53AB518","","DML",
"UPDATE (3)","dmstest","mytable","","Migrate","","table dmstest.mytable:
UPDATE: id[bigint]:2244937 phone_number[character varying]:'phone-number-482'
age[integer]:82 gender[character]:'f' isactive[character]:'true '
date_of_travel[timestamp without time zone]:'2021-09-23 01:03:00.76593'
description[text]:'TEST DATA TEST DATA TEST DATA TEST DATA'"
```

が Time Travel ログを S3 AWS DMS にアップロードする頻度

レプリケーション インスタンスのストレージ使用量を最小限に抑えるために、 は Time Travel ログを定期的に AWS DMS オフロードします。

Time Travel ログは、次の場合に Amazon S3 バケットにプッシュされます。

- ログの現在のサイズが 1 GB を超える場合、 は 5 分以内にログを S3 AWS DMS にアップロードします。したがって、 AWS DMS は実行中のタスクごとに S3 および に対して 1 時間あたり最大 12 AWS KMS 回の呼び出しを行うことができます。
- AWS DMS は、ログのサイズに関係なく、1 時間ごとにログを S3 にアップロードします。
- タスクが停止すると、 はタイムトラベルログを AWS DMS S3 にすぐにアップロードします。

ロギングタスク設定

ログ記録では CloudWatch 、移行プロセス中に Amazon を使用して情報をログに記録します。ロギングタスク設定を使用して、記録するコンポーネントアクティビティと、ログに書き込まれる情報量を指定できます。ログ記録タスク設定は JSON ファイルに書き込まれます。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。

CloudWatch ログ記録は、いくつかの方法で有効にできます。移行タスクを作成する AWS Management Console ときに、 で EnableLoggingオプションを選択できます。または、 AWS DMS API を使用してタスクを作成するtrueときに、 EnableLoggingオプションを に設定することもできます。さらに、タスク設定のロギングセクションの JSON で "EnableLogging": true を指定することもできます。

EnableLogging を に設定するとtrue、 は CloudWatch グループ名とストリーム名を次のように AWS DMS 割り当てます。上記の値を直接設定することはできません。

- CloudWatchLogGroup: dms-tasks-<REPLICATION_INSTANCE_IDENTIFIER>
- CloudWatchLogStream: dms-task-<REPLICATION_TASK_EXTERNAL_RESOURCE_ID>

<REPLICATION_INSTANCE_IDENTIFIER> は、レプリケーションインスタンスの識別子です。<REPLICATION_TASK_EXTERNAL_RESOURCE_ID> は、タスク ARN の <resourcename> セクションの値です。がリソース ARN AWS DMS を生成する方法については、「」を参照してくださいの Amazon リソースネーム (ARN) の構築 AWS DMS。 ARNs

CloudWatch は AWS Identity and Access Management (IAM) と統合され、 AWS アカウントのユーザーが実行できる CloudWatchアクションを指定できます。での IAM の使用の詳細については CloudWatch、「Amazon ユーザーガイド」の「Amazon の Identity and Access Management CloudWatch」および「Amazon CloudWatch API コールのログ記録 CloudWatch 」を参照してください。

タスクログを削除するには、タスク設定のログ記録セクションの JSON で DeleteTaskLogs を true に設定します。

次のタイプのイベントのログ記録を指定できます。

- FILE_FACTORY – ファイルファクトリは、バッチ適用とバッチのロードに使用されるファイルを管理し、Amazon S3 エンドポイントを管理します。
- METADATA_MANAGER – メタデータマネージャは、レプリケーション中のソースとターゲットのメタデータ、パーティション分割、テーブルのステータスを管理します。
- SORTER – SORTER は、SOURCE_CAPTURE プロセスから受信イベントを受け取ります。イベントはトランザクションでバッチ処理され、TARGET_APPLY サービスコンポーネントに渡されます。SOURCE_CAPTURE プロセスが、TARGET_APPLY コンポーネントが処理できるよりも速くイベントを生成した場合、SORTER コンポーネントはバックログのイベントをディスクまたはスワップファイルにキャッシュします。キャッシュされたイベントは、レプリケーションインスタンスのストレージ不足を引き起こすよくある原因となります。

  SORTER サービスコンポーネントは、キャッシュされたイベントの管理、CDC 統計の収集、タスクのレイテンシーの報告を行います。
- SOURCE_CAPTURE – 継続的なレプリケーション (CDC) データがソースデータベースまたはサービスからキャプチャされ、SORTER サービス コンポーネントに渡されます。
- SOURCE_UNLOAD – フルロード中にソースデータベースまたはサービスからデータがアンロードされます。
- TABLES_MANAGER — テーブルマネージャは、キャプチャされたテーブルを追跡し、テーブルの移行順序を管理し、テーブル統計を収集します。
- TARGET_APPLY - データおよびデータ定義言語 (DDL) ステートメントがターゲットデータベースに適用されます。
- TARGET_LOAD — データはターゲットデータベースにロードされます。
- TASK_MANAGER – タスクマネージャーは実行中のタスクを管理し、並列データ処理に向けてタスクをサブタスクに分割します。
- TRANSFORMATION – テーブルマッピング変換イベント。詳細については、「テーブルマッピングを使用して、タスクの設定を指定する」を参照してください。
- VALIDATOR/ VALIDATOR_EXT – VALIDATOR サービスコンポーネントは、データがソースからターゲットに正確に移行されたことを検証します。詳細については、「[Data validation] (データ検証)」を参照してください。

次のログ記録コンポーネントは、LOGGER_SEVERITY_DETAILED_DEBUG ログの重大度レベルを使用すると大量のログを生成します。

- COMMON
- ADDONS
- DATA_STRUCTURE
- COMMUNICATION
- FILE_TRANSFER
- FILE_FACTORY

これらのコンポーネントでは、トラブルシューティング時に DEFAULT 以外のログレベルが必要になることは、ほぼありません。 AWS Support から特に要求されない限り、これらのコンポーネントのログ記録レベルDEFAULTを から変更することはお勧めしません。

上記の 1 つを指定した後、次のリストに示すように、ログに記録される情報の量を指定できます。

緊急度のレベルは、情報の最低レベルから最高レベルの順に並んでいます。高いレベルには、必ず低いレベルの情報が含まれています。

- LOGGER_SEVERITY_ERROR(ロガーエラー重度) - エラー メッセージがログに書き込まれます。
- LOGGER_SEVERITY_WARNING(ロガーエラー警告) - 警告とエラー メッセージがログに書き込まれます。
- LOGGER_SEVERITY_INFO(ロガーエラー情報) - 情報メッセージ、警告、エラー メッセージがログに書き込まれます。
- LOGGER_SEVERITY_DEFAULT(ロガーデフォルト重度) - 情報メッセージ、警告、エラー メッセージがログに書き込まれます。
- LOGGER_SEVERITY_DEBUG(ロガーデバッグ重度) - デバッグ メッセージ、情報メッセージ、警告、エラーメッセージがログに書き込まれます。
- LOGGER_SEVERITY_DETAILED_DEBUG(ロガー詳細デバッグ重度) - すべての情報がログに書き込まれます。

次の JSON の例は、すべてのアクションと緊急度のレベルを記録するためのタスク設定を示しています。

```
…
  "Logging": {
    "EnableLogging": true,
    "LogComponents": [
      {
```

```
        "Id": "FILE_FACTORY",
        "Severity": "LOGGER_SEVERITY_DEFAULT"
      },{
        "Id": "METADATA_MANAGER",
        "Severity": "LOGGER_SEVERITY_DEFAULT"
      },{
        "Id": "SORTER",
        "Severity": "LOGGER_SEVERITY_DEFAULT"
      },{
        "Id": "SOURCE_CAPTURE",
        "Severity": "LOGGER_SEVERITY_DEFAULT"
      },{
        "Id": "SOURCE_UNLOAD",
        "Severity": "LOGGER_SEVERITY_DEFAULT"
      },{
        "Id": "TABLES_MANAGER",
        "Severity": "LOGGER_SEVERITY_DEFAULT"
      },{
        "Id": "TARGET_APPLY",
        "Severity": "LOGGER_SEVERITY_DEFAULT"
      },{
        "Id": "TARGET_LOAD",
        "Severity": "LOGGER_SEVERITY_INFO"
      },{
        "Id": "TASK_MANAGER",
        "Severity": "LOGGER_SEVERITY_DEBUG"
      },{
        "Id": "TRANSFORMATION",
        "Severity": "LOGGER_SEVERITY_DEBUG"
      },{
        "Id": "VALIDATOR",
        "Severity": "LOGGER_SEVERITY_DEFAULT"
      }
    ],
    "CloudWatchLogGroup": null,
    "CloudWatchLogStream": null
  },
…
```

制御テーブルタスク設定

コントロールテーブルは、 AWS DMS タスクに関する情報を提供します。また、現在の移行タスクと今後のタスクの両方を計画および管理するために使用できる有益な統計情報も提供します。

これらのタスク設定は、JSON ファイルで適用することも、 AWS DMS コンソールのタスクの作成ページで詳細設定を選択して適用することもできます。[Apply Exceptions] (例外を適用)] テーブル (dmslogs.awsdms_apply_exceptions) は常にデータベース ターゲットで作成されます。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。

AWS DMS は、フルロード + CDC または CDC のみのタスク中にのみコントロールテーブルを作成し、フルロードのみのタスク中には作成しません。

全ロードタスクと CDC（既存データの移行あらに進行中変更のレプリケーション）タスクと CDC のみ（データ変更のみレプリケーション）タスクでは、以下の内容を含むテーブルを追加することもできます：

- [Replication Status (dmslogs.awsdms_status)] (レプリケーションステータス (dmslogs.awsdms_status)) - このテーブルは、現在のタスクに関する詳細を提供します。これには、タスクステータス、タスクにより消費されるメモリの量、まだターゲットに適用されていない変更の数が含まれます。この表には、 が現在読み取り中のソースデータベース内の位置も示 AWS DMS されています。タスクの全ロードフェーズあるいはデータ キャプチャの変更 (CDC) についても表示します。
- [Suspended Tables (dmslogs.awsdms_suspended_tables)] (停止済みテーブル (dmslogs.awsdms_suspended_tables)) - このテーブルは、停止済みテーブルのリストと、停止理由を示します。
- [Replication History (dmslogs.awsdms_history)] (レプリケーション履歴 (dmslogs.awsdms_history)) - このテーブルは、レプリケーション履歴に関する情報を提供します。この情報には、タスク中に処理されたレコードの数とボリューム、CDC タスク終了時のレイテンシー、およびその他の統計情報などが含まれています。

例外適用テーブル dmslogs.awsdms_apply_exceptions には、以下のパラメータが含まれます。

列	型	説明
TASK_NAME	nvchar	AWS DMS タスクのリソース ID。リソース ID は タスク ARN で確認できる。
TABLE_OWNER	nvchar	テーブルの所有者。

列	型	説明
TABLE_NAME	nvchar	テーブルの名前。
ERROR_TIME	timestamp	時間例外 (エラー) が発生しました。
STATEMENT	nvchar	エラーが発生したときに実行されたステートメント。
ERROR	nvchar	エラーの名前と説明。

レプリケーションステータステーブル (dmslogs.awsdms_status) には、タスクの現在のステータスとターゲットのデータベースが含まれます。これには、次の設定があります。

列	型	説明
SERVER_NAME	nvchar	レプリケーションタスクを実行しているマシンの名前。
TASK_NAME	nvchar	AWS DMS タスクのリソース ID。リソース ID は タスク ARN で確認できる。
TASK_STATUS	varchar	次のいずれかの値になります。 • FULL LOAD • CHANGE PROCESSING (CDC) • 実行中ではない 全ロードされているテーブルが少なくとも 1 つある限り、タスクのステータスを FULL LOAD と設定します。CDC が有効化されている場合、すべてのテーブルがロードされた後、タスクステータスは CHANGE PROCESSING に変

列	型	説明
		更します。タスクは、タスクを開始する前、またはタスクの完了後に、NOT RUNNING に設定されます。
STATUS_TIME	timestamp	タスクの状態のタイムスタンプ。
PENDING_CHANGES	整数	ソースデータベースでコミットされ、レプリケーションインスタンスのメモリとディスクにキャッシュされた変更レコードの数
DISK_SWAP_SIZE	整数	古い、またはオフロードされたトランザクションにより使用されるディスク領域の量。
TASK_MEMORY	整数	使用されている現在のメモリを MB で表示。
SOURCE_CURRENT_POSITION	varchar	AWS DMS 現在読み取り元のソースデータベース内の位置。
SOURCE_CURRENT_TIMESTAMP	timestamp	AWS DMS 現在読み取り元のソースデータベースのタイムスタンプ。
SOURCE_TAIL_POSITION	varchar	最も以前に起動された完了していないトランザクションの位置。この値は、すべての変更を失わずに返すことができる最も新しい位置を示します。

列	型	説明
SOURCE_TAIL _TIMESTAMP	timestamp	最も以前に起動された完了していないトランザクションのタイムスタンプ。この値は、すべての変更を失わずに返すことができる最も新しいタイムスタンプを示します。
SOURCE_TI MESTAMP _APPLIED	timestamp	最新の完了したトランザクションのタイムスタンプ。一括適用のプロセスでは、この値はバッチ内の最新トランザクション完了のタイムスタンプの値を示します。

停止したテーブル (dmslogs.awsdms_suspended_tables) には次のパラメータがあります。

列	型	説明
SERVER_NAME	nvchar	レプリケーションタスクを実行しているマシンの名前。
TASK_NAME	nvchar	AWS DMS タスクの名前
TABLE_OWNER	nvchar	テーブルの所有者。
TABLE_NAME	nvchar	テーブルの名前。
SUSPEND_REASON	nvchar	停止理由
SUSPEND_T IMESTAMP	timestamp	停止が実行された時刻

レプリケーション履歴テーブル (dmslogs.awsdms_history) には、以下のパラメータが含まれます。

列	型	説明
SERVER_NAME	nvchar	レプリケーションタスクを実行しているマシンの名前。
TASK_NAME	nvchar	AWS DMS タスクのリソース ID。リソース ID は タスク ARN で確認できる。
TIMESLOT_TYPE	varchar	次のいずれかの値になります。 • FULL LOAD • CHANGE PROCESSING (CDC) タスクが全ロードと CDC の両方を実行している場合、2 つの履歴レコードがタイムスロットに書き込まれます。
TIMESLOT	timestamp	タイムスロットのタイムスタンプ終了。
TIMESLOT_DURATION	整数	タイムスロットの時間 (分単位)。
TIMESLOT_LATENCY	整数	タイムスロットの終了時のでターゲットレイテンシー (秒単位)。この値は CDC タイムスロットにのみ適用されます。
RECORDS	整数	タイムスロット中に処理されるレコード数。
TIMESLOT_VOLUME	整数	処理されるデータ量を MB で 表示。

[Validation Failure](検証失敗)テーブル (awsdms_validation_failures_v1) には、タスクのすべてのデータ検証の障害が含まれます。詳細については、「データ検証のトラブルシューティング」をご参照ください。

追加の制御テーブル設定には、以下のものが含まれます。

- HistoryTimeslotInMinutes - [Replication History](レプリケーション履歴) テーブルにおける各タイムスロットの長さを指定するには、このオプションを使用します。デフォルトは 5 分です。
- ControlSchema – このオプションを使用して、 AWS DMS ターゲットのコントロールテーブルのデータベーススキーマ名を指定します。このオプションに情報を入力しない場合、テーブルは次の表のとおりデータベースのデフォルトの場所にコピーされます。
  - PostgreSQL、Public
  - Oracle、ターゲットスキーマ
  - Microsoft SQL Server、ターゲットデータベースの dbo
  - MySQL、awsdms_control
  - MariaDB、awsdms_control
  - Amazon Redshift、公開
  - DynamoDB、データベースに個別のテーブルとして作成
  - IBM Db2 LUW、awsdms_control

ストリームバッファタスク設定

ストリームバッファの設定は AWS CLI、以下を含む を使用して設定できます。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。

- StreamBufferCount - 移行タスクのデータストリームバッファの数を指定するには、このオプションを指定します。デフォルトストリームバッファの数は 3 です。この設定の値を大きくすると、データ抽出速度が上昇する可能性があります。ただし、このパフォーマンス向上は、レプリケーションサーバーのソースシステムやインスタンスクラスなど、移行環境に大きく依存します。ほとんどの場合はデフォルトで十分です。
- StreamBufferSizeInMB - 各データストリームバッファの最大サイズを指定するには、このオプションを使用します。デフォルトサイズは 8 MB です。非常に大きい LOB を使用する場合、このオプションの値を大きくする必要がある場合があります。また、ストリームバッファサイズが不十分であることを示すメッセージがログファイルに記録されている場合も、この値を大きくする必要がある可能性があります。このオプションのサイズ

を計算するときは、 [Max LOB size (or LOB chunk size)]*[number of LOB columns]*[number of stream buffers]*[number of tables loading in parallel per task(MaxFullLoadSubTasks)]*3 式を使用できます。

- CtrlStreamBufferSizeInMB - 制御ストリーム バッファのサイズを設定するには、このオプションを使用します。値は MB 単位で、1 ~ 8 が使用できます。デフォルト値は 5 です。かなり多くのテーブル (数万のテーブルなど) を使用している場合、状況によってはこの値を大きくする必要があります。

変更処理のチューニング設定

次の設定は、変更データキャプチャ (CDC) 中にターゲットテーブルの変更を AWS DMS が処理する方法を決定します。これらの設定のいくつかは、ターゲットメタデータパラメータ BatchApplyEnabled の値によって異なります。BatchApplyEnabled パラメータの詳細については、「ターゲットメタデータのタスク設定」をご参照ください。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。

変更処理のチューニング設定には、以下のものが含まれます。

以下の設定は、ターゲットメタデータパラメータ BatchApplyEnabled を true に設定している場合にのみ適用されます。

- BatchApplyPreserveTransaction – true に設定すると、トランザクションの整合性が保持され、必ずバッチにはソースからのトランザクション内のすべての変更が含まれます。デフォルト値は、trueです。この設定は、Oracle ターゲットエンドポイントにのみ適用されます。

  false に設定すると、パフォーマンスを向上させるためにトランザクションの整合性が一時的に失われることがあります。ソースからのトランザクション内のすべての変更が 1 バッチでターゲットに適用されるとは限りません。

  デフォルトでは、 はトランザクションモードで変更 AWS DMS を処理し、トランザクションの整合性を維持します。トランザクションの完全性を一時的に失効できる場合は、代わりに [最適化バッチ] オプションを使用できます。このオプションでは、効率的にトランザクションがグループ化され、効率化のためにバッチに適用します。バッチ最適化適用オプションを使用すると、ほとんどの場合、参照整合性の制約に違反します。そのため、移行プロセス中にこれらの制約をオフにし、カットオーバー プロセスの一環として再度オンにすることをお勧めします。
- BatchApplyTimeoutMin – バッチ変更の各アプリケーション間で AWS DMS 待機する最小時間を秒単位で設定します。デフォルト値は 1 です。

- BatchApplyTimeoutMax – バッチ変更の各アプリケーション間でタイムアウトするまでに AWS DMS 待機する最大時間を秒単位で設定します。デフォルト値は 30 です。
- BatchApplyMemoryLimit – [Batch optimized apply mode] (最適化バッチ適用モード) での前処理に使用されるメモリの最大量 (MB) を設定します。デフォルト値は 500 です。
- BatchSplitSize - 1 つのバッチに適用される変更の最大数を設定します。デフォルト値 0 は、適用される制限がないことを意味します。

以下の設定は、ターゲットメタデータパラメータ BatchApplyEnabled を false に設定している場合にのみ適用されます。

- MinTransactionSize - 各トランザクションに含める変更の最小数を設定します。デフォルト値は 1000 です。
- CommitTimeout – タイムアウトを宣言する前に がトランザクションをバッチで収集 AWS DMS するための最大時間を秒単位で設定します。デフォルト値は 1 です。

双方向レプリケーションの場合、次の設定は、ターゲットメタデータパラメータ BatchApplyEnabled を false に設定している場合にのみ適用されます。

- LoopbackPreventionSettings – これらの設定により、双方向レプリケーションに関係するタスクのペアで進行中の各レプリケーション タスクのループバックを阻止します。ループバック防止は、双方向レプリケーションの両方向で同一の変更が適用されてデータが破損するのを防ぎます。双方向レプリケーションの詳細については、「双方向レプリケーションの実行」をご参照ください。

AWS DMS は、トランザクションがソース、ターゲット、またはその両方に完全にコミットされるまで、トランザクションデータをメモリに保持しようとします。ただし、割り当てたメモリより大きいトランザクションや、指定した制限時間内にコミットされないトランザクションは、ディスクに書き込まれます。

以下の設定は、変更処理のモードに関係なく、変更処理のチューニングに適用されます。

- MemoryLimitTotal - すべてのトランザクションがディスクに書き込まれるまでにメモリで占有可能な最大サイズ (MB) を設定します。デフォルト値は 1024 です。
- MemoryKeepTime - 各トランザクションがディスクに書き込まれるまでにメモリに保持できる最長時間 (秒) を設定します。期間は、 がトランザクションのキャプチャ AWS DMS を開始した時刻から計算されます。デフォルト値は 60 です。

- StatementCacheSize - ターゲットに変更を適用するときに、後で実行するためにサーバーに保存する準備済みステートメントの最大数を設定します。デフォルト値は 50 です。最大値は 200 です。

処理チューニング設定の変更のタスク設定がタスク設定の JSON ファイルでどのように記述されるかを示す例は、次のとおりです。

```
"ChangeProcessingTuning": {
        "BatchApplyPreserveTransaction": true,
        "BatchApplyTimeoutMin": 1,
        "BatchApplyTimeoutMax": 30,
        "BatchApplyMemoryLimit": 500,
        "BatchSplitSize": 0,
        "MinTransactionSize": 1000,
        "CommitTimeout": 1,
        "MemoryLimitTotal": 1024,
        "MemoryKeepTime": 60,
        "StatementCacheSize": 50
}
```

データ レプリケーション タスク中に Amazon S3 ターゲットへの書き込みの頻度を制御するには、cdcMaxBatchInterval と cdcMinFileSize の追加接続属性を設定することができます。これにより、追加のオーバーヘッド オペレーションなしでデータを分析する際のパフォーマンスが向上します。詳細については、「AWS DMS のターゲットとして Amazon S3を使用する場合のエンドポイントの設定」をご参照ください。

データ検証タスクの設定

データがソースからターゲットに正確に移行されたことを確認できます。タスクの検証を有効にすると、 は、テーブルに対して全ロードが実行された直後に、ソースデータとターゲットデータの比較 AWS DMS を開始します。タスクのデータ検証の詳細、要件、データベースサポートのスコープ、レポートするメトリクスについては、「AWS DMS データ検証」をご参照ください。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。

データの検証設定およびその値には、以下のものが含まれます。

- EnableValidation - true に設定すると、データの検証を有効にします。それ以外の場合は、タスクの検証が無効になります。デフォルト値は false です。

- ValidationMode – DMS がソーステーブルに対してターゲットテーブルのデータを検証する方法を制御します。 AWS DMS では、今後の拡張に向けてこの設定を提供しています。現在、デフォルトかつ唯一の有効な値は ですROW_LEVEL。 は、ソーステーブルとターゲットテーブル間のすべての行 AWS DMS を検証します。
- FailureMaxCount - タスク検証が停止する前に、検証を失敗できるレコードの最大数を指定します。デフォルト値は 10,000 です。検証に失敗するレコードの数に関係なく検証を継続するには、この値をソースのレコード数より大きく設定します。
- HandleCollationDiff - このオプションを true に設定すると、検証で比較するソースレコードとターゲットレコードを識別する際に、PostgreSQL と SQL Server のエンドポイントでの列照合の違いが考慮されます。それ以外の場合は、このような列照合の違いは検証で無視されます。列の照合は行の順序を指定でき、データ検証にとってはとても重要となります。HandleCollationDiff を true に設定すると、この照合の違いを自動的に解決し、データ検証における誤検出を防ぎます。デフォルト値は、falseです。
- RecordFailureDelayInMinutes – 検証障害の詳細を報告する前に遅延を分単位で指定します。
- RecordFailureDelayLimitInMinutes - 検証障害の詳細を報告する前に遅延を指定します。通常の場合、 AWS DMS はターゲットに変更行う際の実際の遅延を認識するタスクレイテンシーを使用して、誤検出を防ぎます。この設定は実際の遅延値を上書きし、すべての検証メトリクスを報告する前のより長い遅延の設定を有効にします。デフォルト値は 0 です。
- RecordSuspendDelayInMinutes - FailureMaxCount で設定されたエラーしきい値のため、テーブル検証が中断されるまでの遅延時間を分単位で指定します。
- SkipLobColumns – このオプションを に設定する AWS DMS とtrue、 はタスク検証のテーブルの 部分内のすべての LOB 列のデータ検証をスキップします。デフォルト値は、falseです。
- TableFailureMaxCount – テーブル検証が停止する前に、検証を失敗できるテーブルあたりの最大行数を指定します。デフォルト値は 1,000 です。
- ThreadCount – 検証中に が AWS DMS 使用する実行スレッドの数を指定します。各スレッドは、ソースとターゲットから not-yet-validated データを選択して比較および検証します。デフォルト値は 5 です。を高い数値ThreadCountに設定すると、 AWS DMS は検証をより速く完了できます。ただし、この場合、 AWS DMS はより多くの同時クエリを実行し、ソースとターゲットでより多くのリソースを消費します。
- ValidationOnly – このオプションを true に設定すると、タスクはデータの移行やレプリケーションを実行せずにデータ検証を実行します。デフォルト値は、falseです。タスク作成後に ValidationOnly 設定を変更することはできません。

TargetTablePrepMode を DO_NOTHING (検証のみのタスクのデフォルト) に設定し、移行タイプを次のいずれかに設定する必要があります。

- フルロード — AWS DMS コンソールで既存のデータを移行するようにタスク移行タイプを設定します。または、 AWS DMS API で移行タイプを FULL-LOAD に設定します。
- CDC — AWS DMS コンソールで、タスクの [移行タイプ] を [データ変更のみをレプリケートする] に設定します。または、 AWS DMS API で移行タイプを CDC に設定します。

選択した移行タイプを問わず、検証のみのタスクではデータが実際に移行またはレプリケートされることはありません。

詳細については、「検証のみのタスク」を参照してください。

> **Important**
>
> ValidationOnly の設定はイミュータブルです。タスクの作成後、タスクに対する変更はできません。

- ValidationPartialLobSize - 列に保存されているすべてのデータを検証するのではなく、LOB 列の部分検証を実行するかどうかを指定します。これは、LOB データセット全体ではなく LOB データの一部のみを移行する場合に便利な機能です。この値は、KB 単位です。デフォルト値は 0 で、 AWS DMS はすべての LOB 列データを検証します。例えば、 は、ソースとターゲットの両方の列データの最初の 32KB AWS DMS のみを検証する"ValidationPartialLobSize": 32ことを意味します。
- PartitionSize -ソースとターゲットの両方から比較noために読み取るレコードのバッチサイズを指定します。デフォルトは 10,000 です。
- ValidationQueryCdcDelaySeconds — CDC 更新ごとに、ソースとターゲットの両方で最初の検証クエリが遅延する時間。これにより、移行のレイテンシーが高い場合にリソースの競合が軽減される可能性があります。検証のみのタスクでは、このオプションは自動的に 180 秒に設定されます。デフォルトは 0 です。

たとえば、以下の JSON ではスレッドのデフォルト数の 2 倍のデータ検証を有効化しています。また、ここでは PostgreSQL エンドポイントでの列照合の違いによって生じるレコード順序の相違も考慮されます。また、すべての検証失敗を処理する追加の時間を考慮に入れた検証報告遅延を提供します。

```
"ValidationSettings": {
     "EnableValidation": true,
     "ThreadCount": 10,
     "HandleCollationDiff": true,
     "RecordFailureDelayLimitInMinutes": 30
  }
```

> **Note**
>
> Oracle エンドポイントの場合、 は DBMS_CRYPTO AWS DMS を使用して BLOBs。Oracle エンドポイントで BLOB を使用する場合は、Oracle エンドポイントにアクセスするために使用されるユーザーアカウントに DBMS_CRYPTO での execute 許可を付与します。これを行うには、以下のステートメントを実行します。
>
> ```
> grant execute on sys.dbms_crypto to dms_endpoint_user;
> ```

変更処理の DDL 処理のタスク設定

以下の設定は、変更データキャプチャ (CDC) 中にターゲットテーブルのデータ定義言語 (DDL) の変更を が AWS DMS 処理する方法を決定します。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。

変更処理の DDL を処理するタスク設定には、次のものがあります。

- HandleSourceTableDropped – ソーステーブルが削除されたときにターゲットテーブルを削除するには、このオプションを true に設定します。
- HandleSourceTableTruncatedソーステーブルが切り捨てられたときにターゲットテーブルを切り捨てるには、このオプションを true に設定します。
- HandleSourceTableAltered – ソーステーブルが変更されたときにターゲットテーブルを変更するには、このオプションを true に設定します。

次に、変更処理の DDL を処理するタスク設定が、タスク設定 JSON ファイルにどのように表示されるかを示します。

```
            "ChangeProcessingDdlHandlingPolicy": {
               "HandleSourceTableDropped": true,
               "HandleSourceTableTruncated": true,
               "HandleSourceTableAltered": true
            },
```

> **Note**
>
> 特定のソースでサポートされている DDL ステートメントについては、そのソースについて説明しているトピックをご参照ください。

文字置換タスクの設定

レプリケーションタスクが、 STRINGまたは WSTRING データ型のすべてのソースデータベース列のターゲットデータベースで文字置換を実行するように AWS DMS 指定できます。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。

次のソースデータベースとターゲットデータベースからのエンドポイントを持つ任意のタスクの文字置換を設定できます。

- ソースデータベース。
  - Oracle
  - Microsoft SQL Server
  - MySQL
  - PostgreSQL
  - SAP Adaptive Server Enterprise (ASE)
  - IBM Db2 LUW
- ターゲットデータベース:
  - Oracle
  - Microsoft SQL Server
  - MySQL
  - PostgreSQL
  - SAP Adaptive Server Enterprise (ASE)
  - Amazon Redshift

タスク設定の CharacterSetSettings パラメータを使用して、文字置換を指定できます。これらの文字置換は、16 進数表記の Unicode コードポイント値を使用して指定された文字に対して発生します。両方が指定されている場合は、2 つのフェーズで置換を実装できます。

1. 個々の文字置換 — ソース上の選択した文字の値を、ターゲット上の対応する文字の指定された置換値に置き換え AWS DMS ることができます。CharacterSetSettings の CharacterReplacements 配列を使用して、指定した Unicode コードポイントを持つすべてのソース文字を選択します。また、この配列を使用して、ターゲットの対応する文字の置換コードポイントを指定します。

   特定のコードポイントを持つソースのすべての文字を選択するには、CharacterReplacements 配列の SourceCharacterCodePoint のインスタンスをそのコードポイントに設定します。次に、この配列で TargetCharacterCodePoint の対応するインスタンスを設定することで、すべての同等のターゲット文字の置換コードポイントを指定します。ターゲットキャラクターを置き換えるのではなく削除するには、TargetCharacterCodePoint の適切なインスタンスをゼロ (0) に設定します。CharacterReplacements 配列で SourceCharacterCodePoint 設定と TargetCharacterCodePoint 設定の追加のペアを指定することで、必要な数のターゲットキャラクターの値を置換または削除できます。SourceCharacterCodePoint の複数のインスタンスに同じ値を指定した場合、対応する最後の TargetCharacterCodePoint の設定の値がターゲットに適用されます。

   例えば、CharacterReplacements に次の値を指定するとします。

```
"CharacterSetSettings": {
    "CharacterReplacements": [ {
        "SourceCharacterCodePoint": 62,
        "TargetCharacterCodePoint": 61
        }, {
        "SourceCharacterCodePoint": 42,
        "TargetCharacterCodePoint": 41
        }
    ]
}
```

   この例では、 は、すべての文字をターゲットのソースコードポイント 16 進値 62 に、コードポイント値 61 の文字 AWS DMS に置き換えます。また、 は、すべての文字をターゲットのソースコードポイント 42 に、コードポイント値 41 の文字 AWS DMS に置き換えます。つまり、 AWS DMS は、ターゲット上の文字 'b' のすべてのインスタンスを文字 'a' で置き換えます。同様に、 AWS DMS はターゲット'B'上の文字のすべてのインスタンスを文字 に置き換えます'A'。

2. 文字セットの検証と置換 — 個々の文字置換が完了したら、指定した 1 文字セットにすべてのターゲット文字の有効な Unicode コードポイントがあることを確認 AWS DMS できます。CharacterSetSettings で CharacterSetSupport を使用して、このターゲットキャラクターの検証と変更を設定します。検証文字セットを指定するには、CharacterSetSupport で CharacterSet を文字セットの文字列値に設定します。(指定できる値については、CharacterSetを参照してください) 次のいずれかの方法で、無効なターゲット文字を に AWS DMS 変更できます。

   - 現在のコードポイントに関係なく、すべての無効なターゲット文字に対して 1 つの置換 Unicode コードポイントを指定します。この置換コードポイントを設定するには、CharacterSetSupport の ReplaceWithCharacterCodePoint を指定された値に設定します。
   - ReplaceWithCharacterCodePoint をゼロ (0) に設定して、すべての無効なターゲット文字の削除を設定します。

   例えば、CharacterSetSupport に次の値を指定するとします。

```
"CharacterSetSettings": {
    "CharacterSetSupport": {
        "CharacterSet": "UTF16_PlatformEndian",
        "ReplaceWithCharacterCodePoint": 0
    }
}
```

   この例では、 は、ターゲットで見つかった文字のうち、文字セットで無効な"UTF16_PlatformEndian"文字をすべて AWS DMS 削除します。したがって、16 進値で指定された文字 2FB6 はすべて削除されます。この値は 4 バイトの Unicode コードポイントであり、UTF16 文字セットが 2 バイトのコードポイントを持つ文字のみを受け付けるため、無効です。

> **Note**
>
> レプリケーションタスクは、テーブルマッピングで指定したグローバルまたはテーブルレベルの変換を開始する前に、指定されたすべての文字置換を完了します。テーブルマッピングの詳細については、「テーブルマッピングを使用して、タスクの設定を指定する」をご参照ください。
>
> 文字置換では LOB データ型をサポートしていません。これには、DMS が LOB データ型と見なすすべてのデータ型が含まれます。例えば、Oracle の Extended データ型は LOB と見

なされます。ソースデータ型の詳細については、次の「Oracle のソースデータ型」を参照してください。

が に対して AWS DMS サポートする値は、次の表CharacterSetのとおりです。

UTF-8	ibm-860_P100-1995	ibm-280_P100-1995
UTF-16	ibm-861_P100-1995	ibm-284_P100-1995
UTF-16BE	ibm-862_P100-1995	ibm-285_P100-1995
UTF-16LE	ibm-863_P100-1995	ibm-290_P100-1995
UTF-32	ibm-864_X110-1999	ibm-297_P100-1995
UTF-32BE	ibm-865_P100-1995	ibm-420_X120-1999
UTF-32LE	ibm-866_P100-1995	ibm-424_P100-1995
UTF16_PlatformEndian	ibm-867_P100-1998	ibm-500_P100-1995
UTF16_OppositeEndian	ibm-868_P100-1995	ibm-803_P100-1999
UTF32_PlatformEndian	ibm-869_P100-1995	ibm-838_P100-1995
UTF32_OppositeEndian	ibm-878_P100-1996	ibm-870_P100-1995
UTF-16BE,version=1	ibm-901_P100-1999	ibm-871_P100-1995
UTF-16LE,version=1	ibm-902_P100-1999	ibm-875_P100-1995
UTF-16,version=1	ibm-922_P100-1999	ibm-918_P100-1995
UTF-16,version=2	ibm-1168_P100-2002	ibm-930_P120-1999
UTF-7	ibm-4909_P100-1999	ibm-933_P110-1995
IMAP-mailbox-name	ibm-5346_P100-1998	ibm-935_P110-1999
SCSU	ibm-5347_P100-1998	ibm-937_P110-1999

BOCU-1	ibm-5348_P100-1997	ibm-939_P120-1999
CESU-8	ibm-5349_P100-1998	ibm-1025_P100-1995
ISO-8859-1	ibm-5350_P100-1998	ibm-1026_P100-1995
US-ASCII	ibm-9447_P100-2002	ibm-1047_P100-1995
gb18030	ibm-9448_X100-2005	ibm-1097_P100-1995
ibm-912_P100-1995	ibm-9449_P100-2002	ibm-1112_P100-1995
ibm-913_P100-2000	ibm-5354_P100-1998	ibm-1122_P100-1999
ibm-914_P100-1995	ibm-1250_P100-1995	ibm-1123_P100-1995
ibm-915_P100-1995	ibm-1251_P100-1995	ibm-1130_P100-1997
ibm-1089_P100-1995	ibm-1252_P100-2000	ibm-1132_P100-1998
ibm-9005_X110-2007	ibm-1253_P100-1995	ibm-1137_P100-1999
ibm-813_P100-1995	ibm-1254_P100-1995	ibm-4517_P100-2005
ibm-5012_P100-1999	ibm-1255_P100-1995	ibm-1140_P100-1997
ibm-916_P100-1995	ibm-5351_P100-1998	ibm-1141_P100-1997
ibm-920_P100-1995	ibm-1256_P110-1997	ibm-1142_P100-1997
iso-8859_10-1998	ibm-5352_P100-1998	ibm-1143_P100-1997
iso-8859_11-2001	ibm-1257_P100-1995	ibm-1144_P100-1997
ibm-921_P100-1995	ibm-5353_P100-1998	ibm-1145_P100-1997
iso-8859_14-1998	ibm-1258_P100-1997	ibm-1146_P100-1997
ibm-923_P100-1998	macos-0_2-10.2	ibm-1147_P100-1997
ibm-942_P12A-1999	macos-6_2-10.4	ibm-1148_P100-1997

ibm-943_P15A-2003	macos-7_3-10.2	ibm-1149_P100-1997
ibm-943_P130-1999	macos-29-10.2	ibm-1153_P100-1999
ibm-33722_P12A_P12A-2009_U2	macos-35-10.2	ibm-1154_P100-1999
ibm-33722_P120-1999	ibm-1051_P100-1995	ibm-1155_P100-1999
ibm-954_P101-2007	ibm-1276_P100-1995	ibm-1156_P100-1999
euc-jp-2007	ibm-1006_P100-1995	ibm-1157_P100-1999
ibm-1373_P100-2002	ibm-1098_P100-1995	ibm-1158_P100-1999
windows-950-2000	ibm-1124_P100-1996	ibm-1160_P100-1999
ibm-950_P110-1999	ibm-1125_P100-1997	ibm-1164_P100-1999
ibm-1375_P100-2008	ibm-1129_P100-1997	ibm-1364_P110-2007
ibm-5471_P100-2006	ibm-1131_P100-1997	ibm-1371_P100-1999
ibm-1386_P100-2001	ibm-1133_P100-1997	ibm-1388_P103-2001
windows-936-2000	ISO_2022,locale=ja,version=0	ibm-1390_P110-2003
ibm-1383_P110-1999	ISO_2022,locale=ja,version=1	ibm-1399_P110-2003
ibm-5478_P100-1995	ISO_2022,locale=ja,version=2	ibm-5123_P100-1999
euc-tw-2014	ISO_2022,locale=ja,version=3	ibm-8482_P100-1999
ibm-964_P110-1999	ISO_2022,locale=ja,version=4	ibm-16684_P110-2003

ibm-949_P110-1999	ISO_2022,locale=ko,version=0	ibm-4899_P100-1998
ibm-949_P11A-1999	ISO_2022,locale=ko,version=1	ibm-4971_P100-1999
ibm-970_P110_P110-2006_U2	ISO_2022,locale=zh,version=0	ibm-9067_X100-2005
ibm-971_P100-1995	ISO_2022,locale=zh,version=1	ibm-12712_P100-1998
ibm-1363_P11B-1998	ISO_2022,locale=zh,version=2	ibm-16804_X110-1999
ibm-1363_P110-1997	HZ	ibm-37_P100-1995,swaplfnl
windows-949-2000	x11-compound-text	ibm-1047_P100-1995,swaplfnl
windows-874-2000	ISCII,version=0	ibm-1140_P100-1997,swaplfnl
ibm-874_P100-1995	ISCII,version=1	ibm-1141_P100-1997,swaplfnl
ibm-1162_P100-1999	ISCII,version=2	ibm-1142_P100-1997,swaplfnl
ibm-437_P100-1995	ISCII,version=3	ibm-1143_P100-1997,swaplfnl
ibm-720_P100-1997	ISCII,version=4	ibm-1144_P100-1997,swaplfnl
ibm-737_P100-1997	ISCII,version=5	ibm-1145_P100-1997,swaplfnl

ibm-775_P100-1996	ISCII,version=6	ibm-1146_P100-1997,swaplfnl
ibm-850_P100-1995	ISCII,version=7	ibm-1147_P100-1997,swaplfnl
ibm-851_P100-1995	ISCII,version=8	ibm-1148_P100-1997,swaplfnl
ibm-852_P100-1995	LMBCS-1	ibm-1149_P100-1997,swaplfnl
ibm-855_P100-1995	ibm-37_P100-1995	ibm-1153_P100-1999,swaplfnl
ibm-856_P100-1995	ibm-273_P100-1995	ibm-12712_P100-1998,swaplfnl
ibm-857_P100-1995	ibm-277_P100-1995	ibm-16804_X110-1999,swaplfnl
ibm-858_P100-1997	ibm-278_P100-1995	ebcdic-xml-us

前イメージタスク設定前

Kinesis や Apache Kafka のようなデータストリーミングターゲットに CDC 更新を書き込む場合、更新により変更される前のソースデータベース行の元の値を表示できます。これを可能にするために、はソースデータベースエンジンによって提供されたデータに基づいて、更新イベントの前のイメージ AWS DMS を設定します。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。

これを行うには、ソースデータベースシステムから収集された値を使用して、すべての更新オペレーションに新しい JSON 属性を追加する `BeforeImageSettings` パラメータを使用します。

`BeforeImageSettings` を全ロードと CDC タスクのみ、または CDC のみのタスクに適用することを忘れないでください。全ロードと CDC タスクにより、既存のデータが移行され、継続的変更のレプリケーションをできます。CDC のみのタスクは、データ変更のみのレプリケーションとなります。

全ロードのタスクには BeforeImageSettings を適用しないでください。

BeforeImageSettings で考えられるオプション：

- EnableBeforeImage — true に設定すると、前イメージが有効になります。デフォルトは false です。
- FieldName — 新しい JSON 属性に名前を割り当てます。EnableBeforeImage が true の場合、FieldName は必須であり、空にすることはできません。
- ColumnFilter — 前イメージを使用して追加する列を指定します。テーブルのプライマリキーの一部である列だけを追加するには、デフォルト値 pk-only を使用します。前イメージ値を持つ列を追加するには、all を使用します。変換前イメージは、CLOB や BLOB などのラージバイナリオブジェクト (LOB) データ型をサポートしていないことに注意します。

BeforeImageSettings の使用例：

```
"BeforeImageSettings": {
    "EnableBeforeImage": true,
    "FieldName": "before-image",
    "ColumnFilter": "pk-only"
  }
```

追加のテーブルマッピング設定など、前イメージ設定前の詳細については、「前イメージを使用した Kinesis データストリームの CDC 行の元の値のターゲットとしての表示」をご参照ください。

追加のテーブルマッピング設定を含む、Kafka の前イメージ設定の詳細については、「ターゲットとして Apache Kafka の CDC 行の元の値を表示するために前イメージを使用」をご参照ください。

エラー処理タスクの設定

次の設定を使用して、レプリケーションタスクのエラー処理の動作を設定できます。タスク設定ファイルを使用してタスク設定を設定する方法については、「タスク設定例」をご参照ください。

- DataErrorPolicy – レコードレベルでのデータ処理に関連するエラーが発生した場合に AWS DMS が実行するアクションを決定します。データ処理エラーの例には、変換エラー、変換時のエラー、および不良データが含まれます。デフォルトは LOG_ERROR です。
  - IGNORE_RECORD – タスクは続行され、該当するレコードのデータは無視されます。DataErrorEscalationCount プロパティのエラーカウンターは増分されます。したがって、テーブルにエラー数の制限を設定している場合、このエラーはその制限に向かってカウントされます。

  - LOG_ERROR – タスクは続行され、エラーはタスクログに書き込まれます。
  - SUSPEND_TABLE – タスクは続行されますが、エラーレコードのあるテーブルのデータはエラー状態になり、データ レプリケーションはされません。
  - STOP_TASK – タスクが停止し、手動での介入が必要になります。
- DataTruncationErrorPolicy – データが切り捨てられたときに AWS DMS が実行するアクションを決定します。デフォルトは LOG_ERROR です。
  - IGNORE_RECORD – タスクは続行され、該当するレコードのデータは無視されます。DataErrorEscalationCount プロパティのエラーカウンターは増分されます。したがって、テーブルにエラー数の制限を設定している場合、このエラーはその制限に向かってカウントされます。
  - LOG_ERROR – タスクは続行され、エラーはタスクログに書き込まれます。
  - SUSPEND_TABLE – タスクは続行されますが、エラーレコードのあるテーブルのデータはエラー状態になり、データ レプリケーションはされません。
  - STOP_TASK – タスクが停止し、手動での介入が必要になります。
- DataErrorEscalationPolicy – エラーが最大数(DataErrorEscalationCount パラメータで設定)に達したときに AWS DMS が実行するアクションを決定します。デフォルトは SUSPEND_TABLE です。
  - SUSPEND_TABLE – タスクは続行されますが、エラーレコードのあるテーブルのデータはエラー状態になり、データ レプリケーションはされません。
  - STOP_TASK – タスクが停止し、手動での介入が必要になります。
- DataErrorEscalationCount – 特定のレコードで、データに許可されるエラーの最大数を設定します。この数に到達すると、エラーレコードがあるテーブルのデータは、DataErrorEscalationPolicy で設定されているポリシーに従って処理されます。デフォルトは 0 です。
- EventErrorPolicy – タスク関連のイベントの送信中にエラーが発生した場合に AWS DMS が実行するアクションを決定します。指定できる値は次のとおりです。
  - IGNORE – タスクは続行され、そのイベントに関連付けられたデータはすべて無視されます。
  - STOP_TASK – タスクが停止し、手動での介入が必要になります。
- TableErrorPolicy – 特定のテーブルのデータまたはメタデータの処理中にエラーが発生した場合に、 AWS DMS が実行するアクションを決定します。このエラーは一般のテーブルデータにのみ適用され、特定のレコードに関連するエラーではありません。デフォルトは SUSPEND_TABLE です。

  - SUSPEND_TABLE – タスクは続行されますが、エラーレコードのあるテーブルのデータはエラー状態になり、データ レプリケーションはされません。
  - STOP_TASK – タスクが停止し、手動での介入が必要になります。
- TableErrorEscalationPolicy – エラーが最大数(TableErrorEscalationCount パラメータで設定)に達したときに AWS DMS が実行するアクションを決定します。デフォルトで、唯一のユーザー設定は STOP_TASK です。この設定では、タスクが停止し手動での介入が必要になります。
- TableErrorEscalationCount – 特定のテーブルで、一般データまたはメタデータに許可されるエラーの最大数。この数に到達すると、このテーブルのデータは、TableErrorEscalationPolicy で設定されたポリシーに従って処理されます。デフォルトは 0 です。
- RecoverableErrorCount – 環境エラーが発生したときに、タスクの再開を試みる最大回数。システムが再起動を試みる回数が指定の回数に達すると、タスクが停止し、手動での介入が必要になります。デフォルト値は -1 で、タスク AWS DMS を無期限に再起動するよう に指示します。この値を -1 に設定すると、DMS が試行する再試行回数は、返されるエラータイプによって次のように異なります。
  - 実行状態、復旧可能なエラー： 接続の喪失やターゲット適用の失敗などの復旧可能なエラーが発生した場合、DMS はタスクを 9 回再試行します。
  - 開始状態、回復可能なエラー： DMS はタスクを 6 回再試行します。
  - 実行中の状態、DMS によって処理された致命的なエラー: DMS はタスクを 6 回再試行します。
  - 実行中の状態、DMS で処理されない致命的なエラー: DMS はタスクを再試行しません。

  タスクの再開を試行しない場合には、この値を 0 に設定します。

  DMS タスクを適切に復旧RecoverableErrorIntervalするために十分な間隔で十分な再試行が行われるように、 RecoverableErrorCountと を 値に設定することをお勧めします。致命的なエラーが発生した場合、DMS はほとんどのシナリオで再起動の試行を停止します。
- RecoverableErrorInterval – タスクの再起動の試行の間に AWS DMS が待機する秒数。デフォルトは 5 です。
- RecoverableErrorThrottling – 有効にすると、タスクの再起動を試みる間隔が RecoverableErrorInterval の値に基づいて徐々に増加します。例えば、RecoverableErrorInterval が 5 秒に設定されている場合、次の再試行は 10 秒後、次は 20 秒後、次は 20 秒後、その次は 40 秒後に行われます。デフォルトは true です。

- RecoverableErrorThrottlingMax - が有効になっている場合、タスクの再起動を試行するまでに AWS DMS RecoverableErrorThrottling が待機する最大秒数。デフォルトは 1800 です。
- RecoverableErrorStopRetryAfterThrottlingMax – に設定するとtrue、 は、 ごとに、復旧試行間の AWS DMS 待機の最大秒数に達した後にタスクの再起動を停止しますRecoverableErrorThrottlingMax。
- ApplyErrorDeletePolicy – DELETE オペレーションとの競合がある場合に、 AWS DMS が実行するアクションを決定します。デフォルトは IGNORE_RECORD です。利用できる値には以下のとおりです。
  - IGNORE_RECORD – タスクは続行され、該当するレコードのデータは無視されます。ApplyErrorEscalationCount プロパティのエラーカウンターは増分されます。したがって、テーブルにエラー数の制限を設定している場合、このエラーはその制限に向かってカウントされます。
  - LOG_ERROR – タスクは続行され、エラーはタスクログに書き込まれます。
  - SUSPEND_TABLE – タスクは続行されますが、エラーレコードのあるテーブルのデータはエラー状態になり、データ レプリケーションはされません。
  - STOP_TASK – タスクが停止し、手動での介入が必要になります。
- ApplyErrorInsertPolicy – INSERT オペレーションとの競合がある場合に、 AWS DMS が実行するアクションを決定します。デフォルトは LOG_ERROR です。利用できる値には以下のとおりです。
  - IGNORE_RECORD – タスクは続行され、該当するレコードのデータは無視されます。ApplyErrorEscalationCount プロパティのエラーカウンターは増分されます。したがって、テーブルにエラー数の制限を設定している場合、このエラーはその制限に向かってカウントされます。
  - LOG_ERROR – タスクは続行され、エラーはタスクログに書き込まれます。
  - SUSPEND_TABLE – タスクは続行されますが、エラーレコードのあるテーブルのデータはエラー状態になり、データ レプリケーションはされません。
  - STOP_TASK – タスクが停止し、手動での介入が必要になります。
  - INSERT_RECORD – 挿入されたソースレコードと同じプライマリ キーを含む既存のターゲットレコードがある場合、ターゲットレコードは更新されます。
- ApplyErrorUpdatePolicy – UPDATE オペレーションと競合する欠落データがある場合に AWS DMS が実行するアクションを決定します。デフォルトは LOG_ERROR です。利用できる値には以下のとおりです。

  - IGNORE_RECORD – タスクは続行され、該当するレコードのデータは無視されます。ApplyErrorEscalationCount プロパティのエラーカウンターは増分されます。したがって、テーブルにエラー数の制限を設定している場合、このエラーはその制限に向かってカウントされます。
  - LOG_ERROR – タスクは続行され、エラーはタスクログに書き込まれます。
  - SUSPEND_TABLE – タスクは続行されますが、エラーレコードのあるテーブルのデータはエラー状態になり、データ レプリケーションはされません。
  - STOP_TASK – タスクが停止し、手動での介入が必要になります。
  - UPDATE_RECORD - ターゲットレコードがない場合、欠落しているターゲットレコードがターゲットテーブルに挿入されます。 は、タスクの LOB 列のサポート AWS DMS を完全に無効にします。このオプションを選択するには、Oracle がソースデータベースの場合、すべてのソーステーブルの列に対し、完全なサプリメンタルロギングが有効である必要があります。
- ApplyErrorEscalationPolicy – エラーの最大数 (ApplyErrorEscalationCountパラメータを使用して設定) に達したときに AWS DMS が実行するアクションを決定します。デフォルトは LOG_ERROR です：
  - LOG_ERROR – タスクは続行され、エラーはタスクログに書き込まれます。
  - SUSPEND_TABLE – タスクは続行されますが、エラーレコードのあるテーブルのデータはエラー状態になり、データ レプリケーションはされません。
  - STOP_TASK – タスクが停止し、手動での介入が必要になります。
- ApplyErrorEscalationCount – このオプションでは変更プロセスオペレーションが実施されている間、特定のテーブルに許可される APPLY 競合の最大数を設定します。この数に到達すると、このテーブルのデータは、ApplyErrorEscalationPolicy パラメータで設定されたポリシーに従って処理されます。デフォルトは 0 です。
- ApplyErrorFailOnTruncationDdl – このオプションを true に設定すると、CDC 中に追跡されたいずれかのテーブルで切り捨てが実行された場合に、タスクは失敗します。デフォルトは false です。

  この方法は、DDL テーブルの切り捨てレプリケーションが行われない、PostgreSQL バージョン 11.x 以前またはその他のソース エンドポイントでは機能しません。
- FailOnNoTablesCaptured – このオプションを true に設定すると、タスクに対して定義されたテーブル マッピングによりタスクの開始時にテーブルが見つからなかった場合、タスクは失敗します。デフォルトは false です。
- FailOnTransactionConsistencyBreached – このオプションは CDC でソースとして Oracle を使用するタスクに適用されます。デフォルトは False です。これを true に設定すると、トラン

ザクションが指定されたタイムアウトよりも長い時間開かれていて、削除できる場合、タスクは失敗します。

CDC タスクが Oracle で始まると、CDC を開始する前に、最も古いオープントランザクションが終了するまで期間限定で AWS DMS 待機します。タイムアウトに達するまで最も古いオープントランザクションが閉じない場合、ほとんどの場合、そのトランザクションを無視して CDC AWS DMS を開始します。このオプションを `true` に設定すると、タスクは失敗します。

- `FullLoadIgnoreConflicts` - キャッシュされたイベントを適用するときに`true`「影響を受ける行がゼロ AWS DMS 」を無視し、「重複」エラーを に設定するには、このオプションを に設定します。に設定すると`false`、 AWS DMS はエラーを無視せずにすべてのエラーを報告します。デフォルトは `true` です。

Redshift がターゲットの場合、テーブルロードエラーは、`STL_LOAD_ERRORS` として報告されることに注意します。詳細については、「Amazon Redshift データベース開発者ガイド」の「STL_LOAD_ERRORS」を参照してください。

タスク設定の保存

別のタスクで設定を再利用する場合、タスクの設定を JSON ファイルして保存することができます。JSON ファイルにコピーするタスク設定は、タスクのセクション[Overview details] (概要の詳細)の下で検索できます。

> **Note**
>
> 他のタスクでタスク設定を再利用するときは、`CloudWatchLogGroup` と `CloudWatchLogStream` 属性を削除します。それ以外の場合は、SYSTEM ERROR MESSAGE:Task Settings CloudWatchLogGroup または というエラーが表示されます。作成時に設定 CloudWatchLogStream することはできません。

たとえば、以下の JSON ファイルにはタスクに保存された設定が含まれています。

```
{
    "TargetMetadata": {
        "TargetSchema": "",
        "SupportLobs": true,
        "FullLobMode": false,
        "LobChunkSize": 0,
```

```
        "LimitedSizeLobMode": true,
        "LobMaxSize": 32,
        "InlineLobMaxSize": 0,
        "LoadMaxFileSize": 0,
        "ParallelLoadThreads": 0,
        "ParallelLoadBufferSize": 0,
        "BatchApplyEnabled": false,
        "TaskRecoveryTableEnabled": false,
        "ParallelLoadQueuesPerThread": 0,
        "ParallelApplyThreads": 0,
        "ParallelApplyBufferSize": 0,
        "ParallelApplyQueuesPerThread": 0
    },
    "FullLoadSettings": {
        "TargetTablePrepMode": "DO_NOTHING",
        "CreatePkAfterFullLoad": false,
        "StopTaskCachedChangesApplied": false,
        "StopTaskCachedChangesNotApplied": false,
        "MaxFullLoadSubTasks": 8,
        "TransactionConsistencyTimeout": 600,
        "CommitRate": 10000
    },
    "Logging": {
        "EnableLogging": true,
        "LogComponents": [
            {
                "Id": "TRANSFORMATION",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "SOURCE_UNLOAD",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "IO",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "TARGET_LOAD",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "PERFORMANCE",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
```

```
            },
            {
                "Id": "SOURCE_CAPTURE",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "SORTER",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "REST_SERVER",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "VALIDATOR_EXT",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "TARGET_APPLY",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "TASK_MANAGER",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "TABLES_MANAGER",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "METADATA_MANAGER",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "FILE_FACTORY",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "COMMON",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "ADDONS",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
```

```
            },
            {
                "Id": "DATA_STRUCTURE",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "COMMUNICATION",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "FILE_TRANSFER",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            }
        ]
    },
    "ControlTablesSettings": {
        "ControlSchema": "",
        "HistoryTimeslotInMinutes": 5,
        "HistoryTableEnabled": false,
        "SuspendedTablesTableEnabled": false,
        "StatusTableEnabled": false,
        "FullLoadExceptionTableEnabled": false
    },
    "StreamBufferSettings": {
        "StreamBufferCount": 3,
        "StreamBufferSizeInMB": 8,
        "CtrlStreamBufferSizeInMB": 5
    },
    "ChangeProcessingDdlHandlingPolicy": {
        "HandleSourceTableDropped": true,
        "HandleSourceTableTruncated": true,
        "HandleSourceTableAltered": true
    },
    "ErrorBehavior": {
        "DataErrorPolicy": "LOG_ERROR",
        "DataTruncationErrorPolicy": "LOG_ERROR",
        "DataErrorEscalationPolicy": "SUSPEND_TABLE",
        "DataErrorEscalationCount": 0,
        "TableErrorPolicy": "SUSPEND_TABLE",
        "TableErrorEscalationPolicy": "STOP_TASK",
        "TableErrorEscalationCount": 0,
        "RecoverableErrorCount": -1,
        "RecoverableErrorInterval": 5,
        "RecoverableErrorThrottling": true,
```

```
        "RecoverableErrorThrottlingMax": 1800,
        "RecoverableErrorStopRetryAfterThrottlingMax": true,
        "ApplyErrorDeletePolicy": "IGNORE_RECORD",
        "ApplyErrorInsertPolicy": "LOG_ERROR",
        "ApplyErrorUpdatePolicy": "LOG_ERROR",
        "ApplyErrorEscalationPolicy": "LOG_ERROR",
        "ApplyErrorEscalationCount": 0,
        "ApplyErrorFailOnTruncationDdl": false,
        "FullLoadIgnoreConflicts": true,
        "FailOnTransactionConsistencyBreached": false,
        "FailOnNoTablesCaptured": true
    },
    "ChangeProcessingTuning": {
        "BatchApplyPreserveTransaction": true,
        "BatchApplyTimeoutMin": 1,
        "BatchApplyTimeoutMax": 30,
        "BatchApplyMemoryLimit": 500,
        "BatchSplitSize": 0,
        "MinTransactionSize": 1000,
        "CommitTimeout": 1,
        "MemoryLimitTotal": 1024,
        "MemoryKeepTime": 60,
        "StatementCacheSize": 50
    },
    "PostProcessingRules": null,
    "CharacterSetSettings": null,
    "LoopbackPreventionSettings": null,
    "BeforeImageSettings": null,
    "FailTaskWhenCleanTaskResourceFailed": false
}
```

AWS DMS タスクでのソースデータベースの LOB サポートの設定

ラージバイナリオブジェクト (LOB) をシステム間で移行することは難しい場合があります。 AWS DMS には、LOB の列のチューニングに役立つ多くのオプションが用意されています。によって LOBs と見なされるデータ型とタイミングを確認するには AWS DMS、 ドキュメント AWS DMS を参照してください。

データベース間でデータを移行するとき、特に異機種間で移行するときは、LOB の保存方法の見直しが必要になる場合があります。その場合、LOB データを移行する必要はありません。

LOB を含める場合は、他の LOB 設定を決定できます。

- LOB モードにより、LOB の処理方法が決定されます。
  - フル LOB モード – フル LOB モードでは、サイズに関係なく、すべての LOBsをソースからターゲット AWS DMS に移行します。この設定 AWS DMS では、 は予想される LOBs の最大サイズに関する情報がありません。したがって、LOB は一度に 1 つずつ移行されます。Full LOB mode は非常に低速になることがあります。
  - [Limited LOB mode] (制限付き LOB モード) - 制限付き LOB モードでは、DMS の最大許容 LOB サイズを設定します。このオプションでは、DMS によってメモリが事前に割り当てられ、LOB データを一括でロードできるようになります。最大許容 LOB サイズを超える LOB は切り捨てられ、警告がログファイルに発行されます。制限付き LOB モードでは、完全 LOB モードよりも大幅にパフォーマンスが向上します。可能な限り Limited LOB mode を使用することをお勧めします。最大推奨値は 102,400 KB (100 MB) です。

    > **Note**
    >
    > [Max LOB size (K)](最大 LOB サイズ(K)) オプションを 63 KB より大きい値で使用すると、制限付き LOB モードで実行されるように設定された全ロードのパフォーマンスが低下します。全ロード中、DMS は [Max LOB size (k)](最大 LOB サイズ (k)) 値にコミットレートを掛けてメモリを割り当てて、この積に LOB 列数を乗算します。DMS がそのメモリを事前割り当てできない場合、DMS は SWAP メモリの消費をスタートし、全ロードのパフォーマンスが低下します。したがって、制限された LOB モードを使用するときにパフォーマンスの問題が発生する場合は、許容レベルのパフォーマンスを達成するまでコミットレートを下げることを検討してください。テーブルの LOB ディストリビューションを理解したら、サポートされているエンドポイントでインライン LOB モードを使用することを検討することもできます。
    > LOB サイズの制限を検証するには、`LobMaxSize` (K) を `ValidationPartialLobSize` と同じ値に設定する必要があります。

  - [Inline LOB mode](インライン LOB モード) — インライン LOB モードでは、DMS がインラインで転送する最大 LOB サイズを設定します。指定されたサイズより小さい LOB はインラインで転送されます。指定されたサイズよりも大きい LOB は、フル LOB モードを使用してレプリケーションされます。このオプションを選択すると、ほとんどの LOB が小さい場合に、小さい LOB と大規模な LOB 両方のレプリケーションができます。DMS は S3 や Redshift のようなフル LOB モードをサポートしないエンドポイントでは、インライン LOB モードをサポートしていません。

> **Note**
>
> Oracle では、LOB は可能な限り VARCHAR データ型として扱われます。このアプローチは、 がデータベースからそれらを一括で AWS DMS 取得することを意味します。これは、他の方法よりも大幅に高速です。Oracle での VARCHAR の最大サイズは 32 K です。そのため、Oracle がソースデータベースである場合は、最大許容 LOB サイズを 32 K 未満に設定するのが最適です。

- 制限付き LOB モードで実行されるようにタスクを設定している場合、[Max LOB size (K) (最大 LOB サイズ (K))] オプションによって AWS DMS の最大許容 LOB サイズが決まります。この値よりも大きい LOB はこの値まで切り捨てられます。
- タスクがフル LOB モードを使用するように設定されている場合、 は LOBsして AWS DMS 取得します。[LOB chunk size (K) (LOB チャンクサイズ (K))] オプションにより、各ピースのサイズが決定されます。このオプションを設定するときは、ネットワーク設定の最大許容パケットサイズに特に注意してください。LOB チャンクサイズが最大許容パケットサイズを超えた場合は、切断エラーが表示されることがあります。`LobChunkSize` の推奨値は 64 キロバイトです。`LobChunkSize` の値を 64 キロバイト以上に設定すると、タスクが失敗する可能性があります。
- インライン LOB モードで実行するようにタスクを設定している場合、`InlineLobMaxSize` 設定により、どの LOB DMS がインラインで転送されるかが決まります。

> **Note**
>
> LOB データタイプはプライマリキーが含まれるテーブルおよびビューのみで使用できます。

タスク設定でこれらのオプションを指定する方法の詳細については、「ターゲットメタデータのタスク設定」をご参照ください。

複数のタスクの作成

移行シナリオによっては、複数の移行タスクを作成する必要があります。タスクは個別に動作し、同時に実行できます。各タスクには、独自の初期ロード、CDC、およびログ読み取りプロセスがあります。データ操作言語 (DML) を通じて関連付けられたテーブルは、同じタスクの一部である必要があります。

移行のために複数のタスクを作成する理由として、以下の理由が挙げられます。

- タスクのターゲットテーブルが異なるデータベースに存在している (分散していたり、システムを複数のシステムに分割している場合など)。
- フィルタリングを使用して、大きいテーブルの移行を複数のタスクに分割したい。

> **Note**
>
> タスクごとに独自の変更キャプチャおよびログ読み取りプロセスがあるため、タスク間で変更は調整されません。そのため、複数のタスクを使用して移行を実行する場合は、個々のソーストランザクションを完全に単一のタスクに収めていることを確認します。個々のトランザクションが複数のタスクに分散されていない場合は、複数のタスクを使用して移行を実行できます。

AWS DMS を使用した継続的なレプリケーションのタスクの作成

ソースデータストアから継続的な変更をキャプチャする AWS DMS タスクを作成できます。このキャプチャはデータの移行中に実行できます。サポートされているターゲットデータストアへの初めての (フルロード) 移行が完了した後に、継続的な変更をキャプチャするタスクを作成することもできます。このプロセスは継続的なレプリケーションまたは変更データキャプチャ (CDC) と呼ばれます。AWS DMS では、このプロセスを使用してソースデータストアの継続的な変更をレプリケートします。データベースエンジンのネイティブ API を使用してデータベースログへの変更を収集することが、このプロセスの仕組みです。

> **Note**
>
> ビューを移行できるのは、フルロードタスクのみを使用する場合です。CDC のみのタスク、または完了後に CDC を開始するフルロードタスクの場合、移行の範囲はソースのテーブルのみとなります。フルロードのみのタスクを使用すると、ビューまたはテーブルとビューの組み合わせを移行できます。詳細については、「JSON を使用するテーブル選択および変換を指定する」を参照してください。

各ソースエンジンには、この変更ストリームを特定のユーザーアカウントにパブリッシュするための固有の設定要件があります。ほとんどのエンジンでは、キャプチャプロセスがデータを損失することなく意味のある方法で変更データを使用できるようにするために、追加の設定が必要になります。例

えば、Oracle ではサプリメンタルロギングの追加が必要であり、MySQL では行レベルのバイナリログ (bin ログ) が必要です。

ソースデータベースから継続的な変更を読み取るために、AWS DMS はエンジン固有の API アクションを使用してソースエンジンのトランザクションログから変更を読み取ります。AWS DMS での実際の例は次のとおりです。

- Oracle の場合、AWS DMS は、Oracle LogMiner API または Binary Reader API (bfile API) を使用して継続的な変更を読み取ります。AWS DMS は、システム変更番号 (SCN) に基づいてオンラインまたはアーカイブ REDO ログから継続的な変更を読み取ります。
- Microsoft SQL Server の場合、AWS DMS は MS-Replication または MS-CDC を使用して SQL Server のトランザクションログに情報を書き込みます。次に、SQL Server の `fn_dblog()` 関数または `fn_dump_dblog()` 関数を使用して、ログシーケンス番号 (LSN) に基づいてトランザクションログの変更を読み取ります。
- MySQL の場合、AWS DMS は、行ベースのバイナリログ (binlog) から変更を読み取り、この変更をターゲットに移行します。
- PostgreSQL の場合、AWS DMS は論理レプリケーションスロットを設定して、test_decoding プラグインを使用し、ソースから変更を読み取り、この変更をターゲットに移行します。
- Amazon RDS をソースとして使用する場合は、CDC をセットアップするためにバックアップが有効になっていることを確認することをお勧めします。また、変更ログを十分な時間 (通常は 24 時間で十分) 保持するようにソースデータベースが設定されていることを確認することをお勧めします。各エンドポイントの特定の設定については、次を参照してください。
  - Amazon RDS for Oracle: の マネージド Oracle AWSソースの設定 AWS DMS
  - Amazon RDS for MySQL and Aurora MySQL: AWSマネージド型の MySQL 互換データベースをソースとして使用する AWS DMS
  - Amazon RDS for SQL Server: Cloud SQL Server DB インスタンスでの継続的なレプリケーションのセットアップ
  - Amazon RDS for PostgreSQL と Aurora PostgreSQL: PostgreSQL は必要な WAL を自動的に保持します

継続的なレプリケーションタスクには 2 つのタイプがあります。

- フルロード + CDC: このタスクは既存のデータを移行して、ソースデータベースへの変更に基づいてターゲットデータベースを更新します。

- CDC のみ: このタスクは、ターゲットデータベースにデータが移行した後、継続的な変更を移行します。

CDC 開始点を起点とするレプリケーションの実行

AWS DMS の継続的なレプリケーションタスク (変更データキャプチャのみ) は、複数の開始点から開始できます。これには次が含まれます。

- カスタム CDC 開始時刻から – AWS Management Console または AWS CLI を使用して、レプリケーションを開始するタイムスタンプを AWS DMS に指示できます。AWS DMS は、このカスタム CDC 開始時刻から継続的なレプリケーションタスクを開始します。AWS DMS は、指定されたタイムスタンプ (UTC) を、SQL Server の LSN や Oracle の SCN などのネイティブ開始点に変換します。AWS DMS は、エンジン固有のメソッドを使用して、ソースエンジンの変更ストリームに基づいて移行タスクを開始する場所を決定します。

  > **Note**
  >
  > ソースが Db2 の場合、`StartFromContext` 接続属性を必要なタイムスタンプに設定した場合にのみ CDC 開始時刻をカスタマイズできます。
  > PostgreSQL がソースの場合、カスタム CDC 開始時刻はサポートされません。これは、Oracle や SQL Server とは異なり、PostgreSQL データベースエンジンにはタイムスタンプを LSN や SCN にマップする方法がないためです。

- CDC ネイティブ開始点から – ソースエンジンのトランザクションログのネイティブ開始点から開始することもできます。タイムスタンプはトランザクションログ内の複数のネイティブ開始点を示す可能性もあるため、場合によっては、この方法のほうが適していることがあります。AWS DMS は、この機能を次のソースエンドポイントでサポートしています。
  - SQL Server
  - PostgreSQL
  - Oracle
  - MySQL
  - MariaDB

タスクが作成されると、AWS DMS は CDC の開始点にマークが付けられます。これを変更することはできません。別のCDC開始点を使用するには、新しいタスクを作成します。

CDC ネイティブ開始点の決定

CDC ネイティブ開始点は、CDC を開始できる時間を定義するデータベースエンジンのログ内の開始点です。一括データダンプが既にターゲットに適用されている場合を例にとって説明します。この場合、継続的なレプリケーションのみのタスクのネイティブ開始点を検索できます。データの不整合を回避するには、レプリケーションのみのタスクの開始点は慎重に選択します。DMS は、選択された CDC 開始点の後に開始されたトランザクションをキャプチャします。

サポートされているソースエンジンから CDC ネイティブ開始点を検索する方法の例は次のとおりです。

SQL Server

SQL Server のログシーケンス番号 (LSN) は次の 3 つの部分で構成されています。

- 仮想ログファイル (VLF) シーケンス番号
- ログブロックの開始オフセット
- スロット番号

LSN の例: `00000014:00000061:0001`

トランザクションログのバックアップ設定に基づいて SQL Server 移行タスクの開始点を取得するには、SQL Server の `fn_dblog()` 関数または `fn_dump_dblog()` 関数を使用します。

SQL Server で CDC ネイティブ開始点を使用するには、継続的なレプリケーションの対象の任意のテーブルにパブリケーションを作成します。CDC ネイティブ開始点を使用せずに CDC を使用すると、AWS DMS はパブリケーションを自動的に作成します。

PostgreSQL

PostgreSQL データベースがソースの場合は、CDC 復旧チェックポイントを使用できます。このチェックポイントの値は、ソースデータベース (親タスク) に対して継続的なレプリケーションタスクが実行される際、さまざまな時点で生成されます。チェックポイントの一般的な詳細については、「CDC 開始点としてのチェックポイントの使用」を参照してください。

ネイティブ開始点として使用するチェックポイントを特定するには、データベースの `pg_replication_slots` ビューを使用するか、AWS Management Console から親タスク概要の詳細を使用します。

コンソールで親タスクの概要の詳細を検索するには

1. AWS Management Console にサインインして、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。

   IAM ユーザーとしてサインインしている場合は、AWS DMS にアクセスするための適切なアクセス許可があることを確認します。必要なアクセス許可の詳細については、「AWS DMS の使用に必要な IAM アクセス許可」を参照してください。
2. ナビゲーションペインで、[データベース移行タスク] を選択します。
3. [データベース移行タスク] ページのリストから親タスクを選択すると、親タスクのページが開き、概要の詳細が表示されます。
4. [変更データキャプチャ (CDC)]、[変更データキャプチャ (CDC) の開始位置]、[変更データキャプチャ (CDC) の復旧チェックポイント] の下にあるチェックポイントの値を調べます。

   値は、次のように表示されます。

   ```
   checkpoint:V1#1#000004AF/B00000D0#0#0#*#0#0
   ```

   ここで、4AF/B00000D0 コンポーネントは、このネイティブ CDC 開始点に関して指定が必須です。CDC タスクを作成して、PostgreSQL ソースのこの開始点でレプリケーションを開始するために、DMS API の CdcStartPosition パラメータをこの値に設定します。AWS CLI を使用してこのような CDC タスクを作成する方法の詳細については、「で AWSマネージド PostgreSQL DB インスタンスで CDC を有効にする AWS DMS」を参照してください。

Oracle

システム変更番号 (SCN) は、Oracle データベースで使用される論理的な内部タイムスタンプです。SCN は、データベース内で発生するイベントを順序付けします。これは、トランザクションの ACID プロパティを満たすために必要です。Oracle データベースは SCN を使用して、すべての変更がディスクに書き込まれた場所をマークし、復旧アクションによってこの場所に既に書き込まれた変更が適用されないようにします。Oracle はまた、復旧を停止できるように、SCN を使用してデータセットに対して REDO が存在しないポイントをマークします。

Oracle データベース内の現在の SCN を取得するには、次のコマンドを実行します。

```
SELECT CURRENT_SCN FROM V$DATABASE
```

SCN またはタイムスタンプを使用して CDC タスクを開始すると、オープントランザクションの結果が失われ、このような結果の移行に失敗します。オープントランザクションとは、タスクの開始位置より前に開始され、タスクの開始位置の後にコミットされたトランザクションを指します。SCN とタイムスタンプを特定して、すべてのオープントランザクションを含む時点から CDC タスクを開始できます。詳細については、Oracle オンラインドキュメントの「トランザクション」を参照してください。バージョン 3.5.1 以降では、タスクを開始するために SCN またはタイムスタンプを使用する場合、AWS DMS は openTransactionWindow エンドポイント設定を使用して CDC のみのタスクのオープントランザクションをサポートしています。

この openTransactionWindow 設定を使用する場合、オープントランザクションを処理する期間を分単位で指定する必要があります。AWS DMS はキャプチャ位置を移動して、データキャプチャを開始する新しい位置を検索します。AWS DMS は、必要となる Oracle REDO ログまたはアーカイブ REDO ログからオープントランザクションをスキャンするために新しい開始位置を使用します。

MySQL

MySQL バージョン 5.6.3 のリリース前までは、MySQL のログシーケンス番号 (LSN) は 4 バイトの符号なし整数でした。MySQL バージョン 5.6.3 では、REDO ログファイルのサイズ上限が 4 GB から 512 GB に引き上げられ、LSN は 8 バイトの符号なし整数になりました。この増加は、サイズ情報の保存に追加のバイトが必要になったことを受けてのことです。MySQL 5.6.3 以降で構築された LSN 値を使用するアプリケーションでは、LSN 値を保存して比較するために 32 ビット変数ではなく 64 ビット変数を使用する必要があります。MySQL の LSN の詳細については、「MySQL のドキュメント」を参照してください。

MySQL データベース内の現在の LSN を取得するには、次のコマンドを実行します。

```
mysql> show master status;
```

このクエリは、binlog ファイル名、位置、その他いくつかの値を返します。CDC ネイティブ開始点は、binlog ファイル名と位置の組み合わせになります (mysql-bin-changelog.000024:373 など)。この例では、mysql-bin-changelog.000024 は binlog ファイル名、373 は AWS DMS で変更のキャプチャを開始すべき位置です。

CDC 開始点としてのチェックポイントの使用

継続的なレプリケーションタスクが変更を移行し、AWS DMS では AWS DMS 固有のチェックポイント情報を随時キャッシュします。AWS DMS が作成するチェックポイントには、レプリケーショ

ンエンジンが変更ストリームの復旧点を認識できる情報が含まれています。チェックポイントを使用すると、変更のタイムラインに戻り、失敗した移行タスクを回復できます。また、チェックポイントを使用して、任意の時点で別のターゲットに対して別の継続的なレプリケーションタスクを開始することもできます。

チェックポイント情報の取得には、次の 3 つのいずれかの方法を使用します。

- API オペレーション DescribeReplicationTasks を実行して結果を表示します。タスクごとに情報にフィルターを適用してチェックポイントを検索できます。タスクが停止または失敗した最新のチェックポイントを取得できます。タスクを削除すると、この情報は失われます。
- ターゲットインスタンスで awsdms_txn_state というメタデータテーブルを表示します。テーブルでクエリを実行すると、チェックポイント情報を取得できます。メタデータテーブルを作成するには、タスクの作成時に TaskRecoveryTableEnabled パラメータを Yes に設定します。この設定があると、AWS DMS はチェックポイント情報を継続的にターゲットメタデータテーブルに書き込みます。タスクを削除すると、この情報は失われます。

  メタデータテーブル checkpoint:V1#34#00000132/0F000E48#0#0#*#0#121 のチェックポイントの例は次のとおりです。
- ナビゲーションペインで、[データベース移行タスク] を選択して、[Database migration tasks] ページに表示されるリストから親タスクを選択します。親タスクページが開き、概要の詳細が表示されます。[変更データキャプチャ (CDC)]、[変更データキャプチャ (CDC) の開始位置]、[変更データキャプチャ (CDC) の復旧チェックポイント] の下にあるチェックポイントの値を調べます。チェックポイント値は、次のように表示されます。

  ```
  checkpoint:V1#1#000004AF/B00000D0#0#0#*#0#0
  ```

コミット時点またはサーバー時間のポイントでのタスクの停止

CDC ネイティブ開始点の導入により、AWS DMS は次の時点でタスクを停止することもできます。

- ソースでのコミット時刻
- レプリケーションインスタンスのサーバー時間

必要に応じてタスクを変更して、停止する時刻を UTC で設定できます。タスクは、設定したコミット時間またはサーバー時間に基づいて自動的に停止します。または、タスクの作成時に移行タスクを停止する時間が明らかである場合は、タスクの作成時に停止時刻を設定できます。

> **Note**
>
> 新しい AWS DMS サーバーレスレプリケーションを初めて開始する場合、すべてのリソースを初期化するのに最大 40 分かかることがあります。`server_time` オプションは、リソースの初期化が完了した後にのみ適用されることに注意します。

双方向レプリケーションの実行

AWS DMS タスクを使用して、2 つのシステム間で双方向レプリケーションを実行できます。双方向レプリケーションの場合、2 つのシステム間で同じテーブル (またはテーブルセット) のデータを両方向にレプリケートできます。

例えば、EMPLOYEE テーブルをデータベース A からデータベース B にコピーして、このテーブルに対する変更をデータベース A からデータベース B にレプリケートできます。また、EMPLOYEE テーブルに対する変更をデータベース B から A にレプリケートすることもできます。つまり、双方向にレプリケートすることになります。

> **Note**
>
> AWS DMS 双方向レプリケーションは、プライマリノード、競合解決などの完全なマルチマスターソリューションを意図して設計されているわけではありません。

双方向レプリケーションは、異なるノード上のデータが運用上分離されている状況で使用します。つまり、ノード A で動作しているアプリケーションが変更したデータ要素があり、そのノード A がノード B との双方向レプリケーションを実行するとします。ノード A のそのデータ要素は、ノード B で動作するアプリケーションによって変更されることはありません。

AWS DMS は、次のデータベースエンジンで双方向レプリケーションをサポートしています。

- Oracle
- SQL Server
- MySQL
- PostgreSQL
- Amazon Aurora MySQL 互換エディション
- Aurora PostgreSQL 互換エディション

双方向レプリケーションタスクの作成

AWS DMS 双方向レプリケーションを有効にするには、両方のデータベース(A と B) のソースエンドポイントとターゲットエンドポイントを設定します。例えば、データベース A のソースエンドポイント、データベース B のソースエンドポイント、データベース A のターゲットエンドポイント、データベース B のターゲットエンドポイントを設定します。

次に、ソース A がデータをターゲット B に移動するタスクと、ソース B がデータをターゲット A に移動するタスクの 2 つのタスクを作成します。さらに、各タスクでループバック防止機能が設定されていることを確認します。これにより、両方のタスクのターゲットに同一の変更が適用されることが避けられ、少なくとも一方のタスクのデータが破損するのを防ぐことができます。詳細については、「ループバックの防止」を参照してください。

最も簡単な方法としては、まずデータベース A とデータベース B の両方に同一のデータセットを作成します。次に CDC のみのタスクを 2 つ作成します。1 つは A から B にデータをレプリケートするタスクで、もう 1 つは B から A にデータをレプリケートするタスクです。

AWS DMS を使用してノード A からノード B の新しいデータセット (データベース) をインスタンス化するには、次を実行します。

1. データベース A から B にデータを移動するにはフルロードおよび CDC タスクを使用します。この間、アプリケーションがデータベース B のデータを変更しないことを確認します。
2. フルロードが完了した後、アプリケーションがデータベース B のデータを変更できるようになる前に、データベー ス B の時刻または CDC 開始位置をメモします。手順については、「CDC 開始点を起点とするレプリケーションの実行」を参照してください。
3. この開始時刻または CDC 開始位置を使用して、データベース B から A にデータを移動する CDC 専用タスクを作成します。

> **Note**
>
> フルロードと CDC に使用できる双方向ペアのタスクは 1 つのみです。

ループバックの防止

タスク T1 で AWS DMS がソースデータベース A から変更ログを読み取り、ターゲットデータベース B に変更内容を適用するケースを例に、ループバックの防止について説明します。

次に、2 番目のタスクである T2 がソースデータベース B から変更ログを読み取り、ターゲットデータベース A に適用します。T2 がこれを行う前に、DMS では、ソースデータベース A からターゲットデータベース B に対して行われたのと同じ変更がソースデータベース B に対して行われていないことを確認する必要があります。つまり、DMS は、このような変更がターゲットデータベース A にエコー (ループ) されないことを確認する必要があります。これを行わないと、データベース A のデータが破損する可能性があります。

変更のループバックを回避するには、各双方向レプリケーションタスクに次のタスク設定を追加します。これにより、どちらの方向でもループバックデータの破損が発生しないようにします。

```
{
. . .

  "LoopbackPreventionSettings": {
    "EnableLoopbackPrevention": Boolean,
    "SourceSchema": String,
    "TargetSchema": String
  },

. . .
}
```

LoopbackPreventionSettings タスク設定により、トランザクションが新規であるか、反対方向のレプリケーションタスクからのエコーであるかが定まります。AWS DMS がトランザクションをターゲットデータベースに適用すると、変更を示す DMS テーブル (awsdms_loopback_prevention) が更新されます。各トランザクションをターゲットに適用する前に、DMS はこの awsdms_loopback_prevention テーブルへの参照を含むトランザクションをすべて無視します。したがって、変更は適用されません。

双方向ペアの各レプリケーションタスクにこのようなタスク設定を含めます。このような設定により、ループバック防止機能が有効になります。さらに、この設定は、各エンドポイントの awsdms_loopback_prevention テーブルを含むタスク内の各ソースデータベースとターゲットデータベースのスキーマも指定します。

各タスクがこのようなエコーを識別して破棄できるようにするには、EnableLoopbackPrevention を true に設定します。awsdms_loopback_prevention を含むソースでスキーマを指定するには、SourceSchema をソースデータベース内の該当するスキーマ名に設定します。同じテーブルを含むターゲットでスキーマを指定するには、TargetSchema をターゲットデータベース内の該当するスキーマ名に設定します。

次の例では、レプリケーションタスク T1 とその反対方向のレプリケーションタスク T2 の SourceSchema 設定と TargetSchema 設定が、反対方向の設定で指定されています。

タスク T1 の設定は、次のとおりです。

```
{
. . .

  "LoopbackPreventionSettings": {
    "EnableLoopbackPrevention": true,
    "SourceSchema": "LOOP-DATA",
    "TargetSchema": "loop-data"
  },

. . .
}
```

反対方向のタスク T2 の設定は、次のとおりです。

```
{
. . .

  "LoopbackPreventionSettings": {
    "EnableLoopbackPrevention": true,
    "SourceSchema": "loop-data",
    "TargetSchema": "LOOP-DATA"
  },

. . .
}
```

> **Note**
>
> AWS CLI を使用する場合、create-replication-task コマンドまたは modify-replication-task コマンドのみを使用して、双方向レプリケーションタスクで LoopbackPreventionSettings を設定します。

双方向レプリケーションの制限

AWS DMS の双方向レプリケーションには、次の制限があります。

- ループバック防止は、データ操作言語 (DML) ステートメントのみを追跡します。AWS DMS は、データ定義言語 (DDL) ループバックの防止をサポートしていません。これを行うには、双方向ペアのいずれかのタスクを、DDL ステートメントを除外するように設定します。
- ループバック防止を使用するタスクは、バッチでの変更のコミットはサポートしていません。ループバック防止を使用してタスクを設定するには、`BatchApplyEnabled` を `false` に設定します。
- DMS 双方向レプリケーションには、競合の検出や解決は含まれません。データの不整合を検出するには、両方のタスクでデータ検証を使用します。

[Modifying a task] (タスクの変更)

タスクの設定、テーブルのマッピング、その他の設定を変更する必要がある場合、タスクを変更できます。また、変更したタスクを実行する前に、移行前評価を有効にして実行することもできます。コンソールでタスクを選択し、[Modify] (変更) を選択して、タスクを変更できます。CLI コマンドまたは API オペレーション[ModifyReplicationTask] (レプリケーションタスク変更)を使用することもできます。。

タスクの変更には、いくつかの制限があります。これには以下が含まれます。

- タスクのソースまたはターゲットエンドポイントは変更できません。
- タスクの移行タイプを変更することはできません。
- 実行されたタスクを変更する場合、タスクのステータスは [Stopped] (停止済み) または [Failed](失敗) である必要があります。

タスクの移動

ユースケースに次のいずれかの状況が該当する場合は、タスクを別のレプリケーション インスタンスに移動できます。

- 現在、特定のタイプのインスタンスを使用していて、別のインスタンスタイプに切り替える必要があるとします。
- 現在のインスタンスは多くのレプリケーション タスクによって過負荷になり、ロードを複数のインスタンスに分割する必要があるとします。
- インスタンスストレージがいっぱいで、ストレージやコンピューティングのスケーリングの代替手段として、タスクをそのインスタンスからより強力なインスタンスに移動したいと考えています。

- AWS DMS の新しくリリースされた機能を使いたいが、新しいタスクを作成して移行を再開したくありません。代わりに、レプリケーション インスタンスをこの機能をサポートする新しい AWS DMS バージョンでスピンアップし、既存のタスクをそのインスタンスに移動したいとします。

コンソールでタスクを選択し、[Move] (移動) を選択して、タスクを移動できます。CLI コマンドまたは API オペレーション `MoveReplicationTask` を使用してタスクを移動することもできます。移動できるのは、任意のデータベースエンジンがターゲットエンドポイントとして指定されているタスクです。

ターゲット レプリケーション インスタンスに、移動するタスクを保存するのに十分なストレージ領域があることを確認します。それ以外の場合は、タスクを移動する前に、ターゲット レプリケーション インスタンス用のスペースを確保するようにストレージをスケーリングします。

また、ターゲットレプリケーションインスタンスが現在のレプリケーションインスタンスと同じまたはそれ以降の AWS DMS エンジンバージョンで作成されていることを確認します。

> **Note**
>
> - タスクを、現在常駐する同じレプリケーション インスタンスに移動することはできません。
> - タスクが移動している間、タスクの設定は変更できません。
> - 実行したタスクのステータスは移動する前に [Stopped] (停止) または [Failed] (失敗)、[Failed-move] (移動失敗)である必要があります。

DMS タスクの移動に関連するタスクのステータスは[Moving] (移動中) と[Failed-move](移動失敗) の 2 つあります。ステータス変更の詳細については、「タスクのステータス」をご参照ください。

タスクの移動後、移行前評価を有効にして実行し、移動したタスクを実行する前に、ブロックしている問題をチェックできます。

タスク実行中のテーブルの再ロード

タスクを実行中に、ソースのデータを使用して移行先データベースのテーブルを再ロードできます。タスク実行中にエラーが発生した際、またはパーティション操作によるデータの変更が生じた際に、テーブルの再ロードが必要になる場合があります。タスクから最大で 10 までのテーブルを再ロードできます。

テーブルを再ロードしてもタスクは停止しません。

テーブルを再ロードするには、以下の条件を適用する必要があります。

- タスクが実行中であることが必要です。
- タスクの移行メソッドは、全ロードまたは CDC による全ロードであることが必要です。
- テーブルは複製できません。
- AWS DMS では、以前に読み込んだテーブル定義が保持されます。再ロードオペレーション中に再作成されることはありません。テーブルが再ロードされる前に、ALTER TABLE ADD COLUMN、または DROP COLUMN などの DDL ステートメントを実行すると、再ロードオペレーションが失敗する可能性があります。

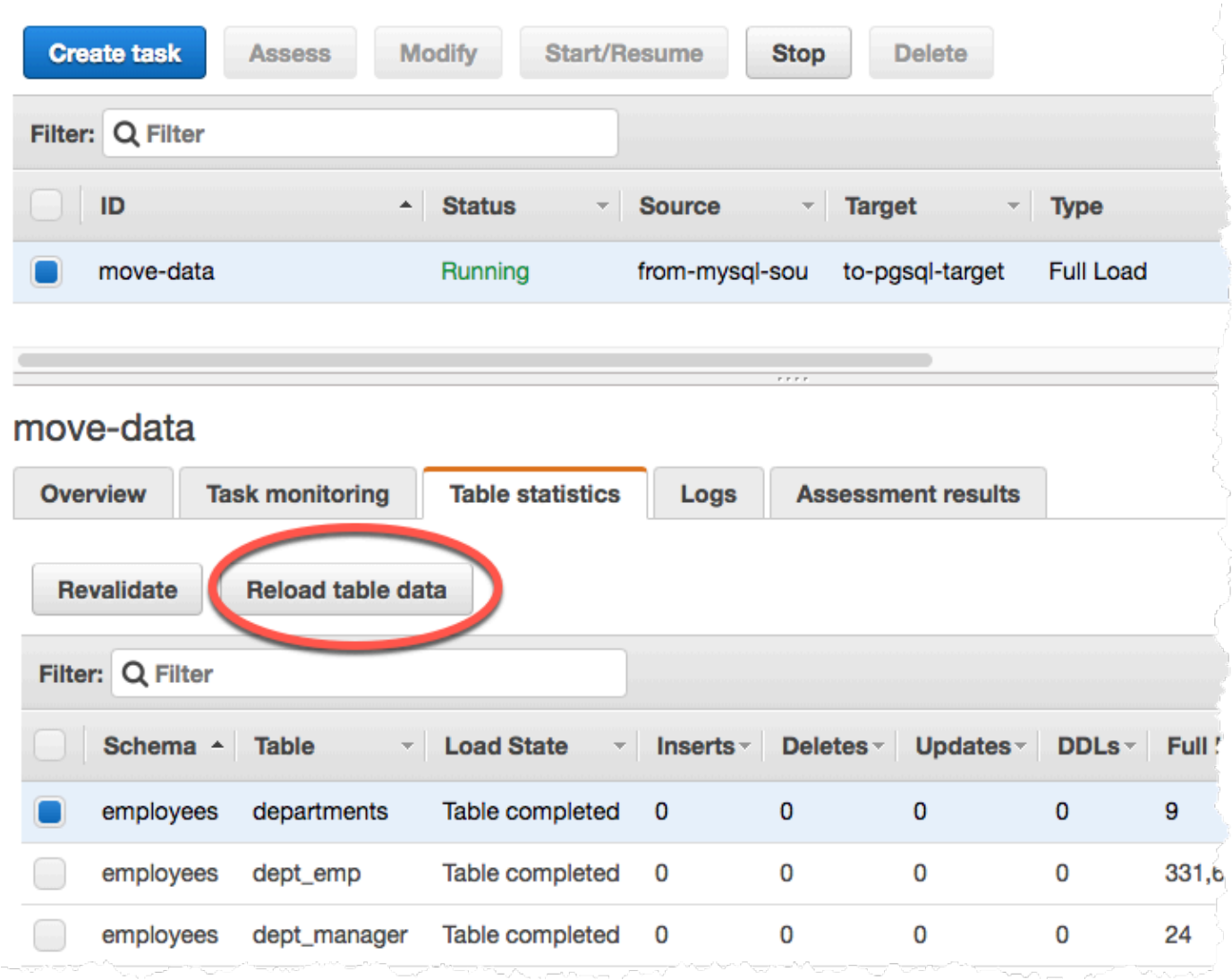
> **Note**
>
> DMS は、テーブルをリロードする前に `TargetTablePrepMode` 設定を適用します。`TargetTablePrepMode` を `DO_NOTHING` に設定した場合は、まずテーブルを手動で切り捨てる必要があります。

AWS Management Console

AWS DMS コンソールを使用してテーブルを再ロードする

1. AWS Management Console にサインインし、AWS DMS コンソール (https://console.aws.amazon.com/dms/v2/) を開きます。

   IAM ユーザーとしてサインインしている場合は、AWS DMS にアクセスするための適切なアクセス許可があることを確認します。必要なアクセス権限の詳細については、「AWS DMSの使用に必要な IAM アクセス許可」をご参照ください。
2. ナビゲーションペインから [Tasks] を選択します。
3. 再ロードするテーブルを含む実行中のタスクを選択します。
4. [テーブル統計] タブを選択します。



5. 再ロードするテーブルを選択します。タスクがすでに実行されていない場合は、テーブルを再ロードできません。
6. [Reload table data (テーブルデータの再ロード)] を選択します。

AWS DMS でテーブルの再ロードを準備中である場合は、コンソールでテーブルの状態が [Table is being reloaded (テーブルを再ロード中)] に変更されます。

テーブルマッピングを使用して、タスクの設定を指定する

テーブルマッピングは、データソース、ソーススキーマ、データ、そしてタスク実行中に必要なすべての変換を指定するための複数のルールタイプを使用します。テーブルマッピングを使用して、データベースで移行する個々のテーブルや、移行に使用するスキーマを指定できます。

テーブルマッピングを使用する場合、フィルタを使用して、テーブルの列からレプリケートするデータを指定できます。さらに、変換を使用して、選択したスキーマ、テーブル、またはビューをターゲットデータベースに書き込む前に変更できます。

トピック

> **Note**
>
> MongoDB ソース エンドポイントのテーブルマッピングを使用する場合、フィルタを使用してレプリケーション対象のデータを指定し、schema_name の代わりにデータベース名を指定します。デフォルト "%" を使用できます。

コンソールからテーブル選択および変換を指定する

を使用して AWS Management Console 、テーブルの選択や変換の指定など、テーブルマッピングを実行できます。コンソールで、[Where (場所)] セクションを使用してスキーマ、テーブル、およびアクションを含めるか除外するかを指定できます。[フィルタ] セクションを使用して、レプリケーションタスクに適用する、テーブルの列の名前と条件を指定します。これら 2 つのアクションを合わせて、選択ルールを作成します。

選択ルールを 1 つ以上指定した後に、変換をテーブルマッピングに含めることができます。変換を使用して、スキーマまたはテーブルの名前を変更したり、スキーマまたはテーブルにプレフィックスやサフィックスを追加したり、テーブルの列を削除したりできます。

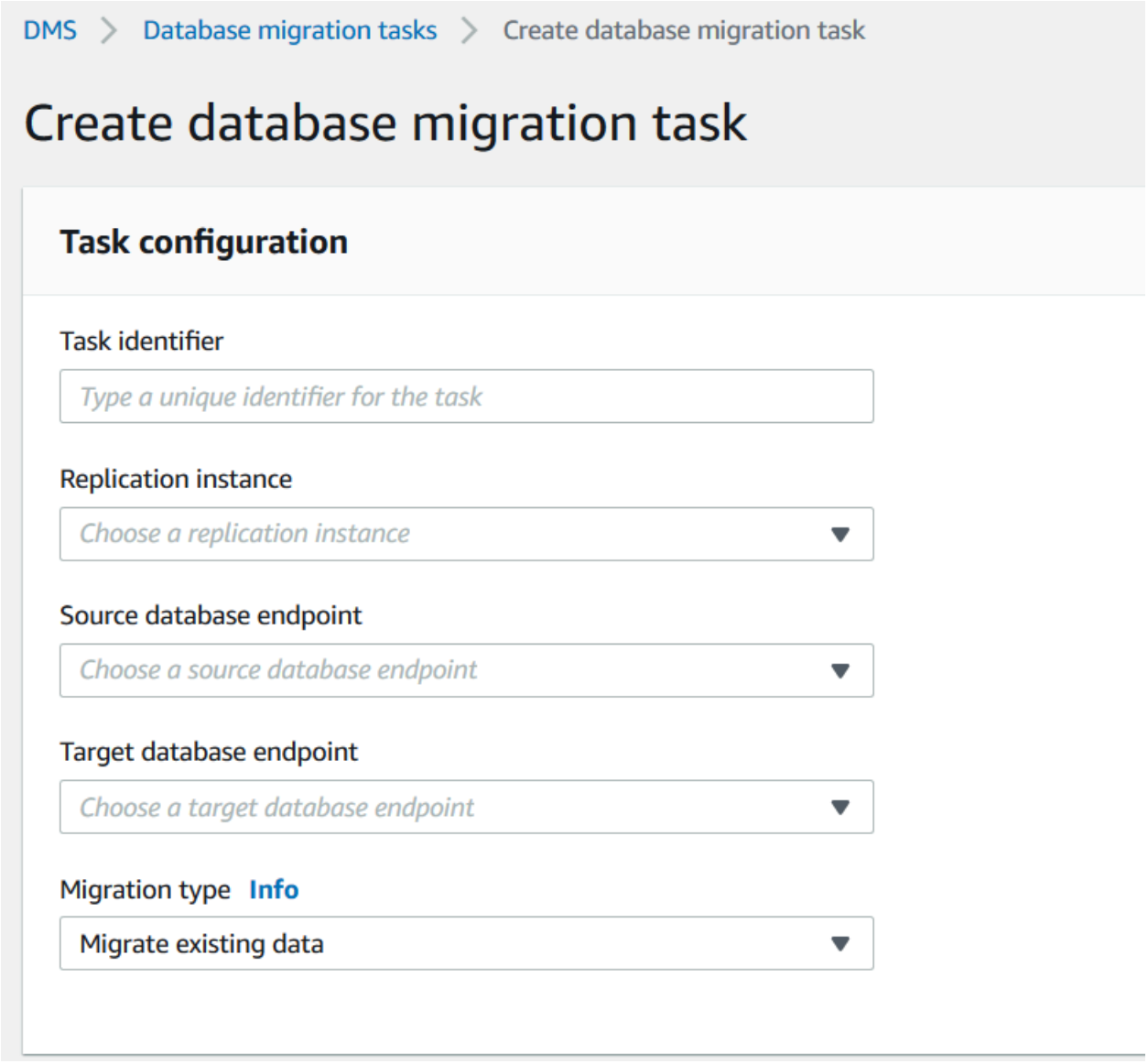
> **Note**
>
> AWS DMS は、スキーマレベル、テーブルレベル、または列レベルごとに複数の変換ルールをサポートしていません。

次の手順は、**EntertainmentAgencySample** という名前のスキーマ内の **Customers** という名前のテーブルに基づいて、選択ルールを設定する方法を示しています。

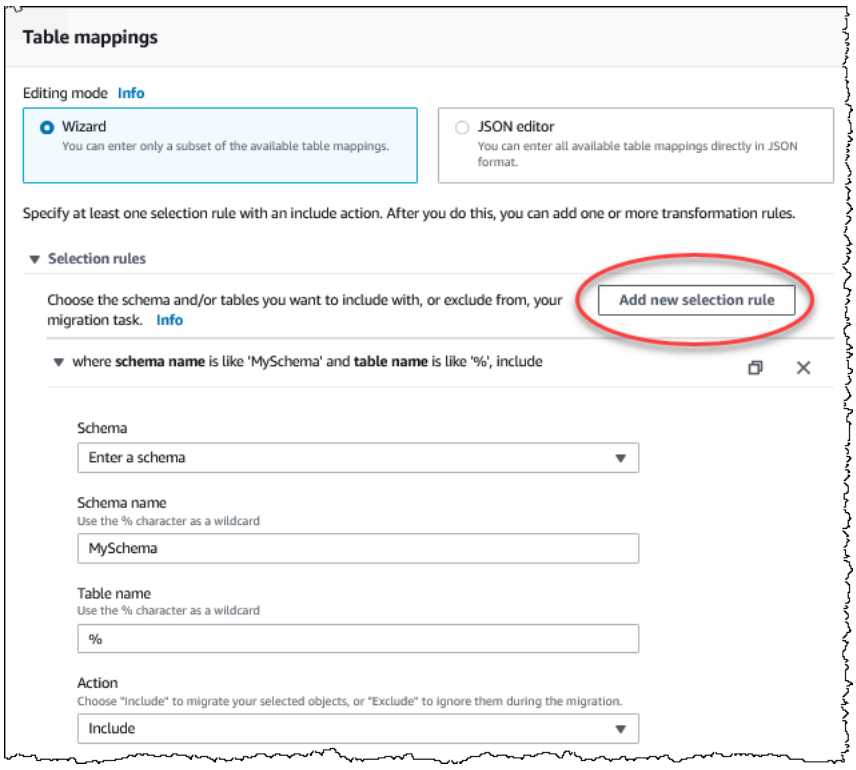
テーブルの選択を指定するには、コンソールを使用して、条件と変換でフィルタリングします。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。

   IAM ユーザーとしてサインインしている場合は、 AWS DMSにアクセスするための適切なアクセス許可があることを確認します。必要なアクセス権限の詳細については、「AWS DMSの使用に必要な IAM アクセス許可」をご参照ください。
2. [ダッシュボード] ページで、[データベース移行タスク] を選択します。
3. [タスクの作成] を選択します。
4. [タスクの設定] セクションで、[タスク識別子]、[レプリケーションインスタンス]、[ソースデータベースエンドポイント]、[ターゲットデータベースエンドポイント]、[移行タイプ] などのタスク情報を入力します。



5. [テーブルマッピング] セクションで、スキーマ名とテーブル名を選択します。スキーマ名またはテーブル名を指定するとき、「%」をワイルドカード値として使用できます。使用できるワイルドカードの詳細については、「the section called “テーブルマッピングのワイルドカード”」をご参照ください。フィルタを使用して、定義されたデータを含める、除外するかなど、実行するアクションを指定します。

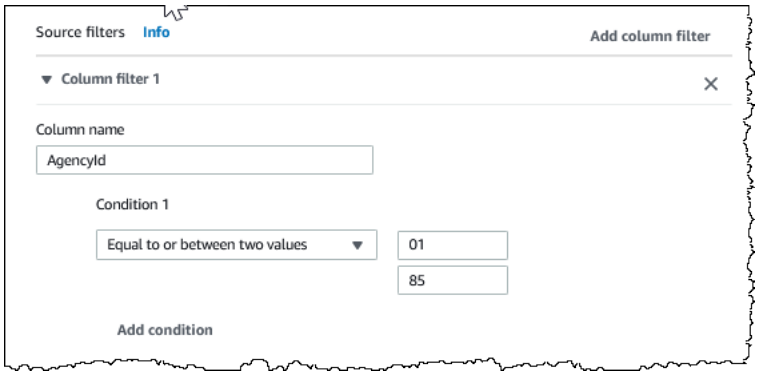


6. [Add column filter (列フィルタの追加)] および [条件の追加] リンクを使用して、フィルタ情報を指定します。

   a. まず、[Add column filter (列フィルタの追加)] を選択して列と条件を指定します。

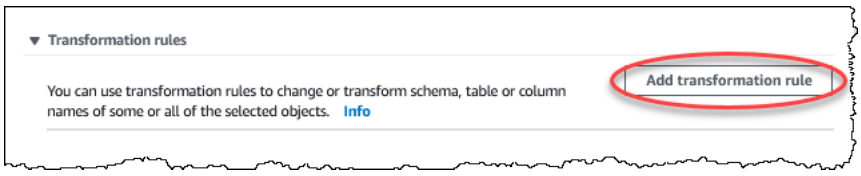
   b. [追加] を選択してその他の条件を追加します。

   以下の例は、**01** から **85** の間に **AgencyIDs** を含む **Customers** テーブルのフィルタを示しています。

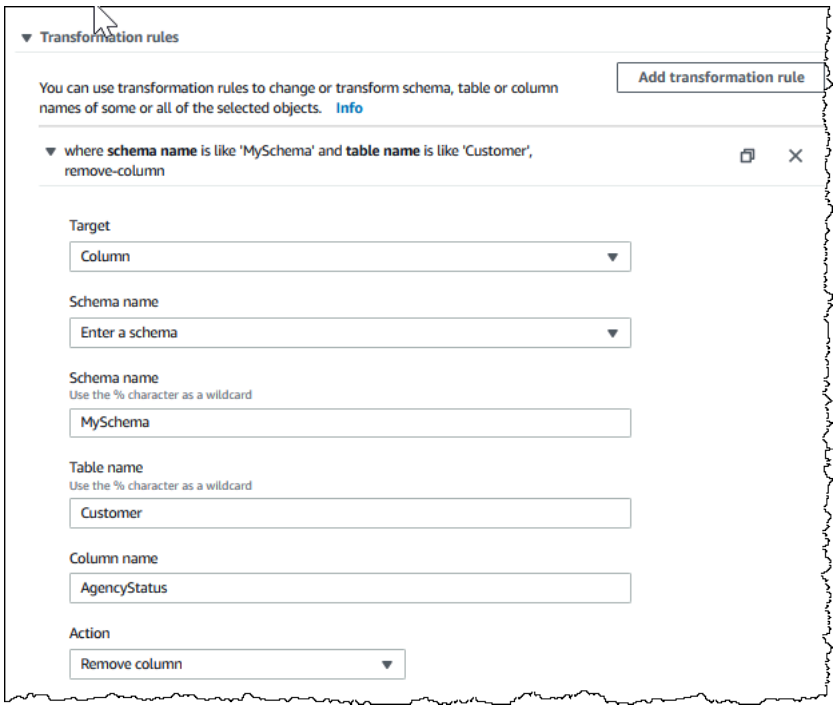


7. 必要な選択を作成したら [新しい選択ルールの追加] をクリックします。

8. 選択ルールを 1 つ以上作成したら、タスクに変換を追加できます。[add transformation rule (変換ルールの追加)] を選択します。



9. 変換を希望するターゲットを選択し、必要な追加情報を入力します。以下の例は、**Customer** テーブルから **AgencyStatus** 列を削除する変換を示しています。



10. [Add transformation rule] を選択します。
11. [Create task] (タスクの作成) を選択します。

> **Note**
>
> AWS DMS は、スキーマレベルまたはテーブルレベルごとに複数の変換ルールをサポートしていません。

JSON を使用するテーブル選択および変換を指定する

移行時に適用するテーブルマッピングを指定するには、JSON ファイルを作成します。コンソールを使用して移行タスクを作成する場合は、この JSON ファイルを参照するか、テーブルマッピングボックスに JSON を直接入力できます。CLI または API を使用して移行を実行する

場合、CreateReplicationTask または ModifyReplicationTask API オペレーションの TableMappings パラメータを使用してこのファイルを指定できます。

AWS DMS は、最大 2 MB のサイズのテーブルマッピング JSON ファイルのみを処理できます。DMS タスクを使用する間は、マッピングルールの JSON ファイルサイズを 2 MB の制限以下に維持することをお勧めします。これにより、タスクの作成中または変更中の予期しないエラーを回避できます。マッピングルールファイルが 2 MB の制限を超える場合は、テーブルを複数のタスクに分割してマッピングルールファイルのサイズを低減し、この制限以下となるようにすることをお勧めします。

使用するテーブル、ビュー、スキーマを指定できます。テーブル、ビュー、スキーマの変換を実行し、 AWS DMS が個々のテーブルとビューをロードする方法の設定を指定することもできます。これらのオプションのテーブルマッピングルールは、次のルールタイプを使用して作成します。

- selection ルール - ロードするソーステーブル、ビュー、スキーマのタイプと名前を識別します。詳細については、「選択ルールと選択アクション」を参照してください。
- transformation ルール - ターゲットにロードする前に、ソースの特定のソーステーブルとスキーマに対する特定の変更または追加を指定します。詳細については、「 変換ルールおよび変換アクション」を参照してください。

  また、新しい列と既存の列の内容を定義するには、変換ルール内で式を使用できます。詳細については、「変換ルール式を使用した列の内容の定義」を参照してください。
- table-settings ルール - DMS タスクが個々のテーブルのデータをロードする方法を指定します。詳細については、「テーブルとコレクション設定のルールとオペレーション」を参照してください。

> **Note**
>
> また、Amazon S3 ターゲットでは、post-processing ルールタイプおよび add-tag ルールアクションを使用して、選択したテーブルとスキーマに S3 オブジェクトにマッピングされたタグ付けを行うことができます。詳細については、「Amazon S3 オブジェクトのタグ付け」を参照してください。
> 次のターゲットでは、object-mapping ルールタイプを使用して選択したスキーマおよびテーブルをターゲットに移行する方法と場所を指定できます:
>
> - Amazon DynamoDB - 詳細については、DynamoDB にデータを移行するためのオブジェクトマッピングの使用をご覧ください。

- Amazon Kinesis - 詳細については、「Kinesis データストリームにデータを移行するためのオブジェクトマッピングの使用」をご参照ください。
- Apache Kafka - 詳細については、「データを Kafka トピックに移行するためのオブジェクトマッピングの使用」をご参照ください。

選択ルールと選択アクション

テーブルマッピングを使用すると、選択ルールと選択アクションを使用することで、使用するテーブル、ビューやスキーマを指定できます。選択ルールタイプを使用するテーブルマッピングルールの場合、次の値を適用できます。

パラメータ	使用できる値:	説明
`rule-type`	`selection`	選択ルール。テーブルマッピングを指定するときは、少なくとも 1 つの選択ルールを定義します。
`rule-id`	数値。	ルールを識別する一意の数値。
`rule-name`	英数字値。	ルールを特定する一意な名前。
`rule-action`	`include`, `exclude`, `explicit`	ルールによって選択されたオブジェクトを含めたり、除外する値。`explicit` を指定した場合、明示的に指定したテーブルおよびスキーマに該当する 1 つだけのオブジェクトを選択して含めることができます。
`object-locator`	以下のパラメータを使用するオブジェクト。 • `schema-name` – スキーマの名前。 • `table-name` - テーブルの名前。 • (オプション) `table-type` – `table | view | all`	ルールが適用されるスキーマ、テーブルまたはビューごとの名前。ルールにテーブルのみを含めるか、ビューのみを含めるか、テーブルとビューの両方を含めるかを指定することもできます。`rule-action` が `include` あるいは `exclude` の場合、`schema-name` および `table-`

パラメータ	使用できる値:	説明
	は、table-name がテーブルまたはビュー、テーブルとビューの両方を参照するかどうかを示します。デフォルトは table です。 AWS DMS は、全ロードタスクでのみビューをロードします。フルロードおよび変更データキャプチャ (CDC) タスクのみがある場合は、ビューをロードするように少なくとも 1 つの full-load-only タスクを設定します。 すべてのターゲットエンドポイントが、フルロード (Amazon OpenSearch Service など) であっても、ビューをレプリケーションのソースとして受け入れるわけではありません。ターゲット エンドポイントの制限を確認します。	name パラメータの一部あるいはすべての値に「%」パーセント記号をワイルドカードとして使用できます。使用できるワイルドカードの詳細については、「the section called “テーブルマッピングのワイルドカード”」をご参照ください。したがって、これらの項目には以下を対応させることができます。 • 単一のスキーマ内の単一のテーブルまたはビューまたはコレクション • 一部またはすべてのスキーマの単一のテーブルまたはビューまたはコレクション • 単一のスキーマの一部またはすべてのテーブルとビュー、または単一のデータベースのコレクション • 一部またはすべてのスキーマの一部またはすべてのテーブルとビュー、または、一部またはすべてのデータベースのコレクション rule-action が explicit の場合、1 つのテーブル、ビューとそのスキーマ (ワイルドカードなし) の正確な名前のみを指定できます。 ビューでサポートされているソースは次のとおりです。 • Oracle • Microsoft SQL Server

パラメータ	使用できる値:	説明
		• PostgreSQL • IBM Db2 LUW • IBM Db2 z/OS • SAP Adaptive Server Enterprise (ASE) • MySQL • AURORA • Aurora Serverless • MariaDB **Note** AWS DMS はソースビューをターゲットビューにロードしません。ソースビューは、ソースのビューと同じ名前のターゲット上の同等のテーブルにロードされます。 コレクションを含むデータベースでサポートされているソースは次のとおりです： • MongoDB • Amazon DocumentDB
`load-order`	正の整数。最大値は 2,147,483,647 です。	テーブルとビューを読み込むための優先度。値が大きいテーブルとビューが最初に読み込まれます。

パラメータ	使用できる値:	説明
filters	オブジェクトの配列。	ソースをフィルタリングする 1 つ以上のオブジェクト。ソースの単一の列でフィルタリングするオブジェクトパラメータを指定します。複数の列をフィルタリングする複数のオブジェクトを指定します。詳細については、「ソースフィルタの使用」をご参照ください。

Example スキーマ内のすべてのテーブルの移行

以下の例では、ソース内の Test という名前のスキーマからすべてのテーブルをターゲットエンドポイントに移行します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            },
            "rule-action": "include"
        }
    ]
}
```

Example スキーマの一部のテーブルの移行

以下の例では、ソース内の Test という名前のスキーマから、先頭が DMS のテーブルを除くすべてのテーブルをターゲットエンドポイントに移行します。

```
{
    "rules": [
        {
```

```
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "selection",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "DMS%"
            },
            "rule-action": "exclude"
        }
    ]
}
```

Example 単一のスキーマで指定した単一のテーブルの移行

以下の例では、ソース内の NewCust スキーマから Customer テーブルをターゲットエンドポイントに移行します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "NewCust",
                "table-name": "Customer"
            },
            "rule-action": "explicit"
        }
    ]
}
```

> **Note**
>
> 複数の選択ルールを指定することで、複数のテーブルとスキーマを明示的に選択できます。

Example 設定順でテーブルを移行

以下の例は、2 つのテーブルを移行します。テーブル `loadfirst` (優先度 1) はテーブル の前に初期化されます`loadsecond`。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "loadsecond"
            },
            "rule-action": "include",
            "load-order": "2"
        },
        {
            "rule-type": "selection",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "loadfirst"
            },
            "rule-action": "include",
            "load-order": "1"
        }
    ]
}
```

> **Note**
>
> load-order はテーブルの初期化で適用されます。MaxFullLoadSubTasks が 1 より大きい場合、後続のテーブルのロードは前回のテーブルのロードが完了するまで待機することはありません。

Example スキーマの一部のビューの移行

以下の例では、ソースの Test という名前のスキーマからターゲット内の同等のテーブルに、一部のビューを移行します。

```
{
   "rules": [
        {
           "rule-type": "selection",
           "rule-id": "2",
           "rule-name": "2",
           "object-locator": {
               "schema-name": "Test",
               "table-name": "view_DMS%",
               "table-type": "view"
            },
           "rule-action": "include"
        }
    ]
}
```

Example スキーマ内のすべてのテーブルとビューの移行

以下の例では、ソースの report という名前のスキーマからターゲット内の同等のテーブルに、すべてのテーブルとビューを移行します。

```
{
   "rules": [
        {
           "rule-type": "selection",
           "rule-id": "3",
           "rule-name": "3",
           "object-locator": {
               "schema-name": "report",
               "table-name": "%",
```

```
                "table-type": "all"
            },
            "rule-action": "include"
        }
    ]
}
```

テーブルマッピングのワイルドカード

このセクションでは、テーブル マッピングのスキーマおよびテーブル名を指定するときに使用できるワイルドカードについて説明します。

ワイルドカード	マッチ
%	0 文字以上
_	1 文字
[_]	リテラルアンダースコア文字
[ab]	文字のセット。例えば、[ab] は「a」または「b」のいずれかに一致します。
[a ~ d]	文字の範囲。例えば、[a-d] は「a」、「b」、「c」、「d」のいずれかに一致します。

ソースエンドポイントとターゲットエンドポイントが Oracle の場合、追加の接続属性 escapeCharacter を使用してエスケープ文字を指定できます。エスケープ文字を使用すると、指定したワイルドカード文字をワイルドカードではないかのように式で使用できます。例えば、このサンプルコードのように、「#」escapeCharacter=# を使用してワイルドカード文字を式内で通常の文字のように使用できます。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "542485267",
```

```
            "rule-name": "542485267",
            "object-locator": { "schema-name": "ROOT", "table-name": "TEST#_T%" },
            "rule-action": "include",
            "filters": []
        }
    ]
}
```

ここで、「#」エスケープ文字を使用すると、「_」ワイルドカード文字が通常の文字として機能します。 はROOT、 という名前のスキーマ内のテーブル AWS DMS を選択します。各テーブルにはプレフィックスTEST_Tとして という名前が付けられます。

変換ルールおよび変換アクション

選択したスキーマ、テーブルまたはビューに適用する変換を指定するには、変換アクションを使用します。変換ルールはオプションです。

制限事項

- 同じオブジェクト (スキーマ、テーブル、列、テーブルとテーブルスペース、またはインデックスとテーブルスペース) に対して複数の変換ルールアクションを適用することはできません。各変換アクションが異なるオブジェクトに対して適用される限り、複数の変換ルールアクションを任意のレベルに適用できます。
- 変換ルールのテーブル名と列名では大文字と小文字が区別されます。例えば、Oracle データベースまたは Db2 データベースのテーブル名と列名は大文字で指定する必要があります。
- 右から左へ記述する言語の列名の変換はサポートされていません。
- 変換は、名前に特殊文字 (#、\、/、-など) を含む列では実行できません。
- BLOB データ型 や CLOB データ型にマップされた列に対してサポートされている変換は、ターゲットでの列の削除のみです。
- AWS DMS では、2 つのソーステーブルを 1 つのターゲットテーブルにレプリケートすることはできません。 AWS DMS は、レプリケーションタスクの変換ルールに従って、テーブルからテーブル、および列から列にレコードをレプリケートします。重複を避けるため、オブジェクト名は一意である必要があります。

  例えば、ソーステーブルに ID という名前の列があり、対応するターゲットテーブルには id という既存の列があるとします。ルールで ADD-COLUMN ステートメントを使用して id という新しい列を追加し、SQLite ステートメントを使用してその列にカスタム値を設定すると、id という名前の重複したあいまいなオブジェクトが作成されます。これはサポートされません。

[値]

変換ルールタイプを使用するテーブルマッピングルールの場合、次の値を適用できます。

パラメータ	使用できる値:	説明
rule-type	transformation	選択ルールにより指定された各オブジェクトにルールを適用する値。特に明記されていない限り、transformation を使用します。
rule-id	数値。	ルールを識別する一意の数値。同じオブジェクト (スキーマ、テーブル、列、テーブル間スペース、インデックステーブルスペース) に複数の変換ルールを指定すると、変換ルールに低い rule-id AWS DMS が適用されます。
rule-name	英数字値。	ルールを特定する一意な名前。
object-locator	以下のパラメータを使用するオブジェクト。 • schema-name – スキーマの名前。MongoDB と Amazon DocumentDB エンドポイントの場合、これはコレクションの集合を保持するデータベースの名前です。 • table-name – テーブルまたはビューまたはコレクションの名前。 • table-tablespace-name - 既存のテーブルのテーブルスペース名。	ルールが適用される各スキーマ、テーブルまたはビュー、テーブルのテーブルスペース、インデックスのテーブルスペース、および列の名前。data-type を除く各 object-locator パラメータの値のすべてあるいは一部で、「%」パーセント記号をワイルドカードとして使用できます。したがって、これらの項目には以下を対応させることができます。 • 単一のスキーマ内の単一のテーブルまたはビュー • 一部またはすべてのスキーマの単一のテーブルまたはビュー

パラメータ	使用できる値:	説明
	• index-tablespace-name - 既存のインデックスのテーブルスペース名。 • column-name - 既存の列の名前。 • data-type - 既存の列データ型の名前。	• 単一のスキーマの一部またはすべてのテーブルとビュー • 一部またはすべてのスキーマの一部またはすべてのテーブルとビュー • 指定したテーブル、ビュー、スキーマ内の 1 つ以上の列。 • 複数の列が指定されている場合、指定された data-type を持つ列。data-type の指定可能な値については 、次の表で説明されている「data-type 」をご参照ください。 また、table-tablespace-name あるいは index-tablespace-name パラメータは、Oracle ソースのエンドポイントと一致させるためのみに使用できます。table-tablespace-name または index-tablespace-name を単一のルールで指定できますが、両方を指定することはできません。したがって、以下の項目のいずれかを一致させることができます。 • 1 つ、一部、あるいはすべてのテーブルのテーブルスペース • 1 つ、一部、あるいはすべてのインデックスのテーブルスペース

パラメータ	使用できる値:	説明
rule-action	add-column ,include-column , remove-column rename convert-lowercase ,convert-uppercase add-prefix ,remove-prefix , replace-prefix add-suffix ,remove-suffix , replace-suffix define-primary-key change-data-type add-before-image-columns	オブジェクトに適用する変換。すべて変換ルールアクションでは、大文字と小文字が区別されます。 rule-action パラメータの add-column 値は、テーブルに列を追加します。ただし、同じテーブルの既存の列と同じ名前の新しい列を追加することはできません。 data-type パラメータおよび expression パラメータとともに使用すると、add-column は新しい列データの値を指定します。 rule-action の change-data-type の値は、column ルールターゲットに対してのみ使用できます。 include-column パラメータのrule-action の値により、テーブルのモードを[drop all columns by default](デフォルトですべての列を削除する)と[include the columns specified](指定した列を含める)に変更します。複数の列は、include-column ルールを複数回呼び出してターゲットに含まれます。 ルールのスキーマ名またはテーブル名にワイルドカード (%) が含まれている場合は define-primary-key は使用できない。 既存のタスクの場合、remove-column 、rename、add-prefix な

パラメータ	使用できる値:	説明
		どのターゲットテーブルスキーマを変更する変換ルールアクションは、タスクを再開するまで有効になりません。変換ルールを追加した後でタスクを再開すると、変更した列で列データが欠落するなど、予期しない動作が発生する可能性があります。変換ルールが正しく動作するには、タスクを再起動する必要があります。
`rule-target`	`schema`, `table`, `column`, `table-tablespace` , `index-tablespace`	変換するオブジェクトのタイプ。 `table-tablespace` および `index-tablespace` の値は、Oracle ターゲットエンドポイントでのみ使用できます。 `object-locator` : `table-tablespace-name` または `index-tablespace-name` 名の一部として指定するパラメータの値を指定してください。
`value`	ターゲットタイプの名前付けルールに従った英数字値。	入力が必要なアクションの新しい値 (`rename` など)。
`old-value`	ターゲットタイプの名前付けルールに従った英数字値。	置き換えが必要なアクションの古い値 (`replace-prefix` など)。

パラメータ	使用できる値:	説明

data-type	type – rule-action が add-column である場合に使用するデータ型、または rule-action が change-data-type である場合は置換データ型。
	
	または、rule-action が change-data-type 、column-name の値が "%"、かつ既存のデータ型を識別するための追加の data-type パラメータが object-locator に含まれる場合の置換データ型の名前。
	
	AWS DMS は、次の DMS データ型の列データ型変換をサポートします。 "bytes", "date", "time", "datetime", "int1", "int2", "int4", "int8", "numeric", "real4", "real8", "string", "uint1", "uint2", "uint4", "uint8", "wstring", "blob", "nclob", "clob", "boolean", "set", "list" "map", "tuple"
	
	precision - 追加された列または置換データ型に精度がある場合は、精度を指定する整数値。
	
	scale – 追加された列または置換データ型に位取り、整数の値、または日時の値がある場合、位取りを指定する整数値
		次は、置換する既存のデータ型を指定する data-type パラメータの例です。
		
		{
		"rules": [{
		"rule-type": "selection",
		"rule-id": "1",
		"rule-name": "1",
		"object-locator": {
		"schema-name": "%",
		"table-name": "%"
		},
		"rule-action": "include"
		},
		{
		"rule-type": "transformation",
		"rule-id": "2",
		"rule-name": "2",
		"rule-target": "column",
		"object-locator": {
		"schema-name": "test",
		"table-name": "table_t",
		"column-name": "col10"
		},
		"rule-action": "change-data-type",
		"data-type": {
		"type": "string",
		"length": "4092",
		"scale": ""
		}
		}
		]
		}

パラメータ	使用できる値:	説明
	length – 新しい列データの長さ (add-column と共に使用する場合)	ここでは、table_t テーブルの col10 列が string データ型に変更されます。
expression	SQLite 構文に続く英数字の値。	rule-action を rename-schema に設定して使用すると、expression パラメータは新しいスキーマを指定します。rule-action を rename-table に設定して使用すると、expression は新しいテーブルを指定します。rule-action を rename-column に設定して使用すると、expression は新しい列の値を指定します。 rule-action を add-column に設定して使用すると、expression は新しい列を構成するデータを指定します。 このパラメータでは式のみがサポートされていることに注意する。演算子とコマンドはサポートされない。 変換ルールに式を使用する方法の詳細については、「変換ルール式を使用した列の内容の定義」をご参照ください。 SQLite 式の詳細については、「SQLite 関数を使用して式を構築する」を参照。

パラメータ	使用できる値:	説明
primary-key-def	以下のパラメータを使用するオブジェクト。 • name - テーブルまたはビューの新しいプライマリ キーまたは一意のインデックスの名前。 • (オプション) origin – 定義する一意のキーのタイプ： primary-key (デフォルト) または unique-index 。 • columns - プライマリ キーあるいは一意のインデックスに表示される順番の列の名前をリストした文字列の配列。	このパラメータでは、変換されたテーブルまたはビューの一意のキーの名前、タイプ、内容を定義できます。rule-action が define-primary-key に設定され、rule-target が table に設定されている場合にこの処理を行います。デフォルトでは、一意のキーはプライマリキーとして定義されます。

パラメータ	使用できる値:	説明
before-image-def	以下のパラメータを使用するオブジェクト。 • column-prefix - 列名の前に付加される値。デフォルト値は、BI_です。 • column-suffix - 列名に付加される値。デフォルトは空です。 • column-filter – pk-only (デフォルト)、non-lob (オプション)、all (オプション) のいずれかの値が必要です。	このパラメータは、前イメージ列を識別するための命名規則を定義し、ターゲット上に作成された前イメージ列を持つことができるソース列を識別するためのフィルターを指定します。rule-action が add-before-image-columns に設定され、rule-target が column に設定されている場合に、このパラメータを指定できます。 column-prefix と column-suffix の両方を空の文字列に設定しないでください。 [column-filter] では以下を選択します。 • pk-only - テーブルのプライマリキーの一部である列のみを追加します。 • non-lob - LOB タイプではない列のみを追加します。 • all - 前イメージの値を持つ任意の列を追加します。 AWS DMS ターゲットエンドポイントの前イメージのサポートの詳細については、以下をご参照ください: • 前イメージを使用した Kinesis データストリームの CDC 行の元の値のターゲットとしての表示

パラメータ	使用できる値:	説明
		• ターゲットとして Apache Kafka の CDC 行の元の値を表示するために前イメージを使用

例

Example スキーマの名前変更

以下の例では、スキーマの名前をソースでの Test からターゲットでの Test1 に変更します。

```
{

    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "rename",
            "rule-target": "schema",
            "object-locator": {
                "schema-name": "Test"
            },
            "value": "Test1"
        }
    ]
}
```

Example テーブル名の変更

以下の例では、テーブルの名前をソースでの Actor からターゲットでの Actor1 に変更します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "rename",
            "rule-target": "table",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Actor"
            },
            "value": "Actor1"
        }
    ]
}
```

Example 列名の変更

以下の例では、テーブル Actor の列の名前をソースでの first_name からターゲットでの fname に変更します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
```

```
        },
         {
            "rule-type": "transformation",
            "rule-id": "4",
            "rule-name": "4",
            "rule-action": "rename",
            "rule-target": "column",
            "object-locator": {
                "schema-name": "test",
                "table-name": "Actor",
                "column-name" : "first_name"
            },
            "value": "fname"
        }
    ]
}
```

Example Oracle テーブルのテーブルスペースの名前変更

次の例では、Oracle ソースの Actor という名前のテーブルの SetSpace という名前のテーブルのテーブルスペースを、Oracle のターゲットエンドポイントで SceneTblSpace に名前変更します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Play",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "rename",
            "rule-target": "table-tablespace",
            "object-locator": {
                "schema-name": "Play",
                "table-name": "Actor",
```

```
                "table-tablespace-name": "SetSpace"
            },
            "value": "SceneTblSpace"
        }
    ]
}
```

Example Oracle インデックスのテーブルスペースの名前変更

以下の例では、Oracle ソースの Actor という名前のテーブルの SetISpace という名前のインデックスのテーブルスペースを、Oracle のターゲットエンドポイントで SceneIdxSpace に名前変更します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Play",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "rename",
            "rule-target": "table-tablespace",
            "object-locator": {
                "schema-name": "Play",
                "table-name": "Actor",
                "table-tablespace-name": "SetISpace"
            },
            "value": "SceneIdxSpace"
        }
    ]
}
```

Example 列の追加

次の例では、スキーマ test のテーブル Actor に datetime 列を追加します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "add-column",
            "rule-target": "column",
            "object-locator": {
                "schema-name": "test",
                "table-name": "actor"
            },
            "value": "last_updated",
            "data-type": {
                "type": "datetime",
                "precision": 6
            }
        }
    ]
}
```

Example 列の削除

以下の例では、ソース内の Actor という名前のテーブルを変換し、先頭文字が col のすべての列をターゲットから削除します。

```
{
  "rules": [{
  "rule-type": "selection",
```

```
  "rule-id": "1",
  "rule-name": "1",
  "object-locator": {
   "schema-name": "test",
   "table-name": "%"
  },
  "rule-action": "include"
 }, {
  "rule-type": "transformation",
  "rule-id": "2",
  "rule-name": "2",
  "rule-action": "remove-column",
  "rule-target": "column",
  "object-locator": {
   "schema-name": "test",
   "table-name": "Actor",
   "column-name": "col%"
  }
 }]
 }
```

Example [Convert to lowercase] (小文字に変換)

以下の例では、テーブルの名前をソースでの ACTOR からターゲットでの actor に変換します。

```
{
 "rules": [{
  "rule-type": "selection",
  "rule-id": "1",
  "rule-name": "1",
  "object-locator": {
   "schema-name": "test",
   "table-name": "%"
  },
  "rule-action": "include"
 }, {
  "rule-type": "transformation",
  "rule-id": "2",
  "rule-name": "2",
  "rule-action": "convert-lowercase",
  "rule-target": "table",
  "object-locator": {
   "schema-name": "test",
   "table-name": "ACTOR"
```

```
  }
 }]
}
```

Example 大文字への変換

次の例では、すべてのテーブルおよびすべてのスキーマ内のすべての列を、ソースでの小文字からターゲットでの大文字に変換します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "convert-uppercase",
            "rule-target": "column",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%",
                "column-name": "%"
            }
        }
    ]
}
```

Example プレフィックスの追加

以下の例では、ソース内のすべてのテーブルを変換し、ターゲットではそれらのテーブルにプレフィックス DMS_ を追加します。

```
{
```

```
  "rules": [{
  "rule-type": "selection",
  "rule-id": "1",
  "rule-name": "1",
  "object-locator": {
   "schema-name": "test",
   "table-name": "%"
  },
  "rule-action": "include"
 }, {
  "rule-type": "transformation",
  "rule-id": "2",
  "rule-name": "2",
  "rule-action": "add-prefix",
  "rule-target": "table",
  "object-locator": {
   "schema-name": "test",
   "table-name": "%"
  },
  "value": "DMS_"
 }]

}
```

Example プレフィックスの置き換え

以下の例では、ソースでプレフィックス Pre_ を含むすべての列を変換し、ターゲットではプレフィックスを NewPre_ に置き換えます。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
```

```
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "replace-prefix",
            "rule-target": "column",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%",
                "column-name": "%"
            },
            "value": "NewPre_",
            "old-value": "Pre_"
        }
    ]
}
```

Example サフィックスの削除

以下の例では、ソース内のすべてのテーブルを変換し、ターゲットではそれらのテーブルからサフィックス _DMS を削除します。

```
{
 "rules": [{
  "rule-type": "selection",
  "rule-id": "1",
  "rule-name": "1",
  "object-locator": {
   "schema-name": "test",
   "table-name": "%"
  },
  "rule-action": "include"
 }, {
  "rule-type": "transformation",
  "rule-id": "2",
  "rule-name": "2",
  "rule-action": "remove-suffix",
  "rule-target": "table",
  "object-locator": {
   "schema-name": "test",
   "table-name": "%"
  },
  "value": "_DMS"
 }]
}
```

Example プライマリキーの定義

次の例では、ターゲットエンドポイントに移行した ITEM テーブルの 3 つの列の ITEM-primary-key という名前のプライマリキーを定義します。

```
{
 "rules": [{
  "rule-type": "selection",
  "rule-id": "1",
  "rule-name": "1",
  "object-locator": {
   "schema-name": "inventory",
   "table-name": "%"
  },
  "rule-action": "include"
 }, {
  "rule-type": "transformation",
  "rule-id": "2",
  "rule-name": "2",
  "rule-action": "define-primary-key",
  "rule-target": "table",
  "object-locator": {
   "schema-name": "inventory",
   "table-name": "ITEM"
  },
  "primary-key-def": {
   "name": "ITEM-primary-key",
   "columns": [
    "ITEM-NAME",
    "BOM-MODEL-NUM",
    "BOM-PART-NUM"
   ]
              }
 }]
}
```

Example 一意のインデックスの定義

次の例では、ターゲットエンドポイントに移行した ITEM テーブルの 3 つの列の ITEM-unique-idx という名前の一意のインデックスを定義します。

```
{
```

```
 "rules": [{
  "rule-type": "selection",
  "rule-id": "1",
  "rule-name": "1",
  "object-locator": {
   "schema-name": "inventory",
   "table-name": "%"
  },
  "rule-action": "include"
 }, {
  "rule-type": "transformation",
  "rule-id": "2",
  "rule-name": "2",
  "rule-action": "define-primary-key",
  "rule-target": "table",
  "object-locator": {
   "schema-name": "inventory",
   "table-name": "ITEM"
  },
  "primary-key-def": {
   "name": "ITEM-unique-idx",
   "origin": "unique-index",
   "columns": [
    "ITEM-NAME",
    "BOM-MODEL-NUM",
    "BOM-PART-NUM"
   ]
              }
 }]
}
```

Example ターゲット列のデータ型の変更

次の例では、SALE_AMOUNT という名前のターゲット列のデータ型を既存のデータ型から int8 に変更します。

```
{
    "rule-type": "transformation",
    "rule-id": "1",
    "rule-name": "RuleName 1",
    "rule-action": "change-data-type",
    "rule-target": "column",
    "object-locator": {
```

```
        "schema-name": "dbo",
        "table-name": "dms",
        "column-name": "SALE_AMOUNT"
    },
    "data-type": {
        "type": "int8"
    }
}
```

Example 前イメージ列の追加

emp_no という名前のソース列の場合、次の例の変換ルールによって、ターゲットに BI_emp_no という名前の新しい列が追加されます。

```
{
 "rules": [{
   "rule-type": "selection",
   "rule-id": "1",
   "rule-name": "1",
   "object-locator": {
    "schema-name": "%",
    "table-name": "%"
   },
   "rule-action": "include"
  },
  {
   "rule-type": "transformation",
   "rule-id": "2",
   "rule-name": "2",
   "rule-target": "column",
   "object-locator": {
    "schema-name": "%",
    "table-name": "employees"
   },
   "rule-action": "add-before-image-columns",
   "before-image-def": {
    "column-prefix": "BI_",
    "column-suffix": "",
    "column-filter": "pk-only"
   }
  }
 ]
}
```

ここでは、次のステートメントは、対応する行の BI_emp_no 列に 1 を代入します。

```
UPDATE employees SET emp_no = 3 WHERE BI_emp_no = 1;
```

サポートされている AWS DMS ターゲットに CDC 更新を書き込む場合、 BI_emp_no列では、どの行の値が更新されたかをemp_no列で判断できます。

変換ルール式を使用した列の内容の定義

新しい列と既存の列の内容を定義するには、変換ルール内で式を使用します。たとえば、式を使用すると、列を追加したり、ソーステーブルヘッダーをターゲットにレプリケートしたりできます。また、式を使用して、ターゲットテーブルのレコードに、ソースで挿入済み、更新済み、または削除済みというフラグを付けることもできます。

トピック

- 式を使用した列の追加
- 式を使用したターゲットレコードのフラグ付け
- 式を使用したソーステーブルヘッダーのレプリケート
- SQLite 関数を使用して式を構築する
- 式を使用したターゲット テーブルへのメタデータ追加

式を使用した列の追加

変換ルールの式を使用してテーブルに列を追加するには、add-column ルールアクションと column ルールターゲットを使用します。

次の使用例は、ITEM テーブルに新しい列を追加します。新しい列名が FULL_NAME に設定されます。データ型は string、長さは 50 文字です。式は、2 つの既存の列の値 FIRST_NAME および LAST_NAME を連結して FULL_NAME を評価します。schema-name および table-name、式パラメータは、ソース データベーステーブル内のオブジェクトを参照します。Value と data-type ブロックは、ターゲット データベーステーブル内のオブジェクトを参照します。

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
```

```
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "add-column",
            "rule-target": "column",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "ITEM"
            },
            "value": "FULL_NAME",
            "expression": "$FIRST_NAME||'_'||$LAST_NAME",
            "data-type": {
                 "type": "string",
                 "length": 50
            }
        }
    ]
}
```

式を使用したターゲットレコードのフラグ付け

ターゲットテーブルのレコードにソーステーブルで挿入済み、更新済み、または削除済みというフラグを付けるには、変換ルールで式を使用します。式は、operation_indicator 関数を使用してレコードにフラグを付けます。ソースから削除されたレコードは、ターゲットから削除されません。代わりに、ターゲットレコードには、ソースから削除されたことを示すユーザー指定の値でフラグが付けられます。

> **Note**
>
> operation_indicator 関数は、ソースデータベースとターゲットデータベースの両方にプライマリキーがあるテーブルでのみ機能します。

たとえば、次の変換ルールは、まずターゲットテーブルに新しい Operation 列を追加します。次に、ソーステーブルからレコードが削除されるたびに、値 D で列を更新します。

```
{
      "rule-type": "transformation",
      "rule-id": "2",
      "rule-name": "2",
      "rule-target": "column",
      "object-locator": {
        "schema-name": "%",
        "table-name": "%"
      },
      "rule-action": "add-column",
      "value": "Operation",
      "expression": "operation_indicator('D', 'U', 'I')",
      "data-type": {
        "type": "string",
        "length": 50
      }
}
```

式を使用したソーステーブルヘッダーのレプリケート

デフォルトでは、ソーステーブルのヘッダーはターゲットにレプリケートされません。レプリケートするヘッダーを指定するには、テーブルの列ヘッダーを含む式を含む変換ルールを使用します。

式では、次の列ヘッダーを使用できます。

[Header] (ヘッダー)	継続的なレプリケーションの値	全ロードの値	データ型
AR_H_STREAM_POSITION	ソースからのストリーム位置の値。この値は、ソースエンドポイントに応じて、システム変更番号 (SCN) またはログシーケンス番号 (LSN) になります。	空の文字列。	STRING
AR_H_TIMESTAMP	変更の時刻を示すタイムスタンプ。	現在のデータがターゲットに到着した時	DATETIME (スケール=7)

[Header] (ヘッダー)	継続的なレプリケーションの値	全ロードの値	データ型
		刻を示すタイムスタンプ。	
AR_H_COMMIT_TIMESTAMP	コミットの時刻を示すタイムスタンプ。	現在の時刻を示すタイムスタンプ。	DATETIME (スケール=7)
AR_H_OPERATION	INSERT、UPDATE、または DELETE	INSERT	STRING
AR_H_USER	変更を行ったユーザーについて、ソースから提供されるユーザー名、ID、またはその他の情報。 このヘッダーは、SQL Server および Oracle (バージョン 11.2.0.3 以上) のソースエンドポイントでのみサポートされます。	オブジェクトに適用する変換。変換ルールアクションでは、大文字と小文字が区別されます。	STRING
H_H_CHANGE_SEQ	タイムスタンプと自動インクリメント番号で構成される、ソース データベースの一意のインクリメント番号。値は、ソース データベースシステムによって異なります。	空の文字列。	STRING

次の例では、ソースからのストリーム位置の値を使用して、ターゲットに新しい列を追加します。SQL サーバー の場合、ストリーム位置の値は、ソース エンドポイントの SCN です。Oracle の場合、ストリーム位置の値はソースエンドポイントの LSN です。

```
{
      "rule-type": "transformation",
     "rule-id": "2",
      "rule-name": "2",
      "rule-target": "column",
      "object-locator": {
        "schema-name": "%",
        "table-name": "%"
      },
      "rule-action": "add-column",
      "value": "transact_id",
      "expression": "$AR_H_STREAM_POSITION",
      "data-type": {
        "type": "string",
        "length": 50
      }
    }
```

次の使用例は、ソースから一意の増分番号を持つ新しい列をターゲットに追加します。この値は、タスクレベルでの 35 桁の一意の数字を表します。最初の 16 桁はタイムスタンプの一部であり、最後の 19 桁は DBMS によって増分された record_id 番号です。

```
{
"rule-type": "transformation",
"rule-id": "2",
"rule-name": "2",
"rule-target": "column",
"object-locator": {
"schema-name": "%",
"table-name": "%"
},
"rule-action": "add-column",
"value": "transact_id",
"expression": "$AR_H_CHANGE_SEQ",
"data-type": {
"type": "string",
"length": 50
}
```

```
}
```

SQLite 関数を使用して式を構築する

テーブル設定を使用して、指定されたオペレーションで選択されたテーブルまたはビューに適用する設定を指定します。テーブル設定ルールはオプションです。

> **Note**
>
> MongoDB と DocumentDB データベースでは、テーブルやビューという概念の代わりに、コレクションにまとめられたドキュメントとしてデータレコード保存します。そのため、MongoDB ソースまたは DocumentDB ソースから移行する場合は、テーブルやビューではなく、選択したコレクションに対する並列ロード設定の範囲セグメンテーションタイプを考慮します。

トピック

- CASE 式の使用
- 例

変換ルール式を作成するために使用できる文字列関数を以下に示します。

文字列関数	説明
`lower(x)`	`lower(x)` 関数はすべての文字が小文字に変換された文字列のコピー *x* を返します。デフォルトの、組み込み `lower` 関数は ASCII 文字に対してのみ機能します。
`upper(x)`	`upper(x)` 関数は、すべての文字が小文字に変換された文字列 *x* のコピーを返す。デフォルトの、組み込み `upper` 関数は ASCII 文字に対してのみ機能します。
`ltrim(x,y)`	`ltrim(x,y)` 関数は、x の左辺から y に現れるすべての文字を削除して形成された文字列を返します。y の値がない場合、`ltrim(x)` は x の左辺からスペースを削除します。

文字列関数	説明
replace(*x*,*y*,*z*)	replace(*x*,*y*,*z*) 関数は、文字列 x 内の文字列 y の出現ごとに文字列 z を代入して形成された文字列を返します。
rtrim(*x*,*y*)	rtrim(*x*,*y*) 関数は、x の右側から y に現れるすべての文字を削除して形成された文字列を返します。y の値がない場合、rtrim(x) は x の右側からスペースを削除します。
substr(*x*,*y*,*z*)	substr(*x*,*y*,*z*) 関数は、*y* 番目の文字で始まる、文字長 *z* の入力文字列 *x* の部分文字列を返します。 *z* が省略され、substr(*x*,*y*) は、*y* 番目の文字で始まる文字列 *x* の最後を通してすべての文字を返します。*x* の左端の文字は番号 1 です。*y* が負の場合、部分文字列の最初の文字は左ではなく右から数えて見つけられます。*z* が負の場合、*y* 番目の文字より前にある abs(z) 文字が返されます。*x* が文字列の場合、文字のインデックスは実際の UTF-8 文字を参照します。*x* が BLOB の場合、インデックスはバイトを参照します。
trim(*x*,*y*)	trim(*x*,*y*) 関数は、*x* の両側から *y* に表示されるすべての文字を削除して形成された文字列を返します。。*y* に値がない場合、trim(*x*)は、*x* の両側からスペースを削除します。

変換ルール式の作成に使用できる LOB 関数を以下に示します。

LOB 関数	説明
hex(x)	hex 関数は BLOB を引数として受け取り、BLOB コンテンツの大文字 16 進文字列バージョンを返します。
randomblob (*N*)	randomblob(*N*) 関数は疑似乱数バイトを含む *N* バイト BLOB を返します。*N* が 1 未満の場合、1 バイトのランダム BLOB が返されます。
zeroblob(N)	zeroblob(*N*) 関数は、0x00 の *N* バイトで構成される BLOB を返します。

変換ルール式の作成に使用できる数値関数を以下に示します。

数値関数	説明
abs(*x*)	abs(*x*) 関数は、数値引数 *x* の絶対値を返します。abs(*x*) 関数は *x* が NULL の場合 NULL を返します。abs(*x*) 関数は、*x* が数値に変換できない文字列または BLOB の場合に 0.0 を返します。
random()	random 関数は、-9,223,372,036,854,775,808 から +9,223,372,036,854,775,807 の擬似乱数を返します。
round (*x*,*y*)	round (*x*,*y*) 関数は四捨五入して小数点の右側の *y* 桁までとした浮動小数点値 *x* を返します。*y* に値がない場合、0 と仮定されます。
max (*x*,*y*...)	複数引数 max 関数は、最大値を持つ引数を返すか、引数が NULL の場合は NULL を返します。 max 関数は、引数を左から右へ検索し照合関数を定義する引数を探します。見つかった場合は、その照合関数をすべての文字列比較に使用します。照合関数を定義する max への引数がない場合、BINARY の照合機能が使用されます。max 関数は、複数の引数を持つ場合は単純な関数ですが、引数が 1 つの場合は集計関数として動作します。
min (*x*,*y*...)	複数 引数min 関数は、最小値を持つ引数を返します。 min 関数は、引数を左から右へ検索し照合関数を定義する引数を探します。見つかった場合は、その照合関数をすべての文字列比較に使用します。照合関数を定義する min への引数がない場合、BINARY の照合機能が使用されます。min 関数は、複数の引数を持つ場合は単純な関数ですが、引数が 1 つの場合は集計関数として動作します。

変換ルール式を作成するために使用できる NULL チェック関数を以下に示します。

NULL チェック関数	説明
coalesce (*x*,*y*...)	coalesce 関数は、最初の非 NULL 引数のコピーを返しますが、すべての引数が NULL の場合は NULL を返します。coalesce 関数には、少なくとも 2 つの引数があります。
ifnull(*x*,*y*)	ifnull 関数は、最初の非 NULL 引数のコピーを返しますが、両方の引数が NULL の場合は NULL を返します。ifnull 関数にはちょうど 2 つの引数があります。ifnull 関数は 2つの引数を持つ coalesce と同じです。
nullif(*x*,*y*)	nullif(*x*,*y*) 関数は、引数が異なる場合は最初の引数のコピーを返しますが、引数が同じ場合は NULL を返します。 nullif(*x*,*y*) 関数は、引数を左から右へ検索し照合関数を定義する引数を探します。見つかった場合は、その照合関数をすべての文字列比較に使用します。nullifのどちらの引数も照合関数を定義していない場合、BINARY 照合機能が使用されます。

変換ルール式を作成するために使用できる日付と時刻関数を以下に示します。

日付および時刻関数	説明
date(*timestring* , *modifier*, *modifier*...)	date 関数は、YYYY-MM-DD 形式で日付を返します。
time(*timestring* , *modifier*, *modifier*...)	time 関数は、HH: MM: SS の形式で時刻を返します。
datetime(*timestring* , *modifier*, *modifier*...)	datetime 関数は、YYYY-MM-DD HH: MM: SS の形式で日付と時刻を返します。
julianday(*timestring* , *modifier*, *modifier*...)	julianday 関数は、グリニッジで紀元前4714年11月24日の正午からの日数を返します。

日付および時刻関数	説明
strftime(*format*, *timestring* , *modifier*, *modifier*...)	strftime 関数は、次のいずれかの変数を使用して、最初の引数として指定された書式文字列に従って日付を返します： %d: 月の日 %H: 時間 00—24 %f: ** 分数秒 SS.SSS %j: 年始からの日 001 ~ 366 %J: ** ユリウス日数 %m: 月 01—12 %M: 分 00 ~ 59 %s: 1970-01-01 からの秒 %S: 秒 00 ~ 59 %w: 週初からの日 0 ~ 6、日曜日 == 0) %W: 00-53 の週 %Y: 0000—9999 年 %%: %

変換ルール式を作成するために使用できるハッシュ関数を以下に示します。

ハッシュ関数	説明
hash_sha256(*x*)	hash 関数は、(SHA-256 アルゴリズムを使用して)入力列のハッシュ値を生成し、生成されたハッシュ値の 16 進値を返します。 式内の hash 関数を使用するには、hash_sha256(*x*) を式に追加し、*x* をソース列名で置換します。

CASE 式の使用

SQLite CASE 式は、条件のリストを評価し、結果に基づいて式を返します。構文を以下に示します。

```
    CASE case_expression
     WHEN when_expression_1 THEN result_1
     WHEN when_expression_2 THEN result_2
     ...
     [ ELSE result_else ]
    END

# Or

     CASE
     WHEN case_expression THEN result_1
     WHEN case_expression THEN result_2
     ...
     [ ELSE result_else ]
    END
```

例

Example ケース条件を使用してターゲット テーブルに新しい文字列列を追加する

例えば、次の変換ルールは、まずターゲットテーブル employee に新しい emp_seniority 列を追加します。給与列に SQLite round 関数を使って、給与が 20,000 に等しいか超えるかをチェックするケース条件を指定します。そうであれば、列は値 SENIOR を取得し、それ以外は何でもその値が JUNIOR となります。

```
  {
      "rule-type": "transformation",
      "rule-id": "2",
      "rule-name": "2",
      "rule-action": "add-column",
      "rule-target": "column",
      "object-locator": {
        "schema-name": "public",
        "table-name": "employee"
      },
      "value": "emp_seniority",
      "expression": " CASE WHEN round($emp_salary)>=20000 THEN ‘SENIOR’ ELSE ‘JUNIOR’
 END",
```

```
      "data-type": {
        "type": "string",
        "length": 50
      }

  }
```

Example ターゲット テーブルに新しい日付列を追加する

次の使用例は、ターゲット テーブル , createdate, に新しい日付列を追加します employee 。SQLite の日付関数 datetime を使用する場合、挿入された各行について、新しく作成されたテーブルに日付が追加されます。

```
{
    "rule-type": "transformation",
    "rule-id": "2",
    "rule-name": "2",
    "rule-action": "add-column",
    "rule-target": "column",
    "object-locator": {
      "schema-name": "public",
      "table-name": "employee"
    },
    "value": "createdate",
    "expression": "datetime ()",
    "data-type": {
      "type": "datetime",
      "precision": 6
    }
}
```

Example ターゲット テーブルに新しい数値列を追加する

次の使用例は、ターゲット テーブル rounded_emp_salary に新しい列 employee を追加します。SQLite round 関数を使って四捨五入された給与を追加します。

```
{
    "rule-type": "transformation",
    "rule-id": "2",
    "rule-name": "2",
    "rule-action": "add-column",
    "rule-target": "column",
```

```
    "object-locator": {
      "schema-name": "public",
      "table-name": "employee"
    },
    "value": "rounded_emp_salary",
    "expression": "round($emp_salary)",
    "data-type": {
      "type": "int8"
    }
}
```

Example ハッシュ関数を使用してターゲット テーブルに新しい文字列カラムを追加する

次の使用例は、ターゲット テーブル employee に新しい列 hashed_emp_number を追加します。SQLite hash_sha256(*x*) 関数は、ソース列 emp_number のターゲットにハッシュ値を作成します。

```
{
    "rule-type": "transformation",
    "rule-id": "2",
    "rule-name": "2",
    "rule-action": "add-column",
    "rule-target": "column",
    "object-locator": {
      "schema-name": "public",
      "table-name": "employee"
    },
    "value": "hashed_emp_number",
    "expression": "hash_sha256($emp_number)",
    "data-type": {
      "type": "string",
      "length": 64
    }
}
```

式を使用したターゲット テーブルへのメタデータ追加

次の式を使用して、メタデータ情報をターゲット テーブルに追加できます :

- $AR_M_SOURCE_SCHEMA – ソーススキーマの名前。
- $AR_M_SOURCE_TABLE_NAME – ソーステーブルの名前。
- $AR_M_SOURCE_COLUMN_NAME – ソーステーブルの列の名前。

- $AR_M_SOURCE_COLUMN_DATATYPE – ソーステーブルの列のデータ型。

Example ソースのスキーマ名を使用してスキーマ名の列を追加する

次の例では、ソースからのスキーマ名を使用して、ターゲットに新しい列 schema_name を追加します。

```
{
    "rule-type": "transformation",
    "rule-id": "2",
    "rule-name": "2",
    "rule-action": "add-column",
    "rule-target": "column",
    "object-locator": {
      "schema-name": "%",
      "table-name": "%"
    },
    "rule-action": "add-column",
    "value":"schema_name",
    "expression": "$AR_M_SOURCE_SCHEMA",
    "data-type": {
       "type": "string",
       "length": 50
    }
}
```

テーブルとコレクション設定のルールとオペレーション

テーブル設定を使用して、指定されたオペレーションで選択されたテーブルまたはビューに適用する設定を指定します。エンドポイントと移行の要件に応じてテーブル設定ルールはオプションです。

テーブルやビューを使用する代わりに、MongoDB および Amazon DocumentDB データベースでは、データレコードをドキュメントとしてまとめて[collections](コレクション) 格納します。MongoDB または Amazon DocumentDB エンドポイントの 1 つのデータベースは、データベース名で識別される特定のコレクションセットです。

MongoDB または Amazon DocumentDB ソースから移行する場合、並列ロード設定の操作は少し異なります。この場合、テーブルおよびビューではなく、選択したコレクションに対する並列ロード設定の自動セグメンテーションまたは範囲セグメンテーション タイプを考慮します。

トピック

- テーブル設定内のワイルドカードの制限
- 選択したテーブルおよびビューさらにコレクションで並列ロードを使用する
- 選択したテーブルまたはビューの LOB 設定を指定
- テーブル設定の例

テーブル設定ルールタイプを使用するテーブルマッピングルールの場合、以下のパラメータを適用することができます。

パラメータ	使用できる値:	説明
`rule-type`	`table-settings`	選択ルールにより指定されたテーブルまたはビューまたはコレクションにルールを適用する値。
`rule-id`	数値。	ルールを識別する一意の数値。
`rule-name`	英数字値。	ルールを特定する一意な名前。
`object-locator`	以下のパラメータを使用するオブジェクト。 • `schema-name` – スキーマの名前。MongoDB と Amazon DocumentDB エンドポイントの場合、これはコレクションの集合を保持するデータベースの名前です。 • `table-name` – テーブルまたはビューまたはコレクションの名前。	特定のスキーマ、テーブル、ビューの名前、または特定のデータベースとコレクションの名前 (ワイルドカードなし)。
`parallel-load`	以下のパラメータを使用するオブジェクト。	`object-locator` オプションで識別されたテーブルまたはビューでの並列

パラメータ	使用できる値:	説明
	• type – 並列ロードを有効にするかどうかを指定します。 そうでない場合、このパラメータは、テーブルまたはビューのパーティション、サブパーティションや並列でロードする他のセグメントを識別するメカニズムもしています。パーティションは、ソーステーブルまたはビューですでに定義され、名前で識別されるセグメントです。 MongoDB エンドポイントと Amazon DocumentDB エンドポイントの場合、パーティションは segments です。 は、関連する自動セグメンテーションパラメータを指定して、これらを自動的に計算 AWS DMS できます。または、範囲セグメンテーションパラメータを使用してこれらを手動で指定することもできます。 Oracle エンドポイントのみでは、サブパーティションはソーステーブルまたはビューですでに定義されて名前によって識別できるセグメントの追加レベルとなります。1 つ以上のテーブル列またはビュー列に値範囲の境界を指定することで、table-settings ルール内の他のセグメントを識別できます。 • partitions – type が partitions-list の場合、こ	ロード (マルチスレッド) オペレーションを指定する値。この場合、次の任意の方法で並行してロードすることができます。 • 使用可能なすべてのパーティションまたはサブパーティションで指定されるセグメントによって。 • パーティションおよびサブパーティションで選択することによって。 • 指定した自動セグメンテーションまたは範囲ベースのセグメントセグメントに基づく。 並列ロードの詳細については、「選択したテーブルおよびビューさらにコレクションで並列ロードを使用する」をご参照ください。

パラメータ	使用できる値:	説明
	の値はすべてのパーティションが並列ロードされるように指定します。 • subpartitions – Oracle エンドポイントのみでは、type が partitions-list の場合、この値はすべてのサブパーティションが並列ロードされるように指定します。 • columns – type が ranges の場合、この値は並列でロードする範囲ベースのセグメントを識別するために使用する列の名前を指定します。 • boundaries – type が ranges の場合、この値は並列でロードされる範囲ベースのセグメントを識別するために使用する columns の値を指定します。	

パラメータ	使用できる値:	説明
type	parallel-load の次のいずれか。 • partitions-auto - テーブルまたはビューのすべてのパーティションが並列でロードされます。各パーティションは個別のスレッドに割り当てられます。 これは、MongoDB および Amazon DocumentDB ソース エンドポイントが並列全ロードの自動セグメンテーション オプションを使用するために必要な設定です。 • subpartitions-auto – (Oracle エンドポイントのみ) テーブルまたはビューのすべてのサブパーティションが並列でロードされます。すべてのサブパーティションは個別のスレッドに割り当てられます。 • partitions-list - テーブルまたはビューで指定されたすべてのパーティションが並列でロードされます。Oracle エンドポイントのみでは、テーブルまたはビューのすべてのサブパーティションが並列でロードされます。指定する各パーティションおよびサブパーティションは、独自のスレッドに割り当てられます。並列ロードするパーティションおよびサブパーティションをパーティション名 (partitions) とサブパーティション名 (subpartitions) で指定します。	並列でロードするテーブルまたはビューまたはコレクションのパーティションまたはサブパーティション、セグメントを識別するメカニズム。

パラメータ	使用できる値:	説明
	• `ranges` – テーブルまたはビューまたはコレクションの範囲指定されたすべてのセグメントが並列ロードされます。識別した各テーブルまたはビューまたはコレクションのセグメントは自己スレッドに割り当てられます。このセグメントを列名 (`columns`) および 列の値 (`boundaries`) で指定します。 PostgreSQL エンドポイントはこのタイプの並列ロードのみに対応します。ソースエンドポイントとして MongoDB と Amazon DocumentDB は、全ロードのこの範囲セグメンテーションタイプと並列自動セグメンテーションタイプの両方に対応しています (`partitions-auto`)。 • none – テーブルまたはビューまたはコレクションは、パーティションやサブパーティションに関係なく、シングルスレッドタスクでロードされます(デフォルト)。詳細については、「[Creating a task] (タスクの作成)」をご参照ください。	

パラメータ	使用できる値:	説明
number-of-partitions	(オプション) type が MongoDB または Amazon DocumentDB エンドポイントの指定されたコレクションについて partitions-auto の場合、このパラメータは移行に使用されるパーティション (セグメント) の総数を指定します。デフォルトは 16 です。	並列ロードされるパーティションの正確な数を指定します。
collection-count-from-metadata	(オプション) MongoDB または Amazon DocumentDB エンドポイントの指定されたコレクションtypepartitions-auto に対して が で、このパラメータが に設定されている場合true、 はパーティション数を決定するために推定コレクション数 AWS DMS を使用します。このパラメータが に設定されている場合false、 は実際のコレクション数 AWS DMS を使用します。デフォルトは true です。	並列ロードするパーティションの数を計算するために、推定収集回数と実際の収集数のどちらを使用するかを指定します。
max-records-skip-per-page	(オプション) type が MongoDB または Amazon DocumentDB エンドポイントの指定されたコレクションについて partitions-auto の場合、これは各パーティションの境界を決定する際に一度にスキップするレコードの数です。 AWS DMS は、ページ割りスキップアプローチを使用して、パーティションの最小境界を決定します。デフォルトは 10,000 です。	各パーティションの境界を決定するときに一度にスキップするレコード数を指定します。デフォルトより比較的高い値を設定すると、タイムアウトし、タスクが失敗する可能性がある。デフォルトから比較的低い値を設定すると、ページあたりのオペレーションが多くなり、全ロードが遅くなります。

パラメータ	使用できる値:	説明
batch-size	(オプション) type が MongoDB または Amazon DocumentDB エンドポイントの指定されたコレクションについて partitions-auto の場合、この整数値により、1 回のラウンドトリップ バッチで返されるドキュメントの数が制限されます。バッチサイズがゼロ (0) の場合、カーソルはサーバー定義の最大バッチサイズを使用します。デフォルトは 0 です。	1 つのバッチで返されるドキュメントの最大数を指定します。各バッチには、サーバーへの往復が必要です。
partitions	type が partitions-list の場合、これは並列でロードするパーティションの名前を指定する文字列の配列です。	並列ロードするパーティションの名前。
subpartitions	(Oracle エンドポイントのみ) type が partitions-list の場合、これは並列してロードするサブパーティションの名前を指定する文字列の配列です。	並列ロードするサブパーティションの名前。
columns	type が ranges の場合、並列ロードする範囲ベースのテーブルセグメントまたはビューセグメントまたはコレクションセグメントを識別する列の名前に設定された文字列の配列です。	並列ロードする範囲ベースのテーブルセグメントまたはビューセグメントまたはコレクションセグメントを識別する列の名前。

パラメータ	使用できる値:	説明
boundaries	type が ranges の場合、列の値の配列の配列。各列の値の配列には、数量の列値と columns で指定された順序が含まれています。列の値の配列はテーブルセグメントまたはビューセグメントまたはコレクションセグメントの上限の境界を指定します。各追加列値の配列は、1 つの追加テーブルセグメントまたはビューセグメンまたはコレクションセグメントの上限の境界を追加します。このような範囲ベースのテーブルセグメントまたはビューセグメントまたはコレクションセグメントはすべて並列ロードされます。	並列ロードする範囲ベースのテーブルパーティションまたはビューパーティションまたはコレクションパーティションを識別する列の値。
lob-settings	以下のパラメータを使用するオブジェクト。 • mode - LOB の移行処理モードを指定します。 • bulk-max-size – mode の設定に応じて、LOB の最大サイズを指定します。	object-locator オプションによって識別されるテーブルまたはビューの LOB 処理を指定する値。指定された LOB 処理は、このテーブルまたはビューのみのすべてのタスク LOB 設定を上書きします。LOB 設定パラメータの使用に関する詳細については、「選択したテーブルまたはビューの LOB 設定を指定」をご参照ください。

パラメータ	使用できる値:	説明
mode	以下の値を使用して、指定されたテーブルまたはビューの LOB の移行処理を指定します。 • limited - (デフォルト)この値は、テーブルまたはビュー内の他のすべての列データ型とともにすべての LOB 移行インラインにより、制限された LOB モードに移行を設定します。主に小さな LOB (100 MB 以下) をレプリケートするときに、この値を使用します。また、bulk-max-size 値を指定します (ゼロは無効です) 。bulk-max-size より大きなサイズの移行された LOB はすべて、設定したサイズに切り捨てられます。 • unlimited - この値は、移行を完全 LOB モードに設定します。レプリケートするほとんどあるいはすべての LOB が 1 GB より大きい場合には、この値を使用します。bulk-max-size 値をゼロに指定する場合、すべての LOB は標準の完全 LOB モードで移行されます。この形式の unlimited モードでは、ソーステーブルまたはビューからのルックアップを使用して、すべての LOB が他の列データタイプとは別に移行されます。bulk-max-size 値をゼロより大きく設定する場合、すべての LOB は組み合わせ完全 LOB モードで移行されます。この形式の	LOB を移行するために使用されるメカニズム。

パラメータ	使用できる値:	説明
	`unlimited` モードでは、`bulk-max-size` よりも大きな LOB は、ソーステーブルルックアップまたはビュールックアップを使用して移行されます。これは、標準完全 LOB モードに類似しています。それ以外の場合は、このサイズ以下の LOB はインラインで移行されます。これは制限付き LOB モードに似ています。使用する形式に関係なく、`unlimited` モードでは LOB が切り捨てられることはありません。 • none - すべてのテーブルまたはビューの LOB はタスクの LOB 設定に従って移行されます。 タスクの LOB 設定の詳細については、「ターゲットメタデータのタスク設定」をご参照ください。 LOB を移行する方法およびこれらのタスクの LOB 設定を指定する方法については、「AWS DMS タスクでのソースデータベースの LOB サポートの設定」をご参照ください。	

パラメータ	使用できる値:	説明
bulk-max-size	この値の効果は mode によって異なります。	LOB の最大サイズ (キロバイト単位)。このオプションは、小さな LOB をレプリケートする必要がある場合、またはターゲットエンドポイントが無制限の LOB サイズをサポートしていない場合にのみ指定します。

テーブル設定内のワイルドカードの制限

次のとおり、"table-settings" ルールでのパーセントのワイルドカード ("%") の使用は、ソースデータベースではサポートされていません。

```
{
    "rule-type": "table-settings",
    "rule-id": "8",
    "rule-name": "8",
    "object-locator": {
        "schema-name": "ipipeline-prod",
        "table-name": "%"
    },
    "parallel-load": {
        "type": "partitions-auto",
        "number-of-partitions": 16,
        "collection-count-from-metadata": "true",
        "max-records-skip-per-page": 1000000,
        "batch-size": 50000
    }
  }
```

図のように"table-settings"ルール"%"で を使用する場合、 は例外を次のように AWS DMS 返します。

```
Error in mapping rules. Rule with ruleId = x failed validation. Exact
schema and table name required when using table settings rule.
```

さらに、 では、 で 1 つのタスクを使用して大量のコレクションをロードしない AWS ことをお勧めしますparallel-load。MaxFullLoadSubTasks タスク設定パラメータ値により、AWS DMS はリソース競合と、並行してロードするセグメント数を制限することに注意します。MaxFullLoadSubTasks の最大値は 49 です。

この代わりに各 "schema-name" と "table-name" を個別に指定して、最大のコレクションのソースデータベースのすべてのコレクションを指定します。また、移行は適切にスケールアップします。例えば、データベース内の大量のコレクションを処理するのに十分な数のレプリケーションインスタンスで複数のタスクを実行します。

選択したテーブルおよびビューさらにコレクションで並列ロードを使用する

移行を高速化してより効率的にするには、選択したリレーショナルテーブルおよびビューおよびコレクションを並列ロードできます。つまり、数本のスレッドを並列使用し、単一セグメント化テーブルまたはビュー、コレクションを移行できます。これを行うには、 は全ロードタスクをスレッドに AWS DMS 分割し、各テーブルセグメントを独自のスレッドに割り当てます。

この並列ロードプロセスを使用すると、まず複数スレッドが複数テーブルおよびビューおよびコレクションをソースのエンドポイントから並列アップロードするようにできます。次に、複数のスレッドを移行し、同じテーブルおよびビューおよびコレクションをターゲットエンドポイントに並列ロードすることができます。一部のデータベースエンジンでは、既存のパーティションあるいはサブパーティションを使用してテーブルおよびビューをセグメント化することができます。他のデータベースエンジンでは、特定のパラメータ (自動セグメンテーション) に従ってコレクション AWS DMS を自動的にセグメント化できます。それ以外の場合は、列の値の範囲を指定することで任意のテーブルまたはビュー、コレクションをセグメント化できます。

並列ロードは以下のソースエンドポイントでサポートされています。

- Oracle
- Microsoft SQL Server
- MySQL
- PostgreSQL
- IBM Db2 LUW
- SAP Adaptive Server Enterprise (ASE)
- MongoDB (並列全ロードの自動セグメンテーションと範囲セグメンテーション オプションのみサポート)
- Amazon DocumentDB (並列全ロードの自動セグメンテーションと範囲セグメンテーション オプションのみをサポート)

MongoDB および Amazon DocumentDB エンドポイントの場合、 は並列全ロードの範囲セグメンテーションオプションのパーティションキーである列に対して次のデータ型 AWS DMS をサポートします。

- ダブル
- 文字列
- ObjectId
- 32 ビット整数
- 64 ビット整数

テーブル設定ルールで使用する並列ロードは、次のターゲットエンドポイントでサポートされます。

- Oracle
- Microsoft SQL Server
- MySQL
- PostgreSQL
- Amazon S3
- SAP Adaptive Server Enterprise (ASE)
- Amazon Redshift
- MongoDB (並列全ロードの自動セグメンテーションと範囲セグメンテーション オプションのみサポート)
- Amazon DocumentDB (並列全ロードの自動セグメンテーションと範囲セグメンテーション オプションのみをサポート)
- Db2 LUW

並行してロードするテーブルとビューの最大数を指定するには、`MaxFullLoadSubTasks` タスク設定を使用します。

並列ロードタスクのサポートされるターゲットのテーブルまたはビューごとの最大スレッド数を指定するには、列値の境界を使用してさらに多くのセグメントを定義できます。

> **Important**
>
> `MaxFullLoadSubTasks` は、並行してロードするテーブルまたはテーブルのセグメント数を制御します。`ParallelLoadThreads` は、ロードを並行して実行するために

移行タスクが使用するスレッド数を制御します。上記の設定は乗算です。そのため、フルロードタスクで使用するスレッドの合計数は、ほぼ ParallelLoadThreads 値と MaxFullLoadSubTasks 値を乗算した値になります (ParallelLoadThreads * MaxFullLoadSubtasks))
多数のフルロードサブタスクがあり、多数の並列ロードスレッドがあるタスクを作成すると、タスクがメモリを大量に消費してエラーとなる可能性があります。

Amazon DynamoDB、Amazon Kinesis Data Streams、Apache Kafka、または Amazon Elasticsearch Service のターゲットのテーブルあたりの最大スレッド数を指定するには、ParallelLoadThreads ターゲットメタデータタスク設定を使用します。

ParallelLoadThreads を使用する際に並列ロードタスクのバッファサイズを指定するには、ParallelLoadBufferSize ターゲットメタデータタスク設定を使用します。

ParallelLoadThreads と ParallelLoadBufferSize の可用性と設定は、ターゲットエンドポイントによって異なります。

ParallelLoadThreads と ParallelLoadBufferSize の設定の詳細については、「ターゲットメタデータのタスク設定」をご参照ください。MaxFullLoadSubTasks の設定の詳細については、「全ロードタスク設定」をご参照ください。ターゲットエンドポイントに固有の情報については、関連するトピックをご参照ください。

並列ロードを使用するには、parallel-load オプションを指定した table-settings タイプのテーブルマッピングルールを作成します。table-settings ルールでは、並列ロードする単一テーブルまたはビュー、コレクションのセグメント化条件を指定できます。これを行うには、parallel-load オプションの type パラメータの複数のオプションのいずれかに設定します。

これを行う方法は、並行ロード用にテーブルまたはビュー、コレクションのセグメント化方法によって異なります。

- パーティション (またはセグメント) を使用 - partitions-auto タイプを使用して、すべての既存のテーブルまたはビューのパーティション (またはセグメント) をロードします。または、partitions-list タイプを指定したパーティション配列で使用して、選択したパーティションのみをロードします。

  MongoDB および Amazon DocumentDB エンドポイントのみ、 partitions-autoタイプと追加のオプションtable-settingsパラメータを使用して AWS DMS 自動的に計算するセグメントごとに、すべてのコレクションまたは指定されたコレクションをロードします。

- (Oracle エンドポイントのみ) サブパーティションを使用 - subpartitions-auto タイプを使用してすべての既存のテーブル サブパーティションまたはビューサブパーティションをロードします。または、partitions-list タイプを指定した subpartitions 配列で使用して、選択したサブパーティションのみをロードします。
- 定義するセグメントを使用 - 列値の境界を使用して定義したテーブルセグメントまたはビューセグメントまたはコレクションセグメントをロードします。これを行うには、ranges タイプを指定した columns 配列および boundaries 配列で使用します。

> **Note**
>
> PostgreSQL エンドポイントはこのタイプの並列ロードのみに対応します。ソースエンドポイントとして MongoDB と Amazon DocumentDB は、全ロードのこの範囲セグメンテーションタイプと並列自動セグメンテーションタイプの両方に対応しています (partitions-auto)。

並列でロードする追加のテーブルまたはビュー、コレクションを識別するには、追加の table-settings オブジェクトを parallel-load オプションで指定します。

次の手順では、各並列ロードタイプの JSON を記述する方法を、単純なものからきわめて複雑なものまで説明します。

すべてのテーブルまたはビューあるいはコレクション パーティション、またはすべてのテーブルまたはビューまたはコレクションのサブパーティションを指定するには

- partitions-auto タイプまたは subpartitions-auto タイプのいずれか (両方は使用できません) を指定して parallel-load を指定します。

  各テーブルまたはビューまたはコレクションのパーティション (セグメント) またはサブパーティションは、自己スレッドに自動割り当てされます。

  エンドポイントによっては、並列ロードには、テーブルまたはビューですでに定義されている場合のみ、パーティションあるいはサブパーティションが含まれます。MongoDB および Amazon DocumentDB ソースエンドポイントの場合、 はオプションの追加のパラメータに基づいてパーティション (またはセグメント) AWS DMS を自動的に計算できます。例えば、number-of-partitions、collection-count-from-metadata、max-records-skip-per-page、batch-size などです。

選択したテーブルまたはビューのパーティション、サブパーティション、またはその両方を指定するには

1. `partitions-list` タイプで `parallel-load` を指定します。
2. (オプション) パーティション名の配列を `partitions` の値として指定したパーティションが含まれます。

   指定されたパーティションがそれぞれ独自のスレッドに割り当てられます。

   > **Important**
   >
   > Oracle エンドポイントの場合、並列ロード用にパーティションとサブパーティションを選択するときに、パーティションとサブパーティションが重複しないようにします。重複するパーティションとサブパーティションを使用してデータを並列ロードすると、エントリが複製されるか、プライマリ キーの重複違反により失敗します。

3. (オプション)、Oracle エンドポイントのみ、サブパーティションを `subpartitions` の値として指定したサブパーティション名の配列を指定してインクルードします。

   指定されたサブパーティションがそれぞれ独自のスレッドに割り当てられます。

   > **Note**
   >
   > 並列ロードには、テーブルまたはビューですでに定義されている場合に、パーティションあるいはサブパーティションが含まれます。

テーブルセグメントまたはビューセグメントを列値の範囲として指定することができます。その場合、次の列の特性について注意してください。

- インデックス化された列を指定するとパフォーマンスが大幅に向上します。
- 最大 10 個の列を指定できます。
- 列を使用して、DOUBLE、FLOAT、BLOB、CLOB、NCLOB の各 AWS DMS データ型でセグメント境界を定義することはできません。
- Null 値のレコードはレプリケートされません。

テーブルセグメントまたはビューセグメント、コレクションセグメントを列値の範囲として指定するには

1. ranges タイプで parallel-load を指定します。
2. columns の値として列名の配列を指定して、テーブルまたはビューのセグメント間の境界を定義します。テーブルまたはビューのセグメント間で境界を定義するすべての列でこれを行います。

   列の順序は重要です。次に説明するように、各境界を定義するうえで最初の列がもっとも重要であり最後の列は最も重要度が低くなります。
3. boundaries の値として境界配列を指定して、すべてのテーブルまたはビューのセグメントでデータ範囲を定義します。境界配列は列値の配列の配列です。これを行うには、次のステップを実行します。

   a. 列と値の配列の各要素を各列に対応する値として指定します。列と値の配列は、定義する各テーブルまたはビューのセグメントの上限の境界を表します。columns 配列の列で指定したものと同じ順序で各列を指定します。

      DATE 列の値を、ソースでサポートされる形式で入力します。

   b. 各列値配列を、テーブルまたはビューの下部からセグメントまでの next-to-top各セグメントの上限として順番に指定します。指定した上限の上にテーブルの行が存在する場合、これらの行はテーブルまたはビューの一番上のセグメントを形成します。そのため、範囲ベースのセグメントの数は、境界配列のセグメント境界の数より 1 つ多くなることがあります。このような範囲ベースのセグメントはそれぞれ個別のスレッドに割り当てられます。

      データテーブル内のすべての列のデータ範囲を定義しない場合でも、null 以外のすべてのテーブルまたはビューのデータがレプリケートされます。

例えば、次のように COL1、COL2、 COL 3 に 3 つの列· 値の配列を定義したとします。

COL1	COL2	COL3
10	30	105
20	20	120
100	12	99

3 つのセグメントの境界を定義したことで、合計で 4 つのセグメントができることがあります。

各セグメントでレプリケートする行の範囲を識別するため、レプリケーションインスタンスは 4 つのセグメントごとにこれらの 3 つの列で検索を適用します。検索は次のようになります。

セグメント 1

以下の条件が true であるすべての行をレプリケートします。最初の 2 列の値が、対応するセグメント 1 の上限値以下です。また、3 番目の列の値は、セグメント 1 の上限値より小さくなります。

セグメント 2

以下の条件が true であるすべての行 (セグメント 1 行を除く) をレプリケートします。最初の 2 列の値が、対応する セグメント 2 の上限値以下です。また、3 番目の列の値は、セグメント 2 の上限値より小さくなります。

セグメント 3

以下の条件が true であるすべての行 (セグメント 2 行を除く) をレプリケートします。最初の 2 列の値が、対応する セグメント 3 の上限値以下です。また、3 番目の列の値は、セグメント 3 の上限値より小さくなります。

セグメント 4

残りのすべての行 (セグメント 1、2、3 の行を除く) をレプリケートします。

この場合、レプリケーションインスタンスが次のように WHERE 句を作成して各セグメントをロードします。

セグメント 1

```
((COL1 < 10) OR ((COL1 = 10) AND (COL2 < 30)) OR ((COL1 = 10) AND
(COL2 = 30) AND (COL3 < 105)))
```

セグメント 2

```
NOT ((COL1 < 10) OR ((COL1 = 10) AND (COL2 < 30)) OR ((COL1 = 10) AND
(COL2 = 30) AND (COL3 < 105))) AND ((COL1 < 20) OR ((COL1 = 20) AND
(COL2 < 20)) OR ((COL1 = 20) AND (COL2 = 20) AND (COL3 < 120)))
```

セグメント 3

```
NOT ((COL1 < 20) OR ((COL1 = 20) AND (COL2 < 20)) OR ((COL1 = 20) AND
(COL2 = 20) AND (COL3 < 120))) AND ((COL1 < 100) OR ((COL1 = 100) AND
(COL2 < 12)) OR ((COL1 = 100) AND (COL2 = 12) AND (COL3 < 99)))
```

セグメント 4

```
NOT ((COL1 < 100) OR ((COL1 = 100) AND (COL2 < 12)) OR ((COL1 = 100)
AND (COL2 = 12) AND (COL3 < 99)))
```

選択したテーブルまたはビューの LOB 設定を指定

1 つ以上の `table-settings` オブジェクトで `lob-settings` オプションを使用してタイプ `table-settings` のテーブルマッピングルールを作成すると、1 つ以上のテーブルに LOB 設定のタスクを設定できます。

選択したテーブルへの LOB 設定の指定は、次のソースエンドポイントでサポートされています。

- Oracle
- Microsoft SQL Server
- MySQL
- PostgreSQL
- IBM Db2 (以下で説明するように、`mode` および `bulk-max-size` 設定に応じます)
- SAP Adaptive Server Enterprise (ASE) (以下で説明するように、`mode` および `bulk-max-size` 設定に応じます)

選択したテーブルまたはビューへの LOB 設定の指定は、次のターゲットエンドポイントでサポートされています。

- Oracle
- Microsoft SQL Server
- MySQL
- PostgreSQL
- SAP ASE (以下で説明するように、`mode` および `bulk-max-size` 設定に応じます)

> **Note**
>
> LOB データタイプはプライマリキーが含まれるテーブルおよびビューのみで使用できます。

選択したテーブルまたはビューの LOB 設定を使用するには、lob-settings オプションを指定して table-settings タイプのテーブルマッピングルールを作成します。これにより、object-locator オプションで識別されたテーブルまたはビューの LOB 処理が指定されます。table-settings ルール内で、以下のパラメータを使用して lob-settings オブジェクトを指定できます。

- mode - 選択したテーブルまたはビューの LOB 移行処理のメカニズムを次のように指定します :
  - limited - デフォルトの制限付き LOB モードは最速で効率的なモードです。すべての LOB が小さいか、またはターゲットのエンドポイントで未制限の LOB サイズがサポートされていない場合のみ、このモードを使用します。また、limited を使用する場合、すべての LOB は bulk-max-size で設定したサイズ内にする必要があります。

    完全ロードタスクにおけるこのモードでは、レプリケーションインスタンスはすべての LOB を他の列のデータ型と一緒にメインテーブルストレージまたはメインビューストレージの一部としてインラインで移行します。ただし、bulk-max-size 値より大きい LOB はすべて、インスタンスによって指定のサイズに切り捨てられます。変更データキャプチャ (CDC) のロードの場合、インスタンスはすべての LOB をソーステーブルの参照を使用して移行します。これは標準の完全 LOB モードと同様です (以下を参照)。

    > **Note**
    >
    > 全ロードタスクに対してのみビューを移行できます。

  - unlimited - 完全 LOB モードの移行メカニズムは、次のように bulk-max-size に設定した値によって異なります:
    - [Standard full LOB mode](標準の完全 LOB モード) - bulk-max-size をゼロに設定すると、レプリケーション インスタンスは標準の完全 LOB モードを使用してすべての LOB を移行します。このモードでは、サイズにかかわらずすべての LOB を移行でソーステーブルまたはソースビューを参照する必要があります。このアプローチでは通常、制限付き LOB モードよりもはるかに低速で移行されます。すべてまたはほとんどの LOB が大きい (1 GB 以上) の場合にのみ、このモードを使用します。

- [Combination full LOB mode](組み合わせ完全 LOB モード) - `bulk-max-size` をゼロ以外の値に設定すると、この完全 LOB モードは制限付き LOB モードと標準完全 LOB モードを組み合わせて使用します。これは完全ロードタスク向けであり、LOB サイズが `bulk-max-size` 値以内の場合、インスタンスは制限付き LOB モードのようにインラインで LOB を移行します。LOB のサイズがこの値より大きい場合は、インスタンスは標準の完全 LOB モードのようにソーステーブルまたはソースビューの参照を使用して LOB を移行します。変更データキャプチャ (CDC) のロードの場合、インスタンスはすべての LOB をソーステーブルの参照を使用して移行します。これは標準の完全 LOB モードと同様です (以下を参照)。これは、LOB に関係なく行われます。

  > **Note**
  >
  > 全ロードタスクに対してのみビューを移行できます。

  このモードにより、移行速度はより高速な制限付き LOB モードと低速な標準の完全 LOB モードの中間になります。このモードは、小さい LOB と大きい LOB が混在しており、ほとんどの LOB が小さい場合にのみ使用します。

  この組み合わせ完全 LOB モードは、つぎのエンドポイントのみで使用できます。

  - IBM Db2 (ソースとして)
  - SAP ASE (ソースあるいはターゲットとして)

  `unlimited` モードに指定するメカニズムに関わらず、インスタンスはすべての LOB を完全に移行し、切り捨ては行われません。
- none - レプリケーション インスタンスはタスク LOB 設定を使用して、選択されたテーブルまたはビューの LOB を移行します。このオプションを使用すると、選択されたテーブルまたはビューで LOB 設定を使用した場合と使用しない場合の移行結果を比較できます。

指定したテーブルまたはビューにレプリケーションに含まれる LOB があるときには、`limited` LOB モードを使用する場合のみ `BatchApplyEnabled` タスクを `true` に設定できます。

場合によっては、`BatchApplyEnabled` を `true` に、また、`BatchApplyPreserveTransaction` を `false` に設定できます。このようなケースでは、テーブルまたはビューに LOB があり、ソースおよびターゲットのエンドポイントが Oracle である場合に、インスタンスは `BatchApplyPreserveTransaction` を `true` に設定します。

- bulk-max-size - 前の項目で説明した mode に応じて、この値を 0 または 0 以外の値 (KB) に設定します。limited モードでは、このパラメータにゼロ以外の値を設定する必要があります。

  インスタンスは LOB をバイナリ形式に変換します。したがって、レプリケートする必要がある最大の LOB を指定するには、サイズを 3 倍します。たとえば、最大の LOB が 2 MB の場合、bulk-max-size を 6000 (6 MB) に設定します。

テーブル設定の例

テーブル設定を使用する説明の例を次に示します。

Example パーティションでセグメント化されたテーブルのロード

次の例では、すべてのパーティションに基づいて並列でロードすることで、ソース内の SALES テーブルを効率的にロードします。

```
{
   "rules": [{
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "HR",
                "table-name": "SALES"
            },
            "parallel-load": {
                "type": "partitions-auto"
            }
        }
     ]
}
```

Example サブパーティションでセグメント化されたテーブルのロード

次の例では、すべてのサブパーティションに基づいて並列でロードすることで、Oracle ソース内の SALES テーブルを効率的にロードします。

```
{
   "rules": [{
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "HR",
                "table-name": "SALES"
            },
            "parallel-load": {
                "type": "subpartitions-auto"
            }
        }
     ]
}
```

Example パーティションのリストでセグメント化されたテーブルのロード

次の例では、特定のパーティションのリストに基づいて並列でロードすることで、ソース内の SALES テーブルをロードします。ここで、指定されたパーティションは ABCD やEFGH などアルファベットの一部で始まる値に命名されます。

```
{
    "rules": [{
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
```

```
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "HR",
                "table-name": "SALES"
            },
            "parallel-load": {
                "type": "partitions-list",
                "partitions": [
                    "ABCD",
                    "EFGH",
                    "IJKL",
                    "MNOP",
                    "QRST",
                    "UVWXYZ"
                ]
            }
        }
    ]
}
```

Example 選択されたパーティションおよびサブパーティションのリストでセグメント化された Oracle テーブルのロード

次の例では、選択されたパーティションおよびサブパーティションのリストを使用して並列でロードすることで、Oracle ソースの SALES テーブルをロードします。ここで、指定されたパーティションは ABCD やEFGH などアルファベットの一部で始まる値に命名されます。指定されたサブパーティションは、数字で始まる値を取って命名されます。たとえば、01234 や 56789 などです。

```
{
    "rules": [{
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
```

```
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "HR",
                "table-name": "SALES"
            },
            "parallel-load": {
                "type": "partitions-list",
                "partitions": [
                    "ABCD",
                    "EFGH",
                    "IJKL",
                    "MNOP",
                    "QRST",
                    "UVWXYZ"
                ],
                "subpartitions": [
                    "01234",
                    "56789"
                ]
            }
        }
    ]
}
```

Example 列の値の範囲でセグメント化されたテーブルのロード

次の例では、SALES_NO および REGION 列の値の範囲で指定されたセグメントを使用して並列でロードすることで、ソース内の SALES テーブルをロードします。

```
{
    "rules": [{
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
```

```
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "HR",
                "table-name": "SALES"
            },
            "parallel-load": {
                "type": "ranges",
                "columns": [
                    "SALES_NO",
                    "REGION"
                ],
                "boundaries": [
                    [
                        "1000",
                        "NORTH"
                    ],
                    [
                        "3000",
                        "WEST"
                    ]
                ]
            }
        }
    ]
}
```

ここで、2 つの列は、SALES_NO および REGION という名前でセグメント範囲に指定されます。2 つの境界は 2 セットの列の値 (["1000","NORTH"] および ["3000","WEST"]) で指定されます。

したがって、これら 2 つの境界によって、並列でロードされる次の 3 つのテーブルセグメントが識別されます。

セグメント 1

SALES_NO が 1000 以下であり REGION が "NORTH" 未満である行。つまり、EAST リージョンで販売数が最大 1000 まで。

セグメント 2

セグメント 1 以外で SALES_NO が 3000 以下であり REGION が "WEST" 未満である行。つまり、NORTH および SOUTH リージョンで販売数が 1,000 を超え 3,000 まで。

セグメント 3

セグメント 1 および セグメント 2 以外の残りすべての行。つまり、「WEST」リージョンで販売数が 3000 超。

Example 1 つは範囲でセグメント化され、もう 1 つはパーティションでセグメント化された 2 つのテーブルのロード

次の例では、SALES テーブルを識別したセグメント境界で並列にロードします。また、前の例と同様に、ORDERS テーブルをすべてのパーティションで並列にロードします。

```
{
    "rules": [{
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "HR",
                "table-name": "SALES"
            },
            "parallel-load": {
                "type": "ranges",
                "columns": [
                    "SALES_NO",
                    "REGION"
                ],
                "boundaries": [
                    [
```

```
                        "1000",
                        "NORTH"
                    ],
                    [
                        "3000",
                        "WEST"
                    ]
                ]
            }
        },
        {
            "rule-type": "table-settings",
            "rule-id": "3",
            "rule-name": "3",
            "object-locator": {
                "schema-name": "HR",
                "table-name": "ORDERS"
            },
            "parallel-load": {
                "type": "partitions-auto"
            }
        }
    ]
}
```

Example 制限付き LOB モードを使用した LOB を含むテーブルのロード

次の例では、ソース内の LOB を含む ITEMS テーブルを、切り捨てられない最大サイズが 100 MB の制限付き LOB モード (デフォルト) を使用してロードします。このサイズよりも大きい LOB はすべて 100 MB に切り捨てられます。すべての LOB は、他のすべての列のデータ型と一緒にインラインでロードされます。

```
{
   "rules": [{
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
```

```
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "INV",
                "table-name": "ITEMS"
            },
            "lob-settings": {
                "bulk-max-size": "100000"
            }
        }
    ]
}
```

Example 標準の完全 LOB モードを使用したテーブルのロード

次の例では、標準の完全 LOB モードを使用して、すべての LOB を切り捨てずに、ソースの ITEMS テーブルをロードします。サイズを問わずすべての LOB は、ソーステーブルの各 LOB ごとに参照を使用して、他のデータ型とは別にロードされます。

```
{
   "rules": [{
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "INV",
                "table-name": "ITEMS"
            },
            "lob-settings": {
                "mode": "unlimited",
                "bulk-max-size": "0"
```

```
            }
        }
    ]
}
```

Example 組み合わせ完全 LOB モードを使用した LOB を含むテーブルのロード

次の例では、組み合わせ完全 LOB モードを使用して、すべての LOB を切り捨てずに、ソースの ITEMS テーブルをロードします。サイズが 100 MB 以内のすべての LOB は他のデータ型とともにインラインでロードされます。制限付き LOB モードと同様です。サイズが 100 MB を超えるすべての LOB は、他のデータ型とは別にロードされます。この個別のロードでは、標準の完全 LOB モードのように、ソーステーブルのそのような各 LOB のルックアップが使用されます。

```
{
   "rules": [{
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "INV",
                "table-name": "ITEMS"
            },
            "lob-settings": {
                "mode": "unlimited",
                "bulk-max-size": "100000"
            }
        }
    ]
}
```

Example タスク LOB 設定を使用した LOB を含むテーブルのロード

次の例では、タスク LOB 設定を使用して、すべての LOB を含め、ソースの ITEMS テーブルをロードします。100 MB の bulk-max-size 設定は無視され、limited または unlimited モードにクイックリセットするためにのみ残されます。

```
{
   "rules": [{
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "INV",
                "table-name": "ITEMS"
            },
            "lob-settings": {
                "mode": "none",
                "bulk-max-size": "100000"
            }
        }
     ]
}
```

ソースフィルタの使用

ソースフィルタを使用すると、ソースからターゲットに転送されるレコードの数とタイプを制限できます。例えば、本社を拠点とする従業員だけがターゲットデータベースに移行されるように指定できます。フィルタは、選択ルールの一部です。データの列にフィルタを適用します。

ソースフィルタは、以下の制約に従う必要があります。

- 選択ルールには、フィルタを設定しないことも、1 つ以上のフィルタを選択することもできます。

- すべてのフィルタには 1 つ以上のフィルタ条件を設定できます。
- 複数のフィルタを使用する場合、フィルタのリストはフィルタ間で AND 演算子を使用しているものとして結合されます。
- 1 つのフィルタ内で複数のフィルタ条件を使用する場合、フィルタ条件のリストはフィルタ条件間で OR 演算子を使用しているものとして結合されます。
- フィルタは `rule-action = 'include'` の場合のみ適用されます。
- フィルタには、列名とフィルタ条件のリストが必要です。フィルター条件には、演算子に応じて、1 つの値、2 つの値、または値なしで関連付けられるフィルター演算子が必要です。
- 列名、テーブル名、ビュー名およびスキーマ名では大文字と小文字が区別されます。Oracle と Db2 の場合は常に大文字を使用する必要があります。
- フィルターは正確な名前のテーブルのみをサポートします。フィルターはワイルドカードをサポートしていません。

ソースフィルタの使用には、次の制限が適用されます。

- フィルターは right-to-left 言語の列を計算しません。
- LOB 列にはフィルタを適用しないでください。
- フィルタは、作成後に更新されていない[immutable](イミュータブル)の列にのみ適用します。ソースフィルターが作成後に更新可能で変更可能な[mutable](ミュータブル) 列に適用された場合、不正な動作が発生する可能性があります。

  例えば、列内の特定の行を除外または含めるフィルターでは、後で行が変更された場合でも、指定された行が常に除外または含められます。列 A で行 1 ~ 10 を除外または含むと、後でそれらが行 11 ~ 20 に変更されたとします。この場合、データが同じでなくなった場合でも、これらのデータは除外または含まれ続けます。

  同様に、フィルタの範囲外の行が後で更新され (または更新後に削除され)、フィルタの定義に従って除外または含める必要があるとします。このような場合は、ターゲットでレプリケートされます。

ソースフィルターを使用する場合、次の追加の懸念が適用されます。

- フィルタリング定義とプライマリキーに含まれる列を使用してインデックスを作成することをお勧めします。

JSON 形式のソースフィルタルールの作成

選択ルールで JSON filters パラメータを使用してソースフィルタを作成できます。filters パラメータは、1 つ以上の JSON オブジェクトの配列を指定します。各オブジェクトには、ソースフィルタのタイプ、列の名前、フィルタ条件を指定するパラメータがあります。これらのフィルタ条件には、1 つ以上のフィルタ演算子およびフィルタ値が含まれています。

次の表は、filters オブジェクトでソースフィルタリングを指定するパラメータを示しています。

パラメータ	値
filter-type	source
column-name	フィルターを適用するソース列の名前を持つパラメータです。名前は、大文字と小文字が区別されます。
filter-conditions	filter-operator 値に応じて、1 つまたは複数のオブジェクトでなる配列には filter-operator パラメータおよびゼロか正の関連値パラメータが含まれています。
filter-operator	次のいずれかの値を持つパラメータ: • lte— 1 つの値以下 • ste— 1 つの値以下 (lte エイリアス) • gte— 1 つ以上の値 • eq— 1 つの値に等しい • noteq— 1 つの値に等しくない • between - 2 つの値に等しい、またはその間 • notbetween – 2 つの値に等しい、またはその間 • null – NULL 個の値 • notnull — NULL 個の値でない
value または start-value と end-value または 値なし	filter-operator と関連付けられた 0 個またはそれ以上の値パラメータ:

パラメータ	値
	• filter-operator が lte または ste、gte、eq、noteqを使用する場合、valueを使用して、1 つの値パラメータを指定します。 • filter-operator が between または notbetween の場合、start-value と end-value を使用して、2 つの値パラメータを指定します。 • filter-operator が null または notnull の場合、値パラメータを指定しません。

以下の例では、ソースフィルタを使用する一般的な方法をいくつか示します。

Example 1 つのフィルタ

次のフィルタは、empid >= 100 のすべての従業員をターゲットデータベースにレプリケートします。

```
{
    "rules": [{
        "rule-type": "selection",
        "rule-id": "1",
        "rule-name": "1",
        "object-locator": {
            "schema-name": "test",
            "table-name": "employee"
        },
        "rule-action": "include",
        "filters": [{
            "filter-type": "source",
            "column-name": "empid",
            "filter-conditions": [{
                "filter-operator": "gte",
                "value": "50"
            },{
                "filter-operator": "noteq",
                "value": "100"
            }]
        }]
    }]
```

```
 }
```

Example 複数のフィルタ演算子

以下のフィルタは、複数のフィルタ演算子を 1 つのデータ列に適用します。このフィルタは、(empid <= 10) または (empid is between 50 and 75) または (empid >= 100) のすべての従業員をターゲットデータベースにレプリケートします。

```
{
    "rules": [{
        "rule-type": "selection",
        "rule-id": "1",
        "rule-name": "1",
        "object-locator": {
            "schema-name": "test",
            "table-name": "employee"
        },
        "rule-action": "include",
        "filters": [{
            "filter-type": "source",
            "column-name": "empid",
            "filter-conditions": [{
                "filter-operator": "lte",
                "value": "10"
            }, {
                "filter-operator": "between",
                "start-value": "50",
                "end-value": "75"
            }, {
                "filter-operator": "gte",
                "value": "100"
            }]
        }]
    }]
}
```

Example 複数のフィルタ

以下のフィルターは、テーブル内の 2 つの列に複数のフィルターを適用します。このフィルタは、(empid <= 100) および (dept = tech) のすべての従業員をターゲットデータベースにレプリケートします。

```
{
    "rules": [{
        "rule-type": "selection",
        "rule-id": "1",
        "rule-name": "1",
        "object-locator": {
            "schema-name": "test",
            "table-name": "employee"
        },
        "rule-action": "include",
        "filters": [{
            "filter-type": "source",
            "column-name": "empid",
            "filter-conditions": [{
                "filter-operator": "lte",
                "value": "100"
            }]
        }, {
            "filter-type": "source",
            "column-name": "dept",
            "filter-conditions": [{
                "filter-operator": "eq",
                "value": "tech"
            }]
        }]
    }]
}
```

Example NULL 値のフィルタリング

以下のフィルターは、空の値でフィルタリングする方法を示しています。これは、dept = NULL のすべての従業員をターゲットデータベースにレプリケートします。

```
{
    "rules": [{
        "rule-type": "selection",
        "rule-id": "1",
        "rule-name": "1",
        "object-locator": {
            "schema-name": "test",
            "table-name": "employee"
        },
        "rule-action": "include",
        "filters": [{
            "filter-type": "source",
            "column-name": "dept",
            "filter-conditions": [{
                "filter-operator": "null"
            }]
        }]
    }]
}
```

Example NOT 演算子を使用したフィルタリング

一部の演算子は、負形式で使用できます。次のフィルタは、(empid is < 50) OR (empid is > 75) のすべての従業員をターゲットデータベースにレプリケートします。

```
{
    "rules": [{
        "rule-type": "selection",
        "rule-id": "1",
        "rule-name": "1",
        "object-locator": {
            "schema-name": "test",
            "table-name": "employee"
        },
        "rule-action": "include",
        "filters": [{
            "filter-type": "source",
            "column-name": "empid",
            "filter-conditions": [{
                "filter-operator": "notbetween",
                "start-value": "50",
```

```
                "end-value": "75"
            }]
        }]
    }]
}
```

Example 混合フィルター演算子の使用

AWS DMS バージョン 3.5.0 から開始し、包括的な演算子と負の演算子を混在させることができます。

次のフィルタは、(empid != 50) AND (dept is not NULL) のすべての従業員をターゲットデータベースにレプリケートします。

```
{
    "rules": [{
        "rule-type": "selection",
        "rule-id": "1",
        "rule-name": "1",
        "object-locator": {
            "schema-name": "test",
            "table-name": "employee"
        },
        "rule-action": "include",
        "filters": [{
            "filter-type": "source",
            "column-name": "empid",
            "filter-conditions": [{
                "filter-operator": "noteq",
                "value": "50"
            }]
        }, {
            "filter-type": "source",
            "column-name": "dept",
            "filter-conditions": [{
                "filter-operator": "notnull"
            }]
        }]
    }]
}
```

その他のフィルター演算子と同時に null を使用する場合は、次の点に注意します。

- 同じフィルター内で包括フィルター条件、負のフィルター条件、null フィルター条件を同時に使用すると、NULL 値を持つレコードはレプリケートされません。
- 同じフィルター内で包含フィルター条件を使用せずに、負のフィルター条件と null フィルター条件を同時に使用すると、データはレプリケートされません。
- null フィルター条件を明示的に設定せずに負のフィルター条件を使用すると、NULL 値を持つレコードはレプリケートされません。

時刻と日付でフィルタリング

インポートするデータを選択するときは、フィルター条件の一部として日付または時刻を指定できます。 はフィルタリングに日付形式 YYYY-MM-DD と時刻形式 YYYY-MM-DD HH:MM:SS AWS DMS を使用します。 AWS DMS 比較関数は SQLite の規則に従います。SQLite のデータ型と日付変換については、SQLite ドキュメントの「SQLite バージョン 3 のデータ型」をご参照ください。

以下の例は、日付をフィルタリングする方法を示しています。これは、empstartdate >= January 1, 2002 のすべての従業員をターゲットデータベースにレプリケートします。

Example 1 つの日付フィルタ

```
{
    "rules": [{
        "rule-type": "selection",
        "rule-id": "1",
        "rule-name": "1",
        "object-locator": {
            "schema-name": "test",
            "table-name": "employee"
        },
        "rule-action": "include",
        "filters": [{
            "filter-type": "source",
            "column-name": "empstartdate",
            "filter-conditions": [{
                "filter-operator": "gte",
                "value": "2002-01-01"
            }]
        }]
```

```
    }]
}
```

タスクの移行前評価の有効化と操作

移行前評価では、データベース移行タスクの特定のコンポーネントを評価して、移行タスクが期待どおりに実行されない可能性のある問題を特定するのに役立ちます。この評価では、新規または変更されたタスクを実行する前に、問題を特定して修正できます。これにより、要件の欠落や既知の制限によるタスク障害に関連する遅延を回避できます。

AWS DMS では、移行前評価の 2 つの異なるオプションにアクセスできます。

- データ型評価： 評価の範囲が制限されたレガシーレポート。
- 移行前評価の実行: データ型評価結果など、さまざまなタイプの個別評価が含まれます。

> **Note**
>
> 移行前評価の実行を選択した場合、データ型評価を個別に選択する必要はありません。

これらのオプションについては、以下のトピックで説明します。

- 移行前評価の実行の指定、スタート、および表示: 移行前 (推奨) 評価の実行では、新規または既存の移行タスク設定に基づいて実行する 1 つ以上の個別の評価を指定します。個々の評価では、移行タイプ、サポートされているオブジェクト、インデックス設定、移行するスキーマやテーブルを識別するテーブルマッピングなどの他のタスク設定などの基準の観点から、サポートされているソースデータベースやターゲットデータベースの特定の要素を評価します。

  例えば、個々の評価では、エンジンのバージョンに基づいて、移行できるソースデータ型やプライマリキー形式、移行できないソースデータ型を評価する AWS DMS 場合があります。 AWS DMS マネジメントコンソールを使用するか、 AWS CLI および SDKs を使用して AWS DMS API にアクセスすることで、最新の評価実行の結果を開始して表示し、タスクの以前のすべての評価実行の結果を表示できます。また、 がこれらの結果を保存 AWS DMS するために選択した Amazon S3 バケット内のタスクに対する以前の評価実行の結果を表示することもできます。

> **Note**
>
> 利用可能な個々の評価数と種類は、経時的に増加する傾向があります。定期更新の詳細については、「個別の評価を指定する」をご参照ください。

- データ型評価の開始と表示 (レガシー) : データ型 (レガシー) 評価は、単一の JSON 構造で単一のタイプの移行前評価の結果を返します。サポートされているリレーショナルソースデータベースインスタンスでは正しく移行されない可能性のあるデータ型です。このレポートは、移行対象として選択されたソースデータベース内のすべてのスキーマとテーブルにあるすべての問題のあるデータ型の結果を返します。

移行前評価の前提条件の作成

このセクションでは、移行前評価の作成に必要な Amazon S3 および IAM リソースについて説明します。

S3 バケットを作成する

AWS DMS は、移行前評価レポートを S3 バケットに保存します。S3 バケットを作成するには、次の手順を実行します。

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. バケットの作成 ページで、dms-bucket-*yoursignin* など、バケットのサインイン名を含むグローバルに一意の名前を入力します。
4. DMS 移行タスク AWS リージョン の を選択します。
5. 残りの設定はそのままにして、バケットの作成 を選択します。

IAM リソースを作成する

DMS は IAM ロールとポリシーを使用して S3 バケットにアクセスし、移行前評価の結果を保存します。

IAM ポリシーを作成するには、次の手順を実行します。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. ナビゲーションペインで、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. [ポリシーの作成] ページで、[JSON] タブをクリックします。
5. 次の JSON コードをエディタに貼り付けて、サンプルコードを置き換えます。*my-bucket* を、前のセクションで作成した Amazon S3 バケットの名前に置き換えます。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:DeleteObject",
            "s3:GetObject",
            "s3:PutObjectTagging"
         ],
         "Resource":[
            "arn:aws:s3:::my-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetBucketLocation"
         ],
         "Resource":[
            "arn:aws:s3:::my-bucket"
         ]
      }
   ]
}
```

6. 次へ: タグ を選択し、次へ: レビュー を選択します。
7. [名前*] には **DMSPremigrationAssessmentS3Policy** と入力して、[ポリシーを作成] をクリックします。

IAM ロールを作成するには、次の手順を実行します。

1. IAM コンソールのナビゲーションペインで [ロール] を選択します。
2. [ロールを作成] を選択します。
3. [信頼されたエンティティを選択] ページの [信頼されたエンティティタイプ] では、AWS [サービス] を選択します。他のサービスのユースケース AWS で、DMS を選択します。
4. DMS チェックボックスをオンにし、次へを選択します。
5. アクセス許可の追加ページで、DMSPremigrationAssessmentS3Policy を選択します。[次へ] をクリックします。
6. [名前、確認、および作成] ページで、[ロール名] に **DMSPremigrationAssessmentS3Role** と入力して、[ロールの作成] をクリックします。
7. [ロール] ページの [ロール名] には、**DMSPremigrationAssessmentS3Role** と入力します。DMSPremigrationAssessmentS3Role を選択します。
8. DMSPremigrationAssessmentS3Role ページで、信頼関係タブを選択します。[Edit trust policy] (信頼ポリシーを編集) を選択します。
9. [信頼ポリシーを編集] ページで、次の JSON コードをエディタに貼り付けて、既存のテキストを置き換えます。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"dms.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

   このポリシーは、移行前評価の実行結果を S3 バケットに入れるアクセス`sts:AssumeRole`許可を DMS に付与します。
10. [ポリシーの更新] を選択します。

移行前評価の実行の指定、スタート、および表示

移行前評価では、新規または既存の移行タスク設定に基づいて実行する 1 つ以上の個別の評価を指定します。個々の評価では、移行タイプ、サポートされるオブジェクト、インデックス構成、および移行するスキーマとテーブルを識別するためのテーブルマッピングなどのその他のタスク設定などの検討事項に応じて、ソースまたはターゲットデータベースの特定の要素を評価します。例えば、個々の評価では、移行できるソースデータ型またはプライマリキー形式と移行できないソースデータ型が評価される場合があります。

個別の評価を指定する

新しい評価実行を作成するときに、タスク設定に適用可能な個々の評価の一部またはすべてを実行するように選択できます。

AWS DMS は、次のリレーショナルソースおよびターゲットデータベースエンジンの移行前評価の実行をサポートします。

- Oracle の評価
- Sql サーバー評価
- MySQL 評価 (MariaDB および Amazon Aurora MySQL 互換エディションを含む)
- PostgreSQL 評価 (Amazon Aurora PostgreSQL 互換エディションを含む)

移行前の評価の実行のスタートと表示

AWS DMS マネジメントコンソール、、 AWS CLIおよび AWS DMS API を使用して、新規または既存の移行タスクの移行前評価の実行を開始できます。

新規または既存のタスクに対して移行前評価をスタートするには

1. AWS DMS マネジメントコンソールで [Database migration tasks] (データベース移行タスク) のページから、次のいずれかを実行します :

   - 新しいタスクを作成して評価するには、タスクの作成 を選択します。[Create database migration task page] (データベース移行タスクの作成ページ) が開きます:

     1. テーブルマッピングなど、タスクの作成に必要なタスク設定を入力します。
     2. 移行前評価セクションでは、移行前評価の実行チェックボックスがオンになっています。このページには、新しいタスクの評価実行を指定するオプションが含まれています。

> **Note**
>
> 新しいタスクを作成するときに、移行前評価の実行を有効にすると、タスクの作成時にタスクを自動的にスタートするオプションが無効になります。評価の実行が完了したら、タスクを手動でスタートできます。

- 既存のタスクを評価するには、データベース移行タスクページで既存のタスクの識別子を選択します。選択した既存のタスクのタスクページが開きます：

  1. [Actions] (アクション) を選択し、[Create premigration assessment] (移行前評価を作成) を選択します。[Create premigration assessment] (移行前評価を作成) ページが開き、既存のタスクに対する評価の実行を指定するオプションが表示されます。

2. 評価実行の一意の名前を入力するか、デフォルト値のままにします。
3. この評価の実行に含める使用可能な個別の評価を選択します。現在のタスク設定に基づいて、使用可能な個別評価のみ選択可能です。デフォルトでは、使用可能なすべての個別評価が有効で、選択されています。
4. 評価結果レポートを保存するために、アカウント内の Amazon S3 バケットとフォルダを検索して選択します。評価実行用のリソースの設定については、「」を参照してください移行前評価の前提条件の作成。
5. 選択した Amazon S3 バケットおよびフォルダへの完全なアカウントアクセス権を持つ IAM ロールを選択または入力します。評価実行用のリソースの設定については、「」を参照してください移行前評価の前提条件の作成。
6. 必要に応じて、Amazon S3 バケットの評価結果レポートの暗号化設定を選択します。S3 バケット暗号化の詳細については、Amazon S3バケットのデフォルトのサーバー側の暗号化動作の設定」を参照してください。
7. 新しいタスクのため[Create task] (タスクの作成) を選択するか、既存のタスクについて[Create] (作成) を選択します。

   [データベース移行タスク] ページが開き、新しいタスクまたは変更されたタスクが [作成中...] の [ステータス] でリストに表示され、タスクの作成後に移行前評価の実行が開始されることを示すバナーメッセージも表示されます。

AWS DMS は、 AWS DMS マネジメントコンソール、 AWS CLI、または AWS DMS API を使用して、最新および以前の移行前評価実行へのアクセスを提供します。

評価実行の結果を表示するには

1. AWS DMS マネジメントコンソールから、データベース移行タスクページで既存のタスクの識別子を選択します。既存のタスクのタスクページが開きます。
2. 既存のタスクページで [Premigration assessments] (移行前評価) タブを選択します。これにより、そのページの移行前評価セクションが開き、評価実行の結果が名前順に時系列順に表示されます。最新の結果がリストの上部に表示されます。結果を表示する評価実行の名前を選択します。

これらの評価結果は、最新の評価の実行名とそのステータスの概要から始まり、指定された個々の評価とそのステータスのリストが続きます。次に、リスト内の名前を選択して、個々の評価のステータスの詳細を調べることができます。結果は、テーブルの列レベルまで表示されます。

アセスメント実行のステータス概要と個々の評価の両方に、[Status] (ステータス) 値が表示されます。この値は、評価実行の全体的なステータスと、個別評価の同様のステータスを示します。以下のリストに示しているのは、評価実行の[Status] (ステータス) 値:

- "cancelling" — 評価の実行はキャンセルされました。
- "deleting" — 評価の実行は削除されました。
- "failed" — 少なくとも 1 つの個別評価が failed ステータスです。
- "error-provisioning" — リソースのプロビジョニング中に内部エラーが発生しました (provisioning ステータス)。
- "error-executing" — 個々の評価の実行中に内部エラーが発生しました (running ステータス)。
- "invalid state" — 評価の実行の状態が不明です。
- "passed" — 個々の評価はすべて完了しており、failed ステータスのものはありません。
- "provisioning" — 個々の評価の実行に必要なリソースがプロビジョニングされています。
- "running" — 個別評価が実行中です。
- "starting" — 評価の実行のスタート中ですが、個別評価に対してリソースがまだプロビジョニングされていません。
- "warning" — 少なくとも 1 つの個別評価が warning ステータスです。

以下のリストに示しているのは、評価実行の個々の評価についての [Status] (ステータス) 値です：

- "cancelled" — 個々の評価は、評価の実行をキャンセルする一環としてキャンセルされました。
- "error" — 個々の評価は正常に完了しませんでした。
- "failed" — 個々の評価が正常に完了し、検証結果が失敗しました。詳細については、結果の詳細をご参照ください。
- "invalid state" — 個々の評価の状態が不明です。
- "passed" — 個々の評価は検証結果が成功で完了しました。
- "pending" — 個々の評価の実行を待っています。
- "running" — 個々の評価中です。
- "warning" — 個々の評価は警告検証結果ありで正常に完了しました。詳細については、結果の詳細をご参照ください。

Amazon S3 で評価実行結果の JSON ファイルを表示することもできます。

Amazon S3 で実行される評価の JSON ファイルを表示するには

1. AWS DMS マネジメントコンソールから、評価実行のステータス概要に表示される Amazon S3 バケットリンクを選択します。これにより、バケットフォルダと、バケットに保存されている他の Amazon S3 オブジェクトのリストが表示されます。結果がバケットフォルダに保存されている場合は、フォルダを開きます。
2. 評価の実行結果は、いくつかの JSON ファイルで確認できます。summary.json ファイルには、評価実行の全体的な結果が含まれています。残りのファイルは、評価の実行に指定された unsupported-data-types-in-source.json といったような個別評価名が付けられます。。これらのファイルには、選択した評価実行の対応する個別評価の結果が含まれます。

既存の移行タスクの移行前評価の実行を開始して結果を表示するには、次の CLI コマンドと AWS DMS API オペレーションを実行します。

- CLI:describe-applicable-individual-assessments、API:DescribeApplicableIndividualAssessments - 1 つ以上のタスク構成パラメータを所与として、新しい移行前評価実行で指定できる個別評価のリストを提供します。
- CLI:start-replication-task-assessment-run、API:StartReplicationTaskAssessmentRun - 既存の移行タスクの 1 つ以上の個別評価に対して、新しい移行前評価の実行を開始します。

- CLI:describe-replication-task-assessment-runs、API:DescribeReplicationTaskAssessmentRuns - フィルター設定に基づいて、移行前評価実行のページ割りされたリストを返します。
- CLI:describe-replication-task-individual-assessments、API:DescribeReplicationTaskIndividualAssessments — フィルター設定に基づいて、個々の評価のページ割りされたリストを返します。
- CLI:cancel-replication-task-assessment-run、API:CancelReplicationTaskAssessmentRun - 1 つの移行前評価実行をキャンセルしますが、削除はしません。
- CLI:delete-replication-task-assessment-run、API:DeleteReplicationTaskAssessmentRun - 1 つの移行前評価実行の記録を削除します。

個々の評価

このセクションでは、個々の移行前評価について説明します。

AWS DMS API を使用して個々の移行前評価を作成するには、StartReplicationTaskAssessmentRunアクションの `IncludeOnly`パラメータにリストされている API キーを使用します。

トピック

- すべてのエンドポイントタイプの評価
- Oracle の評価
- Sql サーバー評価
- MySQL 評価
- MariaDB 評価
- PostgreSQL 評価

すべてのエンドポイントタイプの評価

このセクションでは、すべてのエンドポイントタイプの個々の移行前評価について説明します。

トピック

- サポートされないデータ型

- ラージオブジェクト (LOBsが使用されますが、ターゲット LOB 列は null にできません
- ラージオブジェクト (LOBsがあるが、プライマリキーや一意の制約がないソーステーブル
- CDC または全ロードおよび CDC タスクのみのプライマリキーのないソーステーブル
- CDC タスクのみのプライマリキーのないターゲットテーブル
- サポートされていないソースプライマリキータイプ - 複合プライマリキー

サポートされないデータ型

API キー: `unsupported-data-types-in-source`

DMS がサポートしていないソースエンドポイントのデータ型をチェックします。エンジン間ですべてのデータ型を移行できるわけではありません。

ラージオブジェクト (LOBsが使用されますが、ターゲット LOB 列は null にできません

API キー: `full-lob-not-nullable-at-target`

レプリケーションがフル LOB モードまたはインライン LOB モードを使用する場合に、ターゲットの LOB 列の null 可能性をチェックします。これらの LOB モードを使用する場合、DMS では LOB 列が null である必要があります。この評価では、ソースデータベースとターゲットデータベースがリレーショナルである必要があります。

ラージオブジェクト (LOBsがあるが、プライマリキーや一意の制約がないソーステーブル

API キー: `table-with-lob-but-without-primary-key-or-unique-constraint`

LOB を持つが、プライマリ キーまたは一意キーのないソーステーブルが存在するかどうかをチェックします。テーブルには、DMS が LOBs を移行するためのプライマリキーまたは一意のキーが必要です。この評価では、ソースデータベースがリレーショナルである必要があります。

CDC または全ロードおよび CDC タスクのみのプライマリキーのないソーステーブル

API キー: `table-with-no-primary-key-or-unique-constraint`

フルロードおよび変更データキャプチャ (CDC) 移行、または CDC のみの移行について、ソーステーブルにプライマリキーまたは一意のキーが存在するかどうかを確認します。プライマリキーまたは一意のキーがないと、CDC 移行中にパフォーマンスの問題が発生する可能性があります。この評価では、ソースデータベースをリレーショナルにし、移行タイプに CDC を含める必要があります。

CDC タスクのみのプライマリキーのないターゲットテーブル

API キー: `target-table-has-unique-key-or-primary-key-for-cdc`

CDC のみの移行で作成済みターゲットテーブルに、プライマリ キーまたは一意キーが存在するかどうかをチェックします。DMS が更新や削除を適用すると、プライマリキーや一意のキーがないと、ターゲットでテーブル全体のスキャンが発生する可能性があります。これにより、CDC 移行中にパフォーマンスの問題が発生する可能性があります。この評価では、ターゲットデータベースをリレーショナルにし、移行タイプに CDC を含める必要があります。

サポートされていないソースプライマリキータイプ - 複合プライマリキー

API キー: `unsupported-source-pk-type-for-elasticsearch-target`

Amazon OpenSearch Service に移行するときに、ソーステーブルに複合プライマリキーが存在するかどうかを確認します。ソーステーブルのプライマリキーは、1 つの列のみで構成する必要があります。この評価では、ソースデータベースがリレーショナルで、ターゲットデータベースが DynamoDB である必要があります。

> **Note**
>
> DMS は、ソースプライマリキーが複数の列で構成される OpenSearch サービスターゲットへのソースデータベースの移行をサポートしています。

Oracle の評価

このセクションでは、Oracle のソースエンドポイントを使用する移行タスクの個別の移行前評価について説明します。

> **Note**
>
> このセクションの移行前評価を使用するには、次のアクセス権限を `dms_user` 追加する必要があります。
>
> ```
> grant select on gv_$parameter to dms_user;
> grant select on v_$instance to dms_user;
> grant select on v_$version to dms_user;
> grant select on gv_$ASM_DISKGROUP to dms_user;
> grant select on gv_$database to dms_user;
> grant select on DBA_DB_LINKS to to dms_user;
> ```

```
grant select on gv_$log_History to dms_user;
grant select on gv_$log to dms_user;
grant select on dba_types to dms_user;
grant select on dba_users to dms_user;
grant select on dba_directories to dms_user;
```

Oracle をソースとして使用する場合の権限の詳細については、「のセルフマネージド Oracle ソースに必要なユーザーアカウント権限 AWS DMS」を参照してください。

トピック

- データベースレベルでサプリメンタルロギングを確認する
- スタンバイのために必要な DB リンクが作成されているかを検証する
- LOB データ型と Binary Reader が設定されているかに関する Oracle の検証
- データベースが CDB かを検証します。
- Oracle Database Edition を確認します。
- DMS の Oracle CDC メソッドを検証する
- DMS の Oracle RAC 設定を検証する
- DMS ユーザーがターゲットに対するアクセス許可を持っているかどうかを検証する
- すべての列に補足ログ記録が必要かどうかを検証する
- プライマリキーまたは一意のキーを持つテーブルでサプリメンタルログ記録が有効になっているかどうかを検証する
- SecureFile LOBs があり、タスクがフル LOB モードに設定されているかどうかを検証する
- タスクスコープに含まれるテーブル内で関数ベースのインデックスが使用されているかどうかを確認します。
- グローバル一時テーブルがタスクスコープに含まれるテーブルで使用されているかどうかを確認します。
- オーバーフローセグメントを持つインデックスで編成されたテーブルが、タスクスコープに含まれるテーブルで使用されているかどうかを確認します。
- タスクスコープに含まれるテーブルでマルチレベルネストテーブルが使用されているかどうかを確認します。
- タスクスコープに含まれるテーブルで、非表示の列が使用されているかどうかを確認します。
- ROWID 列に基づくマテリアライズドビューがタスクスコープに含まれるテーブルで使用されているかどうかを確認します。

- Active Data Guard DML リダイレクト機能を使用しているかどうかを確認します。
- ハイブリッドパーティションテーブルが使用されているかどうかを確認します。
- スキーマのみの Oracle アカウントが使用されているかどうかを検証する
- 仮想列が使用されているかどうかを検証する
- タスクスコープで定義されたテーブル名にアポストロフィが含まれているかどうかを確認します。
- タスクスコープで定義された列に XMLType、Long、または Long Raw データ型があるかどうかを検証し、タスク設定で LOB モード設定を確認します。
- ソース Oracle バージョンが でサポートされているかどうかを確認します AWS DMS。
- ターゲットの Oracle バージョンが でサポートされているかどうかを確認します AWS DMS。
- ターゲットの Oracle バージョンが でサポートされているかどうかを確認します AWS DMS。
- DMS ユーザーがデータ検証を使用するために必要なアクセス許可を持っているかどうかを確認します。
- DMS ユーザーが Oracle ASM で Binary Reader を使用するアクセス許可を持っているかどうかを検証する
- DMS ユーザーが Oracle 非 ASM で Binary Reader を使用するアクセス許可を持っているかどうかを検証する
- DMS ユーザーが CopyToTempFolder メソッドで Binary Reader を使用するアクセス許可を持っているかどうかを検証する
- DMS ユーザーが Oracle スタンバイをソースとして使用するアクセス許可を持っているかどうかを検証する
- DMS ソースがアプリケーションコンテナ PDB に接続されているかどうかを検証する
- テーブルに XML データ型がタスクスコープに含まれているかどうかを確認します。
- ソースデータベースでアーカイブログモードが有効になっているかどうかを確認します。
- RDS Oracle のアーカイブログの保持を検証します。
- テーブルに、タスクスコープに含まれる拡張データ型があるかどうかを検証します。
- タスクスコープに含まれるオブジェクト名の長さを検証します。
- DMS ソースが Oracle PDB に接続されているかどうかを検証する
- テーブルにタスクスコープに含まれる空間列があるかどうかを検証します。
- DMS ソースが Oracle スタンバイに接続されているかどうかを確認します。
- ソースデータベースのテーブルスペースが TDE を使用して暗号化されているかどうかを確認します。
- ソースデータベースが Oracle ASM かどうかを検証する

データベースレベルでサプリメンタルロギングを確認する

API キー: `oracle-supplemental-db-level`

この移行前評価では、サプリメンタルロギングがデータベースレベルで有効になっているかを検証します。Oracle データベースを移行のソースとして使用するには、サプリメンタルロギングを有効にする必要があります。

サプリメンタルロギングを有効にするには、次のクエリを使用します。

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA
```

詳細については、「サプリメンタル ロギングの設定」を参照してください。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

スタンバイのために必要な DB リンクが作成されているかを検証する

API キー: `oracle-validate-standby-dblink`

この移行前評価では、Oracle スタンバイデータベースソース用に Dblink が作成されているかどうかを検証します。 AWSDMS_DBLINK は、スタンバイデータベースをソースとして使用するための前提条件です。Oracle スタンバイをソースとして使用する場合、 AWS DMS はデフォルトではオープントランザクションを検証しません。

詳細については、「のソースとしてのセルフマネージド Oracle データベースの使用 AWS DMS」を参照してください。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

LOB データ型と Binary Reader が設定されているかに関する Oracle の検証

API キー: `oracle-binary-lob-source-validation`

この移行前評価では、Oracle LogMiner が Oracle データベースエンドポイントバージョン 12c 以降で使用されているかどうかを検証します。 AWS DMS は、Oracle データベースバージョン 12c からの LOB 列の移行 LogMiner では Oracle をサポートしていません。この評価では、LOB 列があるかもチェックし、適切なレコメンデーションが提供されます。

Oracle を使用しないように移行を設定するには LogMiner、ソースエンドポイントに次の設定を追加します。

```
useLogMinerReader=N;useBfile=Y;
```

詳細については、「CDC での Oracle LogMiner または AWS DMS Binary Reader の使用」を参照してください。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

データベースが CDB かを検証します。

API キー: `oracle-validate-cdb`

この移行前評価では、データベースがコンテナデータベースであるかを検証します AWS DMS は、マルチテナントコンテナのルートデータベース (CDB$ROOT) をサポートしていません。

> **Note**
>
> この評価は、Oracle バージョン 12.1.0.1 以降でのみ必要です。この評価は、Oracle のバージョン 12.1.0.1 以前には適用されません。

詳細については、「のソースとしての Oracle の使用に関する制限 AWS DMS」を参照してください。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

Oracle Database Edition を確認します。

API キー: `oracle-check-cdc-support-express-edition`

この移行前評価では、Oracle ソースデータベースが Express Edition であるかを検証します。 AWS DMS は、Oracle Express Edition (Oracle Database XE) バージョン 18.0 以降の CDC をサポートしていません。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

DMS の Oracle CDC メソッドを検証する

API キー: `oracle-recommendation-cdc-method`

この移行前評価では、過去 7 日間の REDO ログ生成を検証し、CDC に AWS DMS Binary Reader と Oracle のどちらを使用するかを推奨 LogMiner します。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

使用する CDC 方法を決定する方法の詳細については、「CDC での Oracle LogMiner または AWS DMS Binary Reader の使用」を参照してください。

DMS の Oracle RAC 設定を検証する

API キー: `oracle-check-rac`

この移行前評価では、Oracle データベースが Real Application Cluster であるかを検証します。Real Application Cluster データベースは、適切に設定されている必要があります。データベースが RAC に基づいている場合は、Oracle ではなく CDC 用の AWS DMS Binary Reader を使用することをお勧めします LogMiner。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

詳細については、「CDC での Oracle LogMiner または AWS DMS Binary Reader の使用」を参照してください。

DMS ユーザーがターゲットに対するアクセス許可を持っているかどうかを検証する

API キー: `oracle-validate-permissions-on-target`

この移行前評価では、DMS ユーザーがターゲットデータベースに必要なすべてのアクセス許可を持っているかどうかを検証します。

すべての列に補足ログ記録が必要かどうかを検証する

API キー: `oracle-validate-supplemental-logging-all-columns`

この移行前評価では、タスクスコープで説明されているテーブルについて、プライマリキーまたは一意のキーのないテーブルのすべての列にサプリメンタルログが追加されているかどうかを検証します。プライマリキーまたは一意のキーがないテーブルのすべての列に対する補足ログ記録がないと、データの before-and-after イメージは REDO ログで使用できなくなります。DMS では、DML

ステートメントを生成するために、プライマリキーまたは一意のキーを持たないテーブルのサプリメンタルロギングが必要です。

プライマリキーまたは一意のキーを持つテーブルでサプリメンタルログ記録が有効になっているかどうかを検証する

API キー: `oracle-validate-supplemental-logging-for-pk`

この移行前評価では、プライマリキーまたは一意のインデックスを持つテーブルに対してサプリメンタルログが有効になっているかどうかを検証し、エンドポイントレベルで AddSupplementalLogging が有効になっているかどうかも確認します。DMS が変更をレプリケートできるようにするには、プライマリキーまたは一意のキーに基づいてテーブルレベルで補足ログを手動で追加するか、レプリケートされたテーブルに対する ALTER アクセス許可を持つ DMS ユーザーAddSupplementalLogging = trueでエンドポイント設定を使用します。

SecureFile LOBs があり、タスクがフル LOB モードに設定されているかどうかを検証する

API キー: `oracle-validate-securefile-lobs`

この移行前評価では、タスクスコープ内のテーブルに SecureFile LOBs が存在するかどうかをチェックし、LOB 設定を検証します。 SecureFile LOBs は現在、フル LOB モード中にのみサポートされることに注意してください。フル LOB モードでタスクを実行するとパフォーマンスが低下する可能性があるため、パフォーマンスを向上させるために LOB テーブルを別のタスクに割り当てることを検討してください。

タスクスコープに含まれるテーブル内で関数ベースのインデックスが使用されているかどうかを確認します。

API キー: `oracle-validate-function-based-indexes`

この移行前評価では、タスクスコープ内のテーブルの関数ベースのインデックスをチェックします。AWS DMS は関数ベースのインデックスのレプリケーションをサポートしていないことに注意してください。移行後にターゲットデータベースにインデックスを作成することを検討してください。

グローバル一時テーブルがタスクスコープに含まれるテーブルで使用されているかどうかを確認します。

API キー: `oracle-validate-global-temporary-tables`

この移行前評価では、グローバル一時テーブルがタスクテーブルマッピングスコープ内で使用されているかどうかをチェックします。 AWS DMS は、グローバル一時テーブルの移行またはレプリケートをサポートしていないことに注意してください。

オーバーフローセグメントを持つインデックスで編成されたテーブルが、タスクスコープに含まれるテーブルで使用されているかどうかを確認します。

API キー: `oracle-validate-iot-overflow-segments`

オーバーフローセグメントを持つインデックスが編成されたテーブルが、タスクスコープに含まれるテーブルで使用されているかどうかを確認します。オーバーフローセグメントを持つインデックスが編成されたテーブルの CDC をサポート AWS DMS していません。

タスクスコープに含まれるテーブルでマルチレベルネストテーブルが使用されているかどうかを確認します。

API キー: `oracle-validate-more-than-one-nesting-table-level`

この移行前評価では、タスクスコープで使用されるネストされたテーブルのネストレベルをチェックします。 は、1 レベルのテーブルネストのみ AWS DMS をサポートします。

タスクスコープに含まれるテーブルで、非表示の列が使用されているかどうかを確認します。

API キー: `oracle-validate-invisible-columns`

この移行前評価では、タスクスコープで使用されるテーブルに非表示の列があるかどうかを検証します。 AWS DMS は、ソースデータベースの非表示の列からデータを移行しません。表示されない列を移行するには、表示するように変更する必要があります。

ROWID 列に基づくマテリアライズドビューがタスクスコープに含まれるテーブルで使用されているかどうかを確認します。

API キー: `oracle-validate-rowid-based-materialized-views`

この移行前評価では、移行で使用されるマテリアライズドビューが ROWID 列に基づいて作成されるかどうかを検証します。 AWS DMS ROWID データ型または ROWID 列に基づくマテリアライズドビューはサポートされません。

Active Data Guard DML リダイレクト機能を使用しているかどうかを確認します。

API キー: `oracle-validate-adg-redirect-dml`

この移行前評価では、Active Data Guard DML リダイレクト機能を使用しているかどうかを検証します。Oracle 19.0 をソースとして使用する場合、Data AWS DMS Guard DML リダイレクト機能はサポートされていません。

ハイブリッドパーティションテーブルが使用されているかどうかを確認します。

API キー: `oracle-validate-hybrid-partitioned-tables`

この移行前評価では、ハイブリッドパーティションテーブルがタスクスコープで定義されたテーブルに使用されるかどうかを検証します。

スキーマのみの Oracle アカウントが使用されているかどうかを検証する

API キー: `oracle-validate-schema-only-accounts`

この移行前評価では、スキーマ専用アカウントがタスク範囲内にあるかどうかを検証します。

仮想列が使用されているかどうかを検証する

API キー: `oracle-validate-virtual-columns`

この移行前評価では、Oracle インスタンスにタスク範囲内のテーブルに仮想列があるかどうかを検証します。

タスクスコープで定義されたテーブル名にアポストロフィが含まれているかどうかを確認します。

API キー: `oracle-validate-names-with-apostrophes`

この移行前評価では、タスクスコープで使用されるテーブルにアポストロフィが含まれているかどうかを検証します。 AWS DMS はアポストロフィを含む名前のテーブルをレプリケートしません。特定された場合は、そのようなテーブルの名前を変更することを検討してください。または、アポストロフィなしでビューまたはマテリアライズドビューを作成して、これらのテーブルをロードすることもできます。

タスクスコープで定義された列に **XMLType**、**Long**、または **Long Raw** データ型があるかどうかを検証し、タスク設定で LOB モード設定を確認します。

API キー: `oracle-validate-limited-lob-mode-for-longs`

この移行前評価では、タスクスコープで定義されたテーブルに XMLType、Long、または のデータ型があるかどうかを検証しLong Raw、タスク設定が制限サイズ LOB Mode を使用するように設定されているかどうかを確認します。 AWS DMS は、フル LOB モードを使用したこれらのデータ型のレプリケーションをサポートしていません。このようなデータ型を持つテーブルを識別するときは、制限サイズ LOB モードを使用するようにタスク設定を変更することを検討してください。

ソース Oracle バージョンが でサポートされているかどうかを確認します AWS DMS。

API キー: `oracle-validate-supported-versions-of-source`

この移行前評価では、ソース Oracle インスタンスのバージョンが でサポートされているかどうかを検証します AWS DMS。

ターゲットの Oracle バージョンが でサポートされているかどうかを確認します AWS DMS。

API キー: `oracle-validate-supported-versions-of-target`

この移行前評価では、ターゲットの Oracle インスタンスバージョンが でサポートされているかどうかを検証します AWS DMS。

ターゲットの Oracle バージョンが でサポートされているかどうかを確認します AWS DMS。

API キー: `oracle-validate-supported-versions-of-target`

この移行前評価では、ターゲットの Oracle インスタンスバージョンが でサポートされているかどうかを検証します AWS DMS。

DMS ユーザーがデータ検証を使用するために必要なアクセス許可を持っているかどうかを確認します。

API キー: `oracle-prerequisites-privileges-of-validation-feature`

この移行前評価では、DMS ユーザーが DMS データ検証を使用するために必要な権限を持っているかどうかを検証します。データ検証を使用する予定がない場合は、この検証の有効化を無視できます。

DMS ユーザーが Oracle ASM で Binary Reader を使用するアクセス許可を持っているかどうかを検証する

API キー: `oracle-prerequisites-privileges-of-binary-reader-asm`

この移行前評価では、DMS ユーザーが Oracle ASM インスタンスで Binary Reader を使用するために必要な権限を持っているかどうかを検証します。ソースが Oracle ASM インスタンスでない場合、または CDC に Binary Reader を使用していない場合は、この評価の有効化を無視できます。

DMS ユーザーが Oracle 非 ASM で Binary Reader を使用するアクセス許可を持っているかどうかを検証する

API キー: `oracle-prerequisites-privileges-of-binary-reader-non-asm`

この移行前評価では、DMS ユーザーが Oracle 非 ASM インスタンスで Binary Reader を使用するために必要な権限を持っているかどうかを検証します。この評価は、Oracle の ASM 以外のインスタンスがある場合にのみ有効です。

DMS ユーザーが CopyToTempFolder メソッドで Binary Reader を使用するアクセス許可を持っているかどうかを検証する

API キー: `oracle-prerequisites-privileges-of-binary-reader-copy-to-temp-folder`

この移行前評価では、DMS ユーザーが「Copy to Temp Folder」メソッドでバイナリリーダーを使用するために必要な権限を持っているかどうかを検証します。この評価は、Binary Reader の使用中に CopyToTempFolder を使用して CDC の変更を読み取る予定があり、ソースに ASM インスタンスが接続されている場合にのみ関係します。 CopyToTempFolder この機能を使用する予定がない場合は、この評価の有効化を無視できます。

CopyToTempFolder この機能の使用は推奨されません。この機能は非推奨です。

DMS ユーザーが Oracle スタンバイをソースとして使用するアクセス許可を持っているかどうかを検証する

API キー: `oracle-prerequisites-privileges-of-standby-as-source`

この移行前評価では、DMS ユーザーが StandBy Oracle インスタンスをソースとして使用するのに必要な権限を持っているかどうかを検証します。 StandBy Oracle インスタンスをソースとして使用する予定がない場合は、この評価の有効化を無視できます。

DMS ソースがアプリケーションコンテナ PDB に接続されているかどうかを検証する

API キー: `oracle-check-app-pdb`

この移行前評価では、DMS ソースがアプリケーションコンテナ PDB に接続されているかどうかを検証します。DMS は、アプリケーションコンテナ PDB からのレプリケーションをサポートしていません。

テーブルに XML データ型がタスクスコープに含まれているかどうかを確認します。

API キー: `oracle-check-xml-columns`

この移行前評価では、タスクスコープで使用されるテーブルに XML データ型があるかどうかを検証します。また、テーブルに XML データ型が含まれている場合に、タスクが制限付き LOB モードに設定されているかどうかも確認します。DMS は、Oracle XML 列を移行するための制限付き LOB モードのみをサポートしています。

ソースデータベースでアーカイブログモードが有効になっているかどうかを確認します。

API キー: `oracle-check-archivelog-mode`

この移行前評価では、ソースデータベースでアーカイブログモードが有効になっているかどうかを検証します。DMS が変更をレプリケートするには、ソースデータベースでアーカイブログモードを有効にする必要があります。

RDS Oracle のアーカイブログの保持を検証します。

API キー: `oracle-check-archivelog-retention-rds`

この移行前評価では、RDS Oracle データベースのアーカイブログの保持期間が少なくとも 24 時間設定されているかどうかを検証します。

テーブルに、タスクスコープに含まれる拡張データ型があるかどうかを検証します。

API キー: `oracle-check-extended-columns`

この移行前評価では、タスクスコープで使用されるテーブルに拡張データ型があるかどうかを検証します。拡張データ型は DMS バージョン 3.5 以降でのみサポートされることに注意してください。

タスクスコープに含まれるオブジェクト名の長さを検証します。

API キー: `oracle-check-object-30-bytes-limit`

この移行前評価では、オブジェクト名の長さが 30 バイトを超えているかどうかを検証します。DMS は長いオブジェクト名 (30 バイト以上) をサポートしていません。

DMS ソースが Oracle PDB に接続されているかどうかを検証する

API キー: `oracle-check-pdb-enabled`

この移行前評価では、DMS ソースが PDB に接続されているかどうかを検証します。DMS は、Oracle PDB をソースとして Binary Reader を使用する場合にのみ CDC をサポートします。この評価では、DMS が Oracle PDB に接続されているときにタスクがバイナリリーダーを使用するように設定されているかどうかも評価されます。

テーブルにタスクスコープに含まれる空間列があるかどうかを検証します。

API キー: `oracle-check-spatial-columns`

この移行前評価では、テーブルに空間列がタスクスコープに含まれているかどうかを検証します。DMS は、フル LOB モードを使用する空間データ型のみをサポートします。この評価では、DMS が空間列を識別するときに、タスクがフル LOB モードを使用するように設定されているかどうかも評価されます。

DMS ソースが Oracle スタンバイに接続されているかどうかを確認します。

API キー: `oracle-check-standby-db`

この移行前評価では、ソースが Oracle スタンバイに接続されているかどうかを検証します。DMS は、Oracle Standby をソースとしてバイナリリーダーを使用する場合にのみ CDC をサポートします。この評価では、DMS が Oracle スタンバイに接続されているときに、タスクがバイナリリーダーを使用するように設定されているかどうかも評価されます。

ソースデータベースのテーブルスペースが TDE を使用して暗号化されているかどうかを確認します。

API キー: `oracle-check-tde-enabled`

この移行前評価では、ソースがテーブルスペースで TDE 暗号化を有効にしているかどうかを検証します。DMS は、Oracle LogMiner for RDS Oracle を使用する場合、暗号化されたテーブルスペースでのみ TDE をサポートします。

ソースデータベースが Oracle ASM かどうかを検証する

API キー: `oracle-check-asm`

この移行前評価では、ソースが ASM を使用しているかどうかを検証します。ASM 設定のパフォーマンスを向上させるには、ソースエンドポイント設定`readAheadBlocks`に`parallelASMReadThreads`と を追加することを検討してください。

Sql サーバー評価

このセクションでは、Microsoft SQL Server のソースエンドポイントを使用する移行タスクの個別の移行前評価について説明します。

トピック

- データベースの復旧モデルがシンプルかを確認する
- タスクの範囲内のテーブルに計算列が含まれているかを確認する
- タスクの範囲内のテーブルに列ストアインデックスがあるかを確認する
- メモリ最適化テーブルがタスクの範囲に入っているかを確認する
- テンポラルテーブルがタスクの範囲に入っているかを確認する
- 遅延持続性がデータベースレベルで有効になっているかを確認する
- 高速データ復旧がデータベースレベルで有効になっているかを確認する

- テーブルマッピングのプライマリキーを持つテーブルが 10K を超えているかを確認する
- ソースデータベースに特殊文字を含むテーブルまたはスキーマ名があるかどうかを確認します。
- ソースデータベースに、マスキングされたデータを含む列名があるかどうかを確認します。
- ソースデータベースに暗号化されたバックアップがあるかどうかを確認する
- ソースデータベースに URL または Windows Azure にバックアップが保存されているかどうかを確認します。
- ソースデータベースに複数のディスクにバックアップがあるかどうかを確認する
- ソースデータベースに少なくとも 1 つのフルバックアップがあるかどうかを確認します。
- ソースデータベースにスパース列と列構造圧縮があるかどうかを確認します。
- ソースデータベースインスタンスに SQL Server 2008 または SQL Server 2008 R2 のサーバーレベルの監査があるかどうかを確認します。
- ソースデータベースにフル LOB モードのジオメトリ列があるかどうかを確認します。
- ソースデータベースに Identity プロパティを持つ列があるかどうかを確認します。
- DMS ユーザーに FULL LOAD アクセス許可があるかどうかを確認します。
- DMS ユーザーに FULL LOAD および CDC または CDC のみのアクセス許可があるかどうかを確認します。
- オンプレミスデータベースまたは ignoreMsReplicationEnablement EC2 データベースで MS-CDC を使用するときに ECA が設定されているかどうかを確認する
- DMS ユーザーに VIEW DEFINITION アクセス許可があるかどうかを確認します。
- DMS ユーザーが Sysadmin ロールを持たないユーザーの MASTER データベースに対する DATABASE STATE の表示アクセス許可を持っているかどうかを確認します。
- DMS ユーザーが SERVER STATE の表示アクセス許可を持っているかどうかを確認します。

データベースの復旧モデルがシンプルかを確認する

API キー: `sqlserver-check-for-recovery-model`

この移行前評価では、ソースエンドポイント復旧モデルを検証します。 AWS DMS では、継続的なレプリケーションFullのために復旧モデルを Bulk loggedまたは に設定する必要があります。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

詳細については、「SQL Server のソースからの継続的なレプリケーション (CDC) を使用するための前提条件」を参照してください。

タスクの範囲内のテーブルに計算列が含まれているかを確認する

API キー: `sqlserver-check-for-computed-fields`

この移行前評価では、計算された列が存在するかどうかをチェックします。 AWS DMS は、SQL Server の計算された列からの変更のレプリケーションをサポートしていません。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

タスクの範囲内のテーブルに列ストアインデックスがあるかを確認する

API キー: `sqlserver-check-for-columnstore-indexes`

この移行前評価では、列ストアインデックスを持つテーブルが存在するかどうかをチェックします。 AWS DMS は、列ストアインデックスを持つ SQL Server テーブルからの変更のレプリケーションをサポートしていません。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

メモリ最適化テーブルがタスクの範囲に入っているかを確認する

API キー: `sqlserver-check-for-memory-optimized-tables`

この移行前評価では、メモリ最適化テーブルの有無をチェックします。 AWS DMS は、メモリ最適化テーブルからの変更のレプリケーションをサポートしていません。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

テンポラルテーブルがタスクの範囲に入っているかを確認する

API キー: `sqlserver-check-for-temporal-tables`

この移行前評価では、テンポラリテーブルの有無をチェックします。 AWS DMS は、テンポラリテーブルからの変更のレプリケーションをサポートしていません。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

遅延持続性がデータベースレベルで有効になっているかを確認する

API キー: `sqlserver-check-for-delayed-durability`

この移行前評価では、遅延耐久性の有無をチェックします。 AWS DMS は、遅延耐久性を使用するトランザクションからの変更のレプリケーションをサポートしていません。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

高速データ復旧がデータベースレベルで有効になっているかを確認する

API キー: `sqlserver-check-for-accelerated-data-recovery`

この移行前評価では、高速データ復旧の有無をチェックします。 AWS DMS は、高速データ復旧によるデータベースからの変更のレプリケーションをサポートしていません。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

テーブルマッピングのプライマリキーを持つテーブルが 10K を超えているかを確認する

API キー: `sqlserver-large-number-of-tables`

この移行前評価では、プライマリキーを持つテーブルが 10,000 以上あるかをチェックします。MS-Replication で設定されたデータベースでは、プライマリキーがあるテーブル数が過剰な場合、タスクが失敗する可能性があります。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

MS-Replication の設定の詳細については、「オンプレミスまたは Amazon EC2 上のセルフマネージド型 SQL Server のデータ変更のキャプチャ」を参照してください。

ソースデータベースに特殊文字を含むテーブルまたはスキーマ名があるかどうかを確認します。

API キー: `sqlserver-check-for-special-characters`

この移行前評価では、ソースデータベースに次のセットの文字を含むテーブル名またはスキーマ名があるかどうかを検証します。

```
\\ -- \n \" \b \r ' \t ;
```

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

ソースデータベースに、マスキングされたデータを含む列名があるかどうかを確認します。

API キー: `sqlserver-check-for-masked-data`

この移行前評価では、ソースデータベースにマスキングされたデータがあるかどうかを検証します。はマスキングなしでマスキングされたデータを AWS DMS 移行します。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

ソースデータベースに暗号化されたバックアップがあるかどうかを確認する

API キー: `sqlserver-check-for-encrypted-backups`

この移行前評価では、ソースデータベースに暗号化されたバックアップがあるかどうかを検証します。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

ソースデータベースに URL または Windows Azure にバックアップが保存されているかどうかを確認します。

API キー: `sqlserver-check-for-backup-url`

この移行前評価では、ソースデータベースのバックアップが URL または Windows Azure に保存されているかどうかを確認します。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

ソースデータベースに複数のディスクにバックアップがあるかどうかを確認する

API キー: `sqlserver-check-for-backup-multiple-stripes`

この移行前評価では、ソースデータベースに複数のディスクのバックアップがあるかどうかを検証します。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

ソースデータベースに少なくとも 1 つのフルバックアップがあるかどうかを確認します。

API キー: `sqlserver-check-for-full-backup`

この移行前評価では、ソースデータベースに少なくとも 1 つのフルバックアップがあるかどうかを検証します。SQL Server はフルバックアップ用に設定する必要があり、データをレプリケートする前にバックアップを実行する必要があります。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

ソースデータベースにスパース列と列構造圧縮があるかどうかを確認します。

API キー: `sqlserver-check-for-sparse-columns`

この移行前評価では、ソースデータベースにスパース列と列構造圧縮があるかどうかを検証します。DMS は、スパース列と列構造圧縮をサポートしていません。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

ソースデータベースインスタンスに SQL Server 2008 または SQL Server 2008 R2 のサーバーレベルの監査があるかどうかを確認します。

API キー: `sqlserver-check-for-audit-2008`

この移行前評価では、ソースデータベースが SQL Server 2008 または SQL Server 2008 R2 のサーバーレベルの監査を有効にしているかどうかを検証します。DMS には、SQL Server 2008 および 2008 R2 に関連する既知の問題があります。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

ソースデータベースにフル LOB モードのジオメトリ列があるかどうかを確認します。

API キー: `sqlserver-check-for-geometry-columns`

この移行前評価では、SQL Server をソースとして使用するときに、ソースデータベースにフルラージオブジェクト (LOB) モードのジオメトリ列があるかどうかを検証します。データベースにジオメトリ列が含まれている場合は、制限付き LOB モードを使用するか、インライン LOB モードを使用するように`InlineLobMaxSize`タスク設定を設定することをお勧めします。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

ソースデータベースに Identity プロパティを持つ列があるかどうかを確認します。

API キー: `sqlserver-check-for-identity-columns`

この移行前評価では、ソースデータベースに `IDENTITY`プロパティを持つ列があるかどうかを検証します。DMS は、このプロパティを対応するターゲットデータベース列に移行しません。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

DMS ユーザーに FULL LOAD アクセス許可があるかどうかを確認します。

API キー: `sqlserver-check-user-permission-for-full-load-only`

この移行前評価では、DMS タスクのユーザーに FULL LOAD モードでタスクを実行するアクセス許可があるかどうかを検証します。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

DMS ユーザーに FULL LOAD および CDC または CDC のみのアクセス許可があるかどうかを確認します。

API キー: `sqlserver-check-user-permission-for-cdc`

この移行前評価では、DMS ユーザーが FULL LOAD and CDCまたは モードでタスクを実行するアクセス許可を持っているかどうかを確認しますCDC only。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

オンプレミスデータベースまたは **ignoreMsReplicationEnablement** EC2 データベースで MS-CDC を使用するときに ECA が設定されているかどうかを確認する

API キー: sqlserver-check-attribute-for-enable-ms-cdc-onprem

オンプレミスまたは EC2 データベースで MS-CDC を使用するときに、ignoreMsReplicationEnablement追加の接続属性 (ECA) が設定されているかどうかを確認します。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

DMS ユーザーに VIEW DEFINITION アクセス許可があるかどうかを確認します。

API キー: sqlserver-check-user-permission-on-view-definition

この移行前評価では、エンドポイント設定で指定されたユーザーに アクセスVIEW DEFINITION許可があるかどうかを検証します。DMS には、オブジェクト定義を表示するための VIEW DEFINITION アクセス許可が必要です。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

DMS ユーザーが Sysadmin ロールを持たないユーザーの MASTER データベースに対する DATABASE STATE の表示アクセス許可を持っているかどうかを確認します。

API キー: sqlserver-check-user-permission-on-view-database-state

この移行前評価では、エンドポイント設定で指定されたユーザーに アクセスVIEW DATABASE STATE許可があるかどうかを検証します。DMS では、MASTER データベース内のデータベースオブジェクトにアクセスするためにこのアクセス許可が必要です。ユーザーが sysadmin 権限を持っていない場合、DMS にもこのアクセス許可が必要です。DMS では、関数、証明書、ログインを作成し、認証情報を付与するためにこのアクセス許可が必要です。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

DMS ユーザーが SERVER STATE の表示アクセス許可を持っているかどうかを確認します。

API キー: `sqlserver-check-user-permission-on-view-server-state`

この移行前評価では、追加の接続属性 (ECA) で指定されたユーザーに `VIEW SERVER STATE`アクセス許可があるかどうかをチェックします。 `VIEW SERVER STATE`は、サーバー全体の情報と状態をユーザーが表示できるようにするサーバーレベルのアクセス許可です。このアクセス許可は、SQL Server DMVs) と動的管理関数 (DMFsへのアクセスを提供します。このアクセス許可は、DMS ユーザーが CDC リソースにアクセスするために必要です。このアクセス許可は、 `FULL LOAD and CDC`または モードで DMS タスクを実行するために必要です`CDC only`。

詳細については、「のソースとして SQL Server を使用する場合の制限 AWS DMS」を参照してください。

MySQL 評価

このセクションでは、MySQL ソースエンドポイントを使用する移行タスクの個々の移行前評価について説明します。

トピック

- テーブルが Innodb 以外のストレージエンジンを使用しているかどうかを検証する
- 移行に使用されるテーブルで自動インクリメントが有効になっているかどうかを検証する
- DMS CDC をサポートするFULLようにデータベースバイナリログイメージが に設定されているかどうかを検証する
- ソースデータベースが MySQL Read-Replica であるかどうかを検証する
- テーブルにパーティションがあるかどうかを検証し、フルロードタスク設定 target_table_prep_modeに を推奨します。
- DMS がデータベースバージョンをサポートしているかどうかを検証する
- ターゲットデータベースが を 1 local_infileに設定するように設定されているかどうかを検証する
- ターゲットデータベースに外部キーを持つテーブルがあるかどうかを検証する
- タスクスコープ内のソーステーブルにカスケード制約があるかどうかを検証する
- タイムアウト値が MySQL ソースまたはターゲットに適しているかどうかを検証する

テーブルが Innodb 以外のストレージエンジンを使用しているかどうかを検証する

API キー: `mysql-check-table-storage-engine`

この移行前評価では、ソース MySQL データベース内のテーブルに使用されるストレージエンジンが Innodb 以外のエンジンであるかどうかを検証します。DMS は、デフォルトで InnoDB ストレージエンジンを使用してターゲットテーブルを作成します。InnoDB 以外のストレージエンジンを使用する必要がある場合は、ターゲットデータベースにテーブルを手動で作成し、TRUNCATE_BEFORE_LOADまたは をDO_NOTHING全ロードタスク設定として使用するように DMS タスクを設定する必要があります。フルロードタスク設定の詳細については、「」を参照してください全ロードタスク設定。

MySQL エンドポイントの制限の詳細については、「」を参照してくださいMySQL データベースをソースとして使用する場合の制限事項 AWS DMS。

移行に使用されるテーブルで自動インクリメントが有効になっているかどうかを検証する

API キー: `mysql-check-auto-increment`

この移行前評価では、タスクで使用されるソーステーブルで自動インクリメントが有効になっているかどうかを検証します。DMS は、列の AUTO_INCREMENT 属性をターゲットデータベースに移行しません。

MySQL エンドポイントの制限の詳細については、「」を参照してくださいMySQL データベースをソースとして使用する場合の制限事項 AWS DMS。MySQL での ID 列の処理については、「 のIDENTITY 列の処理 AWS DMS: パート 2」を参照してください。

DMS CDC をサポートする**FULL**ようにデータベースバイナリログイメージが に設定されているかどうかを検証する

API キー: `mysql-check-binlog-image`

この移行前評価では、ソースデータベースのバイナリログイメージが に設定されているかどうかをチェックしますFULL。MySQL では、 binlog_row_image変数は、 ROW形式を使用するときにバイナリログイベントがどのように書き込まれるかを決定します。DMS との互換性を確保し、CDC をサポートするには、 binlog_row_image変数を に設定しますFULL。この設定により、DMS は移行中にターゲットデータベースの完全なデータ操作言語 (DML) を構築するための十分な情報を確実に受け取ることができます。

バイナリログイメージを に設定するにはFULL、次の手順を実行します。

- Amazon RDS の場合、この値はFULLデフォルトで です。
- オンプレミスまたは Amazon EC2 で接続されたデータベースの場合は、 my.ini (Microsoft Windows) または (UNIX) my.cnf でbinlog_row_image値を設定します。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

ソースデータベースが MySQL Read-Replica であるかどうかを検証する

API キー: `mysql-check-database-role`

この移行前評価では、ソースデータベースがリードレプリカであるかどうかを検証します。リードレプリカに接続したときに DMS の CDC サポートを有効にするには、 `log_slave_updates`パラメータを に設定しますTrue。セルフマネージド MySQL データベースの使用の詳細については、「」を参照してください自己管理型のMySQL互換データベースをソースとして使用する AWS DMS。

`log_slave_updates` 値を に設定するにはTrue、次の手順を実行します。

- Amazon RDS の場合は、データベースのパラメータグループを使用します。RDS データベースパラメータグループの使用の詳細については、「Amazon RDS ユーザーガイド」の「パラメータグループの使用」を参照してください。
- オンプレミスまたは Amazon EC2 で接続されたデータベースの場合は、 `my.ini` (Microsoft Windows) または (UNIX) `my.cnf` で`log_slave_updates`値を設定します。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

テーブルにパーティションがあるかどうかを検証し、フルロードタスク設定**`target_table_prep_mode`**に を推奨します。

API キー: `mysql-check-table-partition`

この移行前評価では、ソースデータベースにパーティションを持つテーブルが存在するかどうかをチェックします。DMS は、MySQL ターゲットにパーティションのないテーブルを作成します。パーティション分割されたテーブルをターゲットのパーティション分割されたテーブルに移行するには、以下を実行する必要があります。

- ターゲット MySQL データベースにパーティション分割されたテーブルを事前に作成します。
- `TRUNCATE_BEFORE_LOAD` または をフルロードタスク設定`DO_NOTHING`として使用するように DMS タスクを設定します。

MySQL エンドポイントの制限の詳細については、「」を参照してくださいMySQL データベースをソースとして使用する場合の制限事項 AWS DMS。

DMS がデータベースバージョンをサポートしているかどうかを検証する

API キー: `mysql-check-supported-version`

この移行前評価では、ソースデータベースのバージョンが DMS と互換性があるかどうかを検証します。CDC は、Amazon RDS MySQL バージョン 5.5 以前、または MySQL バージョン 8.0.x 以降ではサポートされていません。CDC は、MySQL バージョン 5.6、5.7、または 8.0 でのみサポートされています。サポートされている MySQL バージョンの詳細については、「」を参照してくださいデータ移行のソースエンドポイント。

ターゲットデータベースが を 1 **`local_infile`**に設定するように設定されているかどうかを検証する

API キー: `mysql-check-target-localinfile-set`

この移行前評価では、ターゲットデータベースの `local_infile`パラメータが 1 に設定されているかどうかをチェックします。DMS では、ターゲットデータベースへの全ロード中に「local_infile」パラメータを 1 に設定する必要があります。詳細については、「AWS DMSを使用して MySQL から MySQL へ移行します。」を参照してください。

この評価は、フルロードまたはフルロード、CDC タスクに対してのみ有効です。

ターゲットデータベースに外部キーを持つテーブルがあるかどうかを検証する

API キー: `mysql-check-fk-target`

この移行前評価では、MySQL データベースに移行する全ロードまたは全ロードと CDC タスクに外部キーを含むテーブルがあるかどうかをチェックします。DMS のデフォルト設定では、テーブルをアルファベット順にロードします。外部キーと参照整合性の制約があるテーブルは、親テーブルと子テーブルが同時にロードされない可能性があるため、ロードが失敗する可能性があります。

DMS での参照整合性の詳細については、AWS DMS 移行のパフォーマンスの向上「」トピックの「インデックス、トリガー、参照整合性制約の使用」を参照してください。

タスクスコープ内のソーステーブルにカスケード制約があるかどうかを検証する

API キー: `mysql-check-cascade-constraints`

この移行前評価では、MySQL ソーステーブルにカスケード制約があるかどうかをチェックします。MySQL はこれらのイベントの変更をバイナリログに記録しないため、カスケード制約は DMS タスクによって移行またはレプリケートされません。 AWS DMS はこれらの制約をサポートしていませんが、リレーショナルデータベースターゲットの回避策を使用できます。

大文字と小文字の制約やその他の制約のサポートについては、「 インデックス、外部キー、カスケード更新、または削除が移行されないトピックの移行タスクのトラブルシューティング」の AWS DMS「」を参照してください。

タイムアウト値が MySQL ソースまたはターゲットに適しているかどうかを検証する

API キー: `mysql-check-network-parameter`

この移行前評価では、タスクの MySQL エンドポイントに 、`net_read_timeout`、`net_wait_timeout` `wait_timeout`設定が少なくとも 300 秒に設定されているかどうかをチェックします。これは、移行中の切断を防ぐために必要です。

詳細については、「ターゲット MySQL のインスタンスへの接続は、タスクの実行中に接続が切断されます」を参照してください。

MariaDB 評価

このセクションでは、MariaDB ソースエンドポイントを使用する移行タスクの個々の移行前評価について説明します。

AWS DMS API を使用して個々の移行前評価を作成するには、StartReplicationTaskAssessmentRunアクションの `Include`パラメータにリストされている API キーを使用します。

トピック

- テーブルが Innodb 以外のストレージエンジンを使用しているかどうかを検証する
- 移行に使用されるテーブルで自動インクリメントが有効になっているかどうかを検証する
- DMS CDC をサポートするROWようにデータベースバイナリログ形式が に設定されているかどうかを検証する
- DMS CDC をサポートするFULLようにデータベースバイナリログイメージが に設定されているかどうかを検証する
- ソースデータベースが MariaDB リードレプリカであるかどうかを検証する
- テーブルにパーティションがあるかどうかを検証し、フルロードタスク設定DO_NOTHINGにTRUNCATE_BEFORE_LOADまたは を推奨します。
- DMS がデータベースバージョンをサポートしているかどうかを検証する
- ターゲットデータベースが を 1 local_infileに設定するように設定されているかどうかを検証する
- ターゲットデータベースに外部キーを持つテーブルがあるかどうかを検証する
- タスクスコープ内のソーステーブルにカスケード制約があるかどうかを検証する

- タスクスコープのソーステーブルで列が生成されているかどうかを検証する
- タイムアウト値が MariaDB ソースに適しているかどうかを検証する
- タイムアウト値が MariaDB ターゲットに適しているかどうかを検証する

テーブルが Innodb 以外のストレージエンジンを使用しているかどうかを検証する

API キー: `mariadb-check-table-storage-engine`

この移行前評価では、ソース MariaDB データベース内のテーブルに使用されるストレージエンジンが Innodb 以外のエンジンであるかどうかを検証します。DMS は、デフォルトで InnoDB ストレージエンジンを使用してターゲットテーブルを作成します。InnoDB 以外のストレージエンジンを使用する必要がある場合は、ターゲットデータベースにテーブルを手動で作成し、`TRUNCATE_BEFORE_LOAD`または `DO_NOTHING`全ロードタスク設定として使用するように DMS タスクを設定する必要があります。フルロードタスク設定の詳細については、「」を参照してください全ロードタスク設定。

MariaDB エンドポイントの制限の詳細については、「」を参照してくださいMySQL データベースをソースとして使用する場合の制限事項 AWS DMS。

移行に使用されるテーブルで自動インクリメントが有効になっているかどうかを検証する

API キー: `mariadb-check-auto-increment`

この移行前評価では、タスクで使用されるソーステーブルで自動インクリメントが有効になっているかどうかを検証します。DMS は、列の AUTO_INCREMENT 属性をターゲットデータベースに移行しません。

MariaDB エンドポイントの制限の詳細については、「」を参照してくださいMySQL データベースをソースとして使用する場合の制限事項 AWS DMS。MariaDB での ID 列の処理については、「 のIDENTITY 列の処理 AWS DMS: パート 2」を参照してください。

DMS CDC をサポートする**`ROW`**ようにデータベースバイナリログ形式が に設定されているかどうかを検証する

API キー: `mariadb-check-binlog-format`

この移行前評価では、ソースデータベースのバイナリログ形式が DMS 変更データキャプチャ (CDC) をサポートする`ROW`ように に設定されているかどうかを検証します。

バイナリログ形式を に設定するには`ROW`、次の手順を実行します。

- Amazon RDS の場合は、データベースのパラメータグループを使用します。RDS パラメータグループの使用の詳細については、「Amazon RDS ユーザーガイド」のMySQL バイナリログ記録の設定」を参照してください。
- オンプレミスまたは Amazon EC2 で接続されたデータベースの場合は、 `my.ini` (Microsoft Windows) または (UNIX) `my.cnf` で `binlog_format` 値を設定します。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

セルフホスト MariaDB サーバーの詳細については、「」を参照してください自己管理型のMySQL互換データベースをソースとして使用する AWS DMS。

DMS CDC をサポートする**FULL**ようにデータベースバイナリログイメージが に設定されているかどうかを検証する

API キー: `mariadb-check-binlog-image`

この移行前評価では、ソースデータベースのバイナリログイメージが に設定されているかどうかをチェックしますFULL。MariaDB では、 `binlog_row_image`変数は、 ROW形式を使用するときにバイナリログイベントを書き込む方法を決定します。DMS との互換性を確保し、CDC をサポートするには、 `binlog_row_image`変数を に設定しますFULL。この設定により、DMS は移行中にターゲットデータベースの完全なデータ操作言語 (DML) を構築するための十分な情報を確実に受け取ることができます。

バイナリログイメージを に設定するにはFULL、次の手順を実行します。

- Amazon RDS の場合、この値はFULLデフォルトで です。
- オンプレミスまたは Amazon EC2 で接続されたデータベースの場合は、 `my.ini` (Microsoft Windows) または (UNIX) `my.cnf` で`binlog_row_image`値を設定します。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

セルフホスト MariaDB サーバーの詳細については、「」を参照してください自己管理型のMySQL互換データベースをソースとして使用する AWS DMS。

ソースデータベースが MariaDB リードレプリカであるかどうかを検証する

API キー: `mariadb-check-database-role`

この移行前評価では、ソースデータベースがリードレプリカであるかどうかを検証します。リードレプリカに接続したときに DMS の CDC サポートを有効にするには、 `log_slave_updates`パラメータを に設定します`True`。セルフマネージド MySQL データベースの使用の詳細については、「」を参照してください自己管理型のMySQL互換データベースをソースとして使用する AWS DMS。

`log_slave_updates` 値を に設定するには`True`、次の手順を実行します。

- Amazon RDS の場合は、データベースのパラメータグループを使用します。RDS データベースパラメータグループの使用の詳細については、「Amazon RDS ユーザーガイド」の「パラメータグループの使用」を参照してください。
- オンプレミスまたは Amazon EC2 で接続されたデータベースの場合は、 `my.ini` (Microsoft Windows) または (UNIX) `my.cnf` で`log_slave_updates`値を設定します。

この評価は、フルロードと CDC の移行、CDC のみの移行の場合にのみ利用できます。この評価は、フルロードのみの移行では有効ではありません。

テーブルにパーティションがあるかどうかを検証し、フルロードタスク設定**DO_NOTHING**に**TRUNCATE_BEFORE_LOAD**または を推奨します。

API キー: `mariadb-check-table-partition`

この移行前評価では、ソースデータベースにパーティションを持つテーブルが存在するかどうかをチェックします。DMS は MariaDB ターゲットにパーティションのないテーブルを作成します。パーティション分割されたテーブルをターゲットのパーティション分割されたテーブルに移行するには、次の手順を実行する必要があります。

- ターゲット MariaDB データベースにパーティション分割されたテーブルを事前に作成します。
- `TRUNCATE_BEFORE_LOAD` または をフルロードタスク設定`DO_NOTHING`として使用するように DMS タスクを設定します。

MariaDB エンドポイントの制限の詳細については、「」を参照してくださいMySQL データベースをソースとして使用する場合の制限事項 AWS DMS。

DMS がデータベースバージョンをサポートしているかどうかを検証する

API キー: `mariadb-check-supported-version`

この移行前評価では、ソースデータベースのバージョンが DMS と互換性があるかどうかを検証します。CDC は、Amazon RDS MariaDB バージョン 10.4 以前、または MySQL バージョン 10.11 以降

ではサポートされていません。サポートされている MariaDB バージョンの詳細については、「」を参照してくださいデータ移行のソースエンドポイント。

ターゲットデータベースが を 1 **local_infile**に設定するように設定されているかどうかを検証する

API キー: `mariadb-check-target-localinfile-set`

この移行前評価では、ターゲットデータベースの `local_infile`パラメータが 1 に設定されているかどうかをチェックします。DMS では、ターゲットデータベースへの全ロード中に「local_infile」パラメータを 1 に設定する必要があります。詳細については、「AWS DMSを使用して MySQL から MySQL へ移行します。」を参照してください。

この評価は、全ロードタスクに対してのみ有効です。

ターゲットデータベースに外部キーを持つテーブルがあるかどうかを検証する

API キー: `mariadb-check-fk-target`

この移行前評価では、MariaDB データベースに移行する全ロードまたは全ロードと CDC タスクに外部キーを含むテーブルがあるかどうかをチェックします。DMS のデフォルト設定では、テーブルをアルファベット順にロードします。外部キーと参照整合性の制約があるテーブルは、親テーブルと子テーブルが同時にロードされない可能性があるため、ロードが失敗する可能性があります。

DMS での参照整合性の詳細については、AWS DMS 移行のパフォーマンスの向上「」トピックの「インデックス、トリガー、参照整合性制約の使用」を参照してください。

タスクスコープ内のソーステーブルにカスケード制約があるかどうかを検証する

API キー: `mariadb-check-cascade-constraints`

この移行前評価では、MariaDB ソーステーブルにカスケード制約があるかどうかをチェックします。MariaDB はこれらのイベントの変更をバイナリログに記録しないため、カスケード制約は DMS タスクによって移行またはレプリケートされません。 AWS DMS はこれらの制約をサポートしていませんが、リレーショナルデータベースターゲットの回避策を使用できます。

大文字と小文字の制約やその他の制約のサポートについては、「 インデックス、外部キー、カスケード更新、または削除が移行されないトピックの移行タスクのトラブルシューティング」の AWS DMS「」を参照してください。

タスクスコープのソーステーブルで列が生成されているかどうかを検証する

API キー: `mariadb-check-generated-columns`

この移行前評価では、MariaDB ソーステーブルのいずれかで列が生成されたかどうかを確認します。DMS タスクは、生成された列を移行またはレプリケートしません。

生成された列を移行する方法については、「」を参照してください???。

タイムアウト値が MariaDB ソースに適しているかどうかを検証する

API キー: `mariadb-check-source-network-parameter`

この移行前評価では、タスクの MariaDB ソースエンドポイントに `net_wait_timeout`、`net_read_timeoutwait_timeout`設定が少なくとも 300 秒に設定されているかどうかをチェックします。これは、移行中の切断を防ぐために必要です。

詳細については、「ターゲット MySQL のインスタンスへの接続は、タスクの実行中に接続が切断されます」を参照してください。

タイムアウト値が MariaDB ターゲットに適しているかどうかを検証する

API キー: `mariadb-check-target-network-parameter`

この移行前評価では、タスクの MariaDB ターゲットエンドポイントに `net_wait_timeout`、`net_read_timeoutwait_timeout`設定が少なくとも 300 秒に設定されているかどうかをチェックします。これは、移行中の切断を防ぐために必要です。

詳細については、「ターゲット MySQL のインスタンスへの接続は、タスクの実行中に接続が切断されます」を参照してください。

PostgreSQL 評価

このセクションでは、PostgreSQL ソースエンドポイントを使用する移行タスクの個々の移行前評価について説明します。

トピック

- 移行のためにソースデータベースのバージョンが DMS でサポートされているかどうかを検証する
- ソースデータベースで logical_decoding_work_memパラメータを検証する
- ソースデータベースに長時間実行されるトランザクションがあるかどうかを検証する
- ソースデータベースパラメータを検証する max_slot_wal_keep_size
- ソースデータベースパラメータpostgres-check-maxwalsendersが CDC をサポートするように設定されているかどうかを確認します。

- ソースデータベースが に設定されているかどうかを確認します。 PGLOGICAL
- ソーステーブルのプライマリキーが LOB データ型であるかどうかを検証する
- ソーステーブルにプライマリキーがあるかどうかを検証する
- 準備されたトランザクションがソースデータベースに存在するかどうかを検証する
- DMS CDC をサポートするために が最低限必要な値wal_sender_timeoutに設定されているかどうかを検証する
- ソースデータベースで wal_levelが論理 に設定されているかどうかを検証する

移行のためにソースデータベースのバージョンが DMS でサポートされているかどうかを検証する

API キー: `postgres-check-dbversion`

この移行前評価では、ソースデータベースのバージョンが と互換性があるかどうかを検証します AWS DMS。

ソースデータベースで **`logical_decoding_work_mem`**パラメータを検証する

API キー: `postgres-check-for-logical-decoding-work-mem`

この移行前評価では、ソースデータベースで `logical_decoding_work_mem`パラメータを調整することをお勧めします。トランザクションが長時間実行されているか、サブトランザクションが多数ある可能性のあるトランザクションの多いデータベースでは、論理デコードメモリの消費量が増加し、ディスクにスピルする必要がある可能性があります。これにより、レプリケーション中の DMS ソースのレイテンシーが大きくなります。このようなシナリオでは、 の調整が必要になる場合がありま`logical_decoding_work_mem`す。このパラメータは PostgreSQL バージョン 13 以降でサポートされています。

ソースデータベースに長時間実行されるトランザクションがあるかどうかを検証する

API キー: `postgres-check-longrunningtxn`

この移行前評価では、ソースデータベースで実行時間が 10 分以上続いたトランザクションがあるかどうかを検証します。デフォルトでは、DMS はタスクの開始中に開いているトランザクションをチェックするため、タスクの開始が失敗することがあります。

ソースデータベースパラメータを検証する **`max_slot_wal_keep_size`**

API キー: `postgres-check-maxslot-wal-keep-size`

この移行前評価では、 に設定された値を検証しま
すmax_slot_wal_keep_size。max_slot_wal_keep_size をデフォルト以外の値に設定すると、必要な WAL ファイルの削除により DMS タスクが失敗することがあります。

ソースデータベースパラメータ**postgres-check-maxwalsenders**が CDC をサポートするように設定されているかどうかを確認します。

API キー: postgres-check-maxwalsenders

この移行前評価では、ソースデータベースmax_wal_sendersで に設定された値を検証します。変更データキャプチャ (CDC) をサポートするmax_wal_sendersには、DMS を 1 より大きく設定する必要があります。

ソースデータベースが に設定されているかどうかを確認します。 **PGLOGICAL**

API キー: postgres-check-pglogical

この移行前評価では、CDC をサポートするpglogicalようにshared_preload_libraries値がに設定されているかどうかを検証PGLOGICALします。論理レプリケーションにテストデコードを使用する予定がある場合は、この評価を無視できることに注意してください。

ソーステーブルのプライマリキーが LOB データ型であるかどうかを検証する

API キー: postgres-check-pk-lob

この移行前評価では、テーブルのプライマリキーがラージオブジェクト (LOB) データ型であるかどうかを検証します。ソーステーブルにプライマリキーとして LOB 列がある場合、DMS はレプリケーションをサポートしません。

ソーステーブルにプライマリキーがあるかどうかを検証する

API キー: postgres-check-pk

この移行前評価では、タスクスコープで使用されるテーブルにプライマリキーが存在するかどうかを確認します。DMS は、レプリカ ID がソーステーブルfullで に設定されていない限り、プライマリキーのないテーブルのレプリケーションをサポートしていません。

準備されたトランザクションがソースデータベースに存在するかどうかを検証する

API キー: postgres-check-preparedtxn

この移行前評価では、ソースデータベースに準備済みのトランザクションが存在するかどうかを確認します。ソースデータベースに準備済みのトランザクションがある場合、レプリケーションスロットの作成が応答しなくなる可能性があります。

DMS CDC をサポートするために が最低限必要な値**wal_sender_timeout**に設定されているかどうかを検証する

API キー: `postgres-check-walsenderstimeout`

この移行前評価では、 `wal_sender_timeout`が 10000 ミリ秒 (10 秒) 以上に設定されているかどうかを確認します。CDC を使用する DMS タスクには、最低 10000 ミリ秒 (10 秒) が必要で、値が 10000 未満の場合、失敗します。

ソースデータベースで **wal_level**が論理 に設定されているかどうかを検証する

API キー: `postgres-check-wallevel`

この移行前評価では、 `wal_level`が論理的に設定されているかどうかを確認します。DMS CDC が機能するには、ソースデータベースでこのパラメータを有効にする必要があります。

データ型評価の開始と表示 (レガシー)

> **Note**
>
> このセクションでは、レガシーコンテンツについて説明します。前述した移行前評価の実行を使用することをお勧めします移行前評価の実行の指定、スタート、および表示。
> データ型評価はコンソールでは使用できません。API または CLI を使用してのみデータ型評価を実行でき、タスクの S3 バケットでデータ型評価の結果のみを表示できます。

データ型評価は、ターゲットがデータ型をサポートしていないため、正しく移行されない可能性があるソースデータベース内のデータ型を識別します。この評価中に、 は移行タスクのソースデータベーススキーマを AWS DMS 読み取り、列データ型のリストを作成します。次に、このリストを、でサポートされているデータ型の事前定義されたリストと比較します AWS DMS。移行タスクにサポートされていないデータ型がある場合、 AWS DMS はレポートを作成し、移行タスクにサポートされていないデータ型があるかどうかを確認できます。移行タスクにサポートされていないデータ型がない場合、レポートは作成 AWS DMS されません。

AWS DMS では、次のリレーショナルデータベースのデータ型評価レポートの作成がサポートされています。

- Oracle
- SQL Server
- PostgreSQL
- MySQL
- MariaDB
- Amazon Aurora

CLI と SDKs を使用して AWS DMS API にアクセスし、データ型評価レポートを開始および表示できます。

- CLI では、start-replication-task-assessment コマンドを使用してデータ型評価をスタートし、describe-replication-task-assessment-results コマンドを使用して、最新データ型評価レポートを JSON 形式で表示します。
- AWS DMS API は StartReplicationTaskAssessmentオペレーションを使用してデータ型評価を開始し、 DescribeReplicationTaskAssessmentResultsオペレーションを使用して最新のデータ型評価レポートを JSON 形式で表示します。

データ型評価レポートには、サポートされていないデータ型と、各データ型の列数を一覧表示する概要が含まれます。これには、サポートされていないデータ型のスキーマ、テーブル、列など、サポートされていない各データ型のデータ構造のリストが含まれます。このレポートを使用してソースデータ型を変更し、移行プロセスを向上させることができます。

サポートされていないデータ型のレベルは 2 つあります。レポートに「サポートされていない」と表示されたデータ型を移行することはできません。レポートに[partially supported](部分的にサポートされている) と表示されたデータ型は別のデータ型に変換できる場合もありますが、期待どおり移行されない可能性があります。

次の例に、表示する可能性のあるデータ型評価レポートの見本を示します。

```
{
    "summary":{
        "task-name":"test15",
        "not-supported":{
            "data-type": [
                "sql-variant"
            ],
            "column-count":3
```

```
        },
        "partially-supported":{
            "data-type":[
                "float8",
                "jsonb"
            ],
            "column-count":2
        }
    },
    "types":[
        {
            "data-type":"float8",
            "support-level":"partially-supported",
            "schemas":[
                {
                    "schema-name":"schema1",
                    "tables":[
                        {
                            "table-name":"table1",
                            "columns":[
                                "column1",
                                "column2"
                            ]
                        },
                        {
                            "table-name":"table2",
                            "columns":[
                                "column3",
                                "column4"
                            ]
                        }
                    ]
                },
                {
                    "schema-name":"schema2",
                    "tables":[
                        {
                            "table-name":"table3",
                            "columns":[
                                "column5",
                                "column6"
                            ]
                        },
                        {
```

```
                            "table-name":"table4",
                            "columns":[
                                "column7",
                                "column8"
                            ]
                        }
                    ]
                }
            ]
        },
        {
            "datatype":"int8",
            "support-level":"partially-supported",
            "schemas":[
                {
                    "schema-name":"schema1",
                    "tables":[
                        {
                            "table-name":"table1",
                            "columns":[
                                "column9",
                                "column10"
                            ]
                        },
                        {
                            "table-name":"table2",
                            "columns":[
                                "column11",
                                "column12"
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}
```

AWS DMS は、最新および以前のすべてのデータ型評価を、 によってアカウントで作成された Amazon S3 バケット AWS DMS に保存します。Amazon S3 バケット名は次の形式になります。*[CustomerId]* (カスタマーID) はお客様の ID で、*[customerDNS]* (カスタマーDNS) は内部識別子です。

```
dms-customerId-customerDNS
```

> **Note**
>
> デフォルトでは、 AWS アカウントにつき最大で 100 個の Amazon S3 バケットを作成できます。 AWS DMS はアカウントにバケットを作成するため、バケットの制限を超えないようにしてください。そうしないと、データ型の評価は失敗します。

特定の移行タスクのデータ型評価レポートはすべて、タスク識別子を持つ名前のバケット フォルダに保存されます。各レポートのファイル名は、 yyyy-mm-dd-hh-mm 形式のデータ型評価の日付です。Amazon S3 マネジメントコンソールから、以前のデータ型タスク評価レポートを表示し比較できます。

AWS DMS は、これらのレポート用に作成された S3 バケットへのアクセスを許可する AWS Identity and Access Management (IAM) ロールも作成します。ロール名は dms-access-for-tasks です。このロールでは AmazonDMSRedshiftS3Role ポリシーが使用されます。の実行時にResourceNotFoundFaultエラーが発生した場合はStartReplicationTaskAssessment、「トラブルシューティング」セクションResourceNotFoundFaultの「」を参照して、dms-access-for-tasksロールを手動で作成する方法について説明します。

評価実行のトラブルシューティング

次に、 で評価レポートを実行する際の問題のトラブルシューティングに関するトピックを示します AWS Database Migration Service。これらのトピックは、一般的な問題の解決に役立ちます。

トピック

- ResourceNotFoundFault 実行時 StartReplicationTaskAssessment

ResourceNotFoundFault 実行時 StartReplicationTaskAssessment

StartReplicationTaskAssessment アクションを実行するときに、次の例外が発生することがあります。

```
An error occurred (ResourceNotFoundFault) when calling the
 StartReplicationTaskAssessment operation: Task assessment has not been run or dms-
access-for-tasks IAM Role not configured correctly
```

この例外が発生した場合は、次の手順を実行してdms-access-for-tasksロールを作成します。

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. [ロールの作成] を選択します。
4. [信頼されたエンティティを選択] ページで、[信頼されたエンティティタイプ] に を選択します。
5. エディタに次の JSON を貼り付け、既存のテキストを置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "Service": "dms.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

   前述のポリシーは、 にアクセスsts:AssumeRole許可を付与します AWS DMS。AmazonDMSRedshiftS3Role ポリシーを追加すると、DMS はアカウントに S3 バケットを作成し、データ型評価結果をこの S3 バケットに入れることができます。
6. [次へ] をクリックします。
7. アクセス許可の追加ページで、AmazonDMSRedshiftS3Role ポリシーを検索して追加します。[次へ] をクリックします。
8. 名前、レビュー、作成ページで、ロールに という名前を付けますdms-access-for-tasks。[Create role] (ロールの作成) を選択します。

タスク設定の補足データを指定する

一部の AWS DMS エンドポイントのレプリケーションタスクを作成または変更する場合、そのタスクで移行を実行するための追加情報が必要になる場合があります。DMS コンソールのオプションを使用して、この追加情報を指定できます。または、DMS API オペレーションの

CreateReplicationTask または ModifyReplicationTask の TaskData パラメータを使用して指定することもできます 。

ターゲット エンドポイントが Amazon Neptune の場合、テーブルマッピングを補足するマッピングデータを指定する必要があります。この補足マッピングデータは、ソースのリレーショナルデータを Neptune データベースが使用できるターゲット グラフデータに変換する方法を指定します。この場合、2 つの形式のいずれかを使用できます。詳細については、「Amazon Neptune の Gremlin と R2RML をターゲットとして使用するグラフマッピングルールの指定」を参照してください。

AWS DMS タスクのモニタリング

モニタリングは、AWS DMS と 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うえで重要な部分です。マルチポイント障害が発生した場合のデバッグが容易になるように、AWS ソリューションのすべての部分からモニタリングデータを収集する必要があります。AWS は、AWS DMS タスクとリソースをモニタリングして、潜在的なインシデントに対応するためのツールをいくつか提供しています。

AWS DMS イベントと通知

AWS DMS は Amazon Simple Notification Service (Amazon SNS) を使用して、レプリケーションインスタンスが作成された際や削除された際など、AWS DMS イベントが発生すると、通知を送信します。AWS DMS はイベントをサブスクライブできるカテゴリにグループ化するため、そのカテゴリのイベントが発生した際に通知を受け取ることができます。例えば、特定のレプリケーションインスタンスの作成カテゴリをサブスクライブすると、レプリケーションインスタンスに影響を及ぼす作成に関連したイベントが発生する都度、通知が届きます。このような通知は、E メールメッセージ、テキストメッセージ、HTTP エンドポイントへの呼び出しなど、AWS リージョンの Amazon SNS でサポートされている任意の形式で使用できます。詳細については、「AWS Database Migration Service での Amazon SNS イベントと通知の使用」を参照してください。

タスクのステータス

タスクのステータスを確認したり、タスクのコントロールテーブルをモニタリングしたりして、タスクの進行状況をモニタリングできます。タスクのステータスは、AWS DMS タスクとタスクに関連するリソースのステータスを提供します。これには、タスクが作成中、開始中、実行中、停止中であることの表示があります。またテーブルのフルロードが開始されたか進行中かなど、タスクが移行中のテーブルの現在のステータス、テーブルに対して発生した挿入、削除、更新の数などの詳細も提供されます。タスクとタスクのリソースのステータスのモニタリングの詳細については、「タスクのステータス」と「タスク実行中のテーブルの状態」を参照してください。コントロールテーブルの詳細については、「制御テーブルタスク設定」を参照してください。

Amazon CloudWatch アラームとログ

Amazon CloudWatch アラームを使用すると、指定した期間にわたって 単一または複数のタスクメトリクスをモニタリングできます。メトリクスが指定されたしきい値を超えると、Amazon SNS トピックに通知が送信されます。CloudWatch アラームは特定の状態にあるという理由ではアクションを呼び出しません。むしろ、状態は変化し、指定された期間にわたりその状態が維持される必要があります。AWS DMS は、移行プロセス中に CloudWatch を使用してタスク

情報もログに記録します。タスクログに関する情報は、AWS CLI または AWS DMS API を使用して確認できます。AWS DMS で CloudWatch を使用する方法の詳細については、「Amazon CloudWatch を使用したレプリケーションタスクのモニタリング」を参照してください。AWS DMS メトリクスのモニタリングの詳細については、「AWS Database Migration Service のメトリクス」を参照してください。AWS DMS タスクログの使用の詳細については、「AWS DMS タスクログの表示と管理」を参照してください。

Time Travel ログ

レプリケーションタスクのログを記録してデバッグするには、AWS DMS Time Travel を使用できます。この方法の場合、Amazon S3 を使用してログを保存し、暗号化キーを使用して暗号化します。日時フィルターを使用して S3 ログを取得し、必要に応じてログを表示、ダウンロード、難読化できます。これにより、「時間を遡って」データベースのアクティビティを調査できます。

Time Travel は、DMS がサポートする PostgreSQL ソースエンドポイントと、DMS がサポートする PostgreSQL と MySQL のターゲットエンドポイントで使用できます。Time Travel は、フルロードタスクと CDC タスクと、CDC のみのタスクでのみ有効にできます。Time Travel を有効にしたり、既存の Time Travel 設定を変更するには、タスクが停止していることを確認する必要があります。

Time Travel ログの詳細については「Time Travel タスクの設定」を参照してください。Time Travel ログの使用のベスト プラクティスについては、「Time Travel を使用したレプリケーションタスクのトラブルシューティング」を参照してください。

AWS CloudTrail ログ

AWS DMS は、ユーザー、IAM ロール、または AWS DMS の AWS サービスが実行したアクションの記録を提供するサービスである AWS CloudTrail と統合されています。CloudTrail は、AWS DMS コンソールからのコールや AWS DMS API オペレーションへのコード呼び出しなどの AWS DMS のすべての API コールをイベントとして取得します。証跡を作成すると、AWS DMS のイベントを含む CloudTrail イベントの Amazon S3 ケットへの継続的配信を有効にできます。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確認できます。CloudTrail が収集した情報を使用して、AWS DMS 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。詳細については、「AWS CloudTrail を使用した AWS DMS API コールのログ記録」を参照してください。

データベースのログ

AWS Management Console、AWS CLI、または AWS データベースサービスの API を使用して、タスクエンドポイントのデータベースログを表示、ダウンロード、モニタリングできます。詳細

については、「AWS ドキュメント」でデータベースサービスのドキュメントを参照してください。

詳細については、次のトピックを参照してください。

トピック

- タスクのステータス
- タスク実行中のテーブルの状態
- Amazon CloudWatch を使用したレプリケーションタスクのモニタリング
- AWS Database Migration Service のメトリクス
- AWS DMS タスクログの表示と管理
- AWS CloudTrail を使用した AWS DMS API コールのログ記録
- AWS DMS コンテキストのログ記録

タスクのステータス

タスクのステータスはタスクの状態を提供します。タスクでありえるステータスは次の表のとおりです。

タスクのステータス	説明
作成中	AWS DMS がタスクを作成している。
実行中	タスクが指定された移行処理を実行中。
停止	タスクは停止された。
停止中	タスクは停止中。これは通常、タスクへのユーザーによる介入を示す。
削除中	タスクは削除中であり、通常、ユーザーによる介入のリクエストによるもの。
失敗	タスクが失敗した。詳細については、タスクのログファイルを参照。

タスクのステータス	説明
エラー	エラーが原因でジョブは停止した。タスクのエラーの簡潔な説明は、[概要] タブの最後のエラーメッセージセクションに記載されている。
エラーを伴って実行中	タスクがエラーステータスで実行されている。これは通常、タスク内の単一または複数のテーブルを移行できなかったことを示す。タスクは選択ルールに従ってその他のテーブルのロードを続ける。
起動中	タスクはレプリケーションインスタンスとソースエンドポイント、ターゲットエンドポイントに接続している。フィルターと変換が適用されている。
準備完了	タスクを実行する準備ができている。通常、このステータスは「作成中」ステータスの後に続くステータス。
変更中	タスクが変更されている。タスク設定は通常、ユーザーアクションにより変更される。
移動中	タスクは別のレプリケーションインスタンスに移動中。移動が完了するまで、レプリケーションはこの状態のままとなる。レプリケーションタスクの移動中に許可される唯一のオペレーションはタスクの削除。
Failed-move	ターゲットレプリケーションインスタンスに十分なストレージ領域がないなど、何らかの理由でタスクの移動が失敗した。レプリケーションタスクがこの状態の場合、タスクを開始、変更、移動、または削除できる。
テスト中	このタスクに指定されたデータベース移行は、StartReplicationTaskAssessmentRun オペレーションまたは StartReplicationTaskAssessment オペレーションの実行に応じてテストされている。

タスクのステータスバーには、タスクの推定進行状況が表示されます。この推定の正確性は、ソースデータベースのテーブル統計の品質によって異なります。テーブルの統計が優れているほど、推定も

正確になります。予測される列の統計がないテーブルが 1 つのみのタスクでは、いかなる完了率の推定も提供されることはありません。この場合、タスクの状態とロードされた行の表示を使用して、タスクが実際に実行されて進行中であることを確認できます。

DMS コンソールの「最終更新」列は、AWS DMS がテーブルのテーブル統計レコードを最後に更新した時刻のみを示すことに注意します。これは、テーブルの最終更新時刻を示すものではありません。

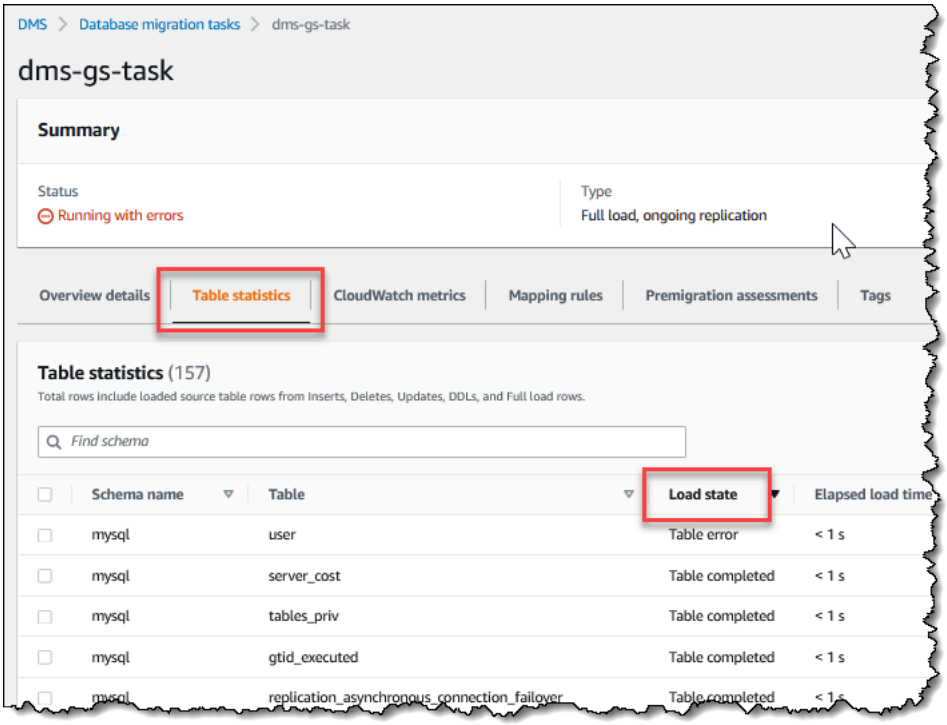
DMS コンソールを使用する以外にも、次の例のとおり、AWS CLI で `aws dms describe-replication-tasks` コマンドを使用して、タスクのステータスを含む現在のレプリケーションタスクの説明の出力を取得できます。

```
{
    "ReplicationTasks": [
        {
            "ReplicationTaskIdentifier": "moveit2",
            "SourceEndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA",
            "TargetEndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:EOM4SFKCZEYHZBFGAGZT3QEC5U",
            "ReplicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:T3OM7OUB5NM2LCVZF7JPGJRNUE",
            "MigrationType": "full-load",
            "TableMappings": ...output omitted... ,
            "ReplicationTaskSettings": ...output omitted... ,
            "Status": "stopped",
            "StopReason": "Stop Reason FULL_LOAD_ONLY_FINISHED",
            "ReplicationTaskCreationDate": 1590524772.505,
            "ReplicationTaskStartDate": 1590619805.212,
            "ReplicationTaskArn": "arn:aws:dms:us-
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII",
            "ReplicationTaskStats": {
                "FullLoadProgressPercent": 100,
                "ElapsedTimeMillis": 0,
                "TablesLoaded": 0,
                "TablesLoading": 0,
                "TablesQueued": 0,
                "TablesErrored": 0,
                "FreshStartDate": 1590619811.528,
                "StartDate": 1590619811.528,
                "StopDate": 1590619842.068
            }
        }
```

```
    ]
}
```

タスク実行中のテーブルの状態

AWS DMS コンソールは移行中にテーブルの状態に関する情報を更新します。表示される可能性のある状態値は、次の表のとおりです。



状態	説明
テーブルが存在しません	AWS DMS はソースエンドポイントでテーブルを見つけることができない。
Before load	フルロードプロセスは有効になっているが、まだ開始されていない。
フルロード	フルロードプロセスが進行中。
Table completed	フルロードは完了した。
Table cancelled	テーブルのロードがキャンセルされた。

状態	説明
テーブルエラー	テーブルのロード時にエラーが発生した。

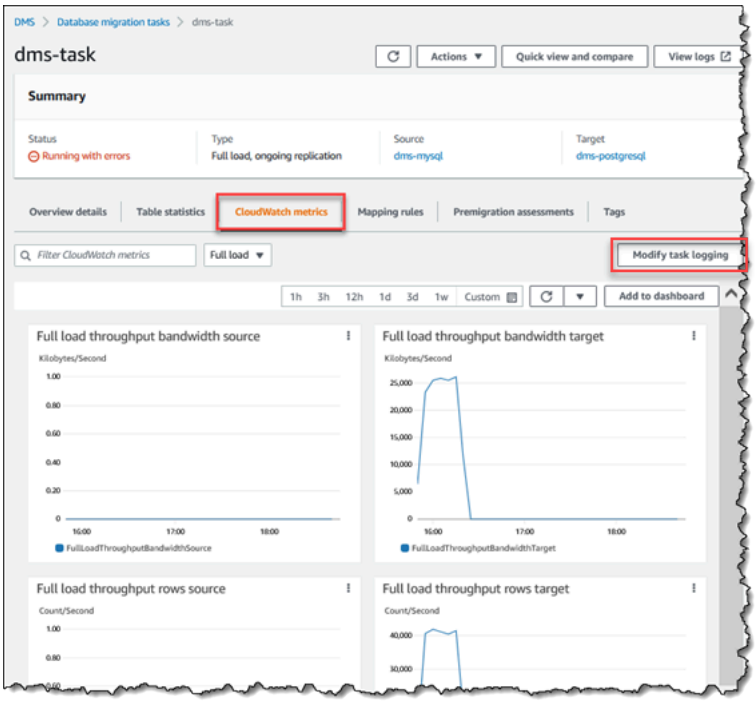
Amazon CloudWatch を使用したレプリケーションタスクのモニタリング

Amazon CloudWatch のアラームやイベントを使用すると、移行をより詳細に追跡できます。Amazon CloudWatch の詳細については、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch とは」、「Amazon CloudWatch Events」、「Amazon CloudWatch Logs とは」を参照してください。Amazon CloudWatch の使用には料金が発生することに注意します。

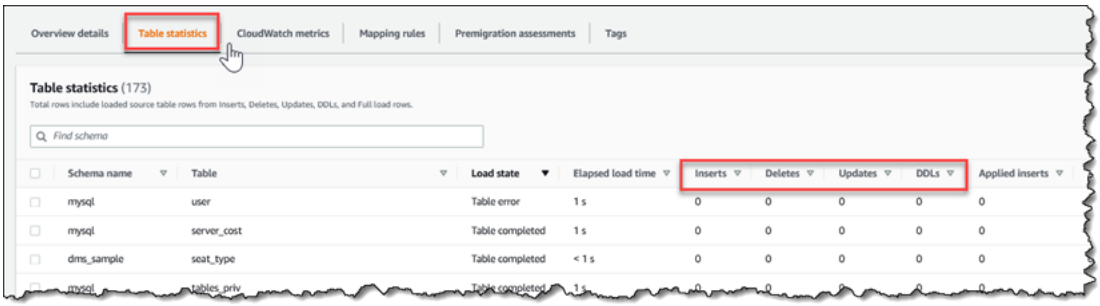
レプリケーションタスクで CloudWatch ログが作成されない場合は、「トラブルシューティングガイド」の「AWS DMS は CloudWatch ログを作成しない」を参照してください。

AWS DMS コンソールには、次のとおり、タスクのステータス、完了%、経過時間、テーブル統計など、各タスクの基本的な CloudWatch 統計が表示されます。レプリケーションタスクを選択して、[CloudWatch メトリクス] タブをクリックします。

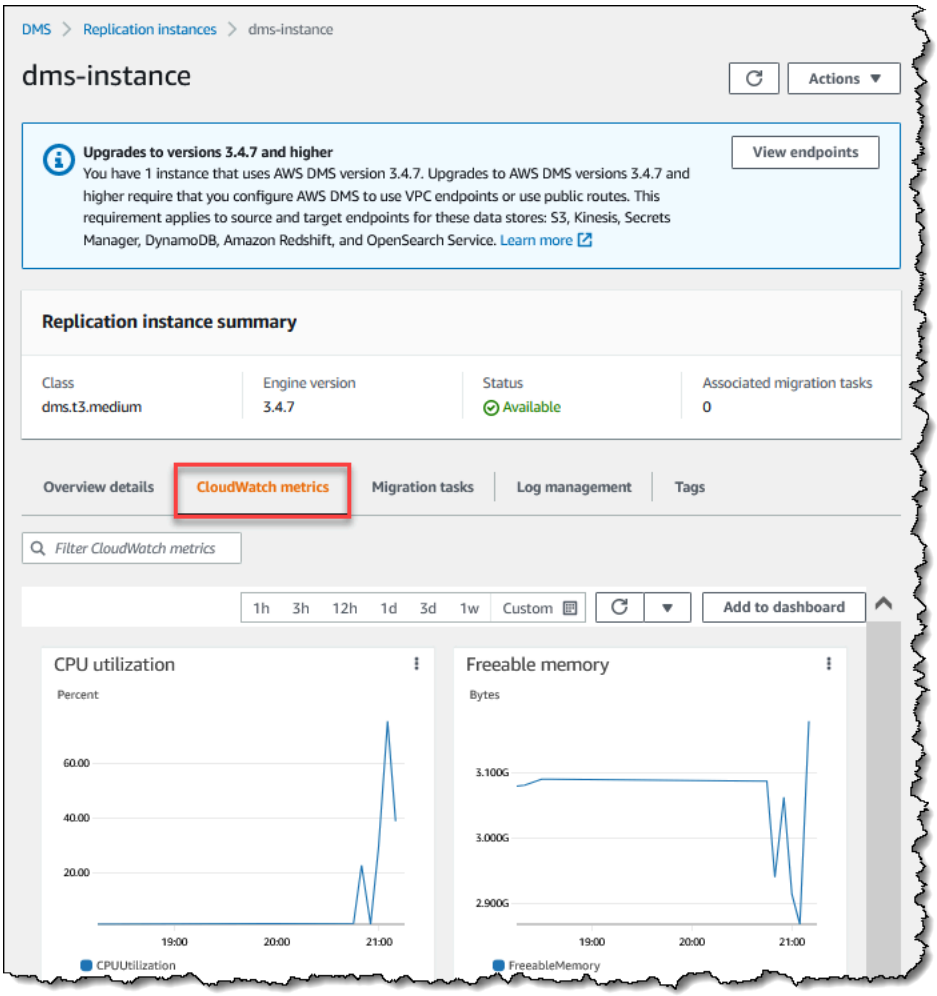
CloudWatchタスクログ設定を確認して変更するには、[タスクログ記録の変更] を選択します。詳細については、「ロギングタスク設定」を参照してください。



AWS DMS コンソールでは、[テーブル統計] タブをクリックすると、挿入、削除、更新の数など、各テーブルのパフォーマンス統計が表示されます。



さらに、[レプリケーションインスタンス] ページでレプリケーションインスタンスを選択して、[CloudWatch メトリクス] タブをクリックすると、インスタンスのパフォーマンスメトリクスを確認できます。



AWS Database Migration Service のメトリクス

AWS DMS は、次の統計を提供します。

- ホストのメトリクス – Amazon CloudWatch が提供するレプリケーションホストのパフォーマンスと使用率の統計。使用できるメトリクスの完全なリストについては、「レプリケーションインスタンスのメトリクス」を参照してください。
- レプリケーションタスクメトリクス – 受信した変更とコミットした変更、ソースデータベースとターゲットデータベースの両方のレプリケーションホストとの間の待機時間などのレプリケーションタスクの統計。使用できるメトリクスの完全なリストについては、「レプリケーションのタスクメトリクス」を参照してください。
- テーブルメトリクス – 完了した挿入、更新、削除、と DDL ステートメントの数などの移行処理中のテーブルの統計。

タスクメトリクスは、レプリケーションホストとソースエンドポイント間の統計と、レプリケーションホストとターゲットエンドポイント間の統計に分割されています。関連する 2 つの統計を組み合わせると、タスクの合計統計を導き出せます。例えば、CDCLatencySource 値と CDCLatencyTarget 値を組み合わせると、タスクの合計レイテンシー、つまりレプリカの遅延を調べることができます。

タスクメトリクス値は、ソースデータベースの現在のアクティビティの影響を受ける可能性があります。例えば、トランザクションが開始してもコミットされていない場合、CDCLatencySource メトリクスはトランザクションがコミットされるまで増大し続けます。

レプリケーションインスタンスの場合、FreeableMemory メトリクスには説明が必要です。解放可能なメモリは、実際に利用できる空きメモリを示すものではありません。これは、解放してその他の用途に使用できる現在使用中のメモリであり、レプリケーションインスタンスで使用されているバッファとキャッシュの組み合わせです。

[FreeableMemory] メトリクスは、実際に利用できる空きメモリを反映していないとはいえ、[FreeableMemory] メトリクスと [SwapUsage] メトリクスを組み合わせると、レプリケーションインスタンスが過負荷になっているかが明らかになります。

この 2 つのメトリクスは、次の条件を基にモニタリングします。

- FreeableMemory メトリクスが 0 に近づいているか。
- SwapUsage メトリクスが増加したり変動しているか。

上記の 2 つの状態のいずれかが認められた場合は、よりサイズが大きいレプリケーションインスタンスへの移行を検討する必要がある可能性があります。レプリケーションインスタンスで実行されるタスクの数とタイプを減らすことも検討する必要があります。フルロードタスクでは、変更をレプリケートするのみのタスクよりも多くのメモリを必要とします。

AWS DMS 移行タスクの実際のメモリ要件を見積もるには、次のパラメータを使用します。

LOB 列

移行範囲の各テーブルの LOB 列の平均数

並列にロードするテーブルの最大数

AWS DMS が単一のタスクで並列にロードするテーブルの最大数。

デフォルト値は 8 です。

LOB チャンクのサイズ

AWS DMS がデータをターゲットデータベースにレプリケートするために使用する LOB チャンクのキロバイト単位のサイズ

フルロード時のレートの確定

AWS DMS が並列に転送できるレコードの最大数。

デフォルト値は 10,000 です。

LOB のサイズ

個々の LOB のキロバイト単位の最大サイズ

一括の配列のサイズ

エンドポイントドライバーがフェッチまたは処理する行の最大数。この値は、ドライバーの設定によって異なります。

デフォルト値は 1,000 です。

上記の値を決定したら、次のいずれかの方法を使用して、移行タスクに必要なメモリ量を見積もることができます。これらの方法は、移行タスクの [LOB 列設定] で選択したオプションによって異なります。

- [完全 LOB モード] の場合は、次の式を使用します。

```
Required memory = (LOB columns) * (Maximum number of tables to load in
parallel) * (LOB chunk size) * (Commit rate during full load)
```

ソーステーブルに平均 2 つの LOB 列があり、LOB チャンクのサイズが 64 KB である例を想定します。`Maximum number of tables to load in parallel` と `Commit rate during full load` のデフォルト値を使用する場合、タスクに必要なメモリの量は、次のとおりです。

```
Required memory = 2 * 8 * 64 * 10,000 = 10,240,000 KB
```

> **Note**
>
> フルロード時の [フルロード時のレートの確定] 値を下げるには、AWS DMS コンソールを開いて、[データベース移行タスク] を選択し、タスクを作成するか、変更します。[詳細設定] を展開して、[フルロード時のレートの確定] 値を入力します。

- [制限付き LOB モード] の場合は、次の式を使用します。

  ```
  Required memory = (LOB columns) * (Maximum number of tables to load in
  parallel) * (LOB size) * (Bulk array size)
  ```

  ソーステーブルに平均 2 つの LOB 列があり、個々の LOB の最大サイズが 4,096 KB である例を想定します。`Maximum number of tables to load in parallel` と `Bulk array size` のデフォルト値を使用する場合、タスクに必要なメモリの量は、次のとおりです。

  ```
  Required memory = 2 * 8 * 4,096 * 1,000 = 65,536,000 KB
  ```

AWS DMS が変換を最適に実行するには、変換の発生時に CPU が利用できる必要があります。CPU に過負荷がかかり、十分な CPU リソースがないと、移行が遅くなる可能性があります。AWS DMS では、特に Oracle から PostgreSQL へなどの異種の移行やレプリケーションを実行する場合、CPU の負荷が増大する場合があります。このような状況の場合、C4 レプリケーションインスタンスクラスの使用が適切な選択となります。詳細については、「移行に適した AWS DMS レプリケーションインスタンスの選択」を参照してください。

レプリケーションインスタンスのメトリクス

レプリケーションインスタンスのモニタリングには、次の統計に関する Amazon CloudWatch メトリクスがあります。

メトリクス	説明
AvailableMemory	スワップなしで新しいアプリケーションを起動するために使用できるメモリ量の推定値。詳細については、「Linux man-pages」の「/proc/memInfo 」セクションの「MemAvailable 」値を参照。 単位: バイト
CPUAllocated	タスクに最大に割り当てられた CPU の割合 (%、0 は無制限を意味する)。 AWS DMS は、CloudWatch コンソールの ReplicationInstanceIdentifer と ReplicationTaskIdentifier を使用する。このメトリクスを表示するには、ReplicationInstanceIdentifier, ReplicationTaskIdentifier カテゴリを使用する。 単位: 割合 (%)
CPUUtilization	インスタンスで現在使用されている、割り当てられた vCPU (仮想 CPU) の割合 単位: 割合 (%)
DiskQueueDepth	ディスクへのアクセスを待機している未処理の読み取りまたは書き込み (I/O) のリクエストの数 単位: カウント
FreeStorageSpace	利用できるストレージ領域の容量 単位: バイト
FreeMemory	アプリケーション、ページ キャッシュ、カーネル独自のデータ構造で使用できる物理メモリの量。詳細については、「Linux man-pages」の「/proc/memInfo 」セクションの「MemFree」値を参照。 単位: バイト
FreeableMemory	利用できるランダムアクセスメモリの量

メトリクス	説明
	単位: バイト
MemoryAllocated	タスクに対するメモリの最大割り当て (0 は制限がないことを意味する)。 AWS DMS は、CloudWatch コンソールの ReplicationInstanceIdentifer と ReplicationTaskIdentifier を使用する。このメトリクスを表示するには、ReplicationInstanceIdentifier, ReplicationTaskIdentifier カテゴリを使用する。 単位: MiB
WriteIOPS	1 秒あたりのディスク書き込み I/O オペレーションの平均数 単位: カウント/秒
ReadIOPS	1 秒あたりのディスク読み取り I/O オペレーションの平均数 単位: カウント/秒
WriteThroughput	1 秒あたりにディスクに書き込まれる平均バイト数 単位: バイト/秒
ReadThroughput	1 秒あたりにディスクから読み込まれる平均バイト数 単位: バイト/秒
WriteLatency	ディスク I/O (出力) オペレーションごとにかかる平均時間 単位: ミリ秒
ReadLatency	ディスク I/O (入力) オペレーションごとにかかる平均時間 単位: ミリ秒
SwapUsage	レプリケーションインスタンスで使用されるスワップ領域の量 単位: バイト

メトリクス	説明
NetworkTransmitThroughput	レプリケーションインスタンス上の送信ネットワークトラフィック。これには、モニタリングとレプリケーションに使用されるお客様のデータベーストラフィックと AWS DMS トラフィックの両方が含まれる。 単位: バイト/秒
NetworkReceiveThroughput	レプリケーションインスタンス上の受信ネットワークトラフィック。これには、モニタリングとレプリケーションに使用されるお客様のデータベーストラフィックと AWS DMS トラフィックの両方が含まれる。 単位: バイト/秒

レプリケーションのタスクメトリクス

レプリケーションタスクのモニタリングには、次の統計のメトリクスがあります。

メトリクス	説明
FullLoadThroughputBandwidthTarget	ターゲットのフルロードで送信される送信データ量 (KB/秒)
FullLoadThroughputRowsTarget	ターゲットのフルロードでの発信変更数 (1 秒あたりの行数)
CDCIncomingChanges	ターゲットへの適用を待機している、特定の時点での変更イベントの合計数。これは、ソースエンドポイントのトランザクション変化率の測定値と同じではないことに注意する。このメトリクスの数値が高い場合は、通常、キャプチャした変更を AWS DMS がタイムリーに適用できず、ターゲットのレイテンシーが高くなっていることを示す。
CDCChangesMemorySource	メモリ内に蓄積され、ソースからのコミットを待機している行の量。このメトリクスは CDCChangesDiskSource と同時に確認する。

メトリクス	説明
CDCChangesMemoryTarget	メモリ内に蓄積され、ターゲットへのコミットを待機している行の量。このメトリクスは CDCChangesDiskTarget と同時に確認する。
CDCChangesDiskSource	ディスクに蓄積され、ソースからのコミットを待機している行の量。このメトリクスは CDCChangesMemorySource と同時に確認する。
CDCChangesDiskTarget	ディスクに蓄積され、ターゲットへのコミットを待機している行の量。このメトリクスは CDCChangesMemoryTarget と同時に確認する。
CDCThroughputBandwidthTarget	ターゲットに送信される送信データ量 (KB/秒) CDCThroughputBandwidth は、サンプリングポイントで送信された送信データを記録する。タスクのネットワークトラフィックが見つからない場合、値はゼロになる。CDC が長時間実行トランザクションを発行しないと、ネットワークトラフィックが記録されない場合がある。
CDCThroughputRowsSource	ソースから受信したタスクの変更数 (行数/秒)
CDCThroughputRowsTarget	ターゲットに送信されたタスクの変更数 (行数/秒)

メトリクス	説明
CDCLatencySource	ソースエンドポイントからキャプチャした最後のイベントと AWS DMS インスタンスの現在のシステムタイムスタンプの間のギャップ (秒単位)。CDCLatencySource は、ソースとレプリケーションインスタンス間のレイテンシーを示す。CDCLatencySource が高いと、ソースからの変更をキャプチャするプロセスが遅延していることを意味する。継続的なレプリケーションのレイテンシーを特定するには、このメトリクスを CDCLatencyTarget とともに確認する。CDCLatencySource と CDCLatencyTarget の両方が高い場合は、まず CDCLatencySource を調べる。 ソースとレプリケーションインスタンスの間にレプリケーションの遅延がない場合、CDCSourceLatency は 0 になることがある。レプリケーションタスクがソースのトランザクションログの次のイベントを読み取ろうとし、最後にソースから読み取った時と比較して新しいイベントがない場合にも、CDCSourceLatency がゼロになる可能性がある。このような状況の場合、タスクは CDCSourceLatency を 0 にリセットする。

メトリクス	説明
CDCLatencyTarget	ターゲットでのコミットを待機している最初のイベントのタイムスタンプと AWS DMS インスタンスの現在のタイムスタンプとの間のギャップ (秒単位)。ターゲットのレイテンシーは、レプリケーションインスタンスのサーバー時間と、ターゲットコンポーネントに転送された最も古い未確定のイベント ID との差。つまり、ターゲットのレイテンシーは、レプリケーションインスタンスと、適用されても TRG エンドポイントがまだ確認していない最も古いイベントとの間のタイムスタンプの差 (99%)。CDCLatencyTarget が高い場合は、変更イベントをターゲットに適用するプロセスの遅延を示す。継続的なレプリケーションのレイテンシーを特定するには、このメトリクスを CDCLatencySource と同時に確認する。CDCLatencyTarget が高くても CDCLatencySource が高くない場合は、次の点を調べる。 • ターゲットにプライマリキーまたはインデックスがない • ターゲットインスタンスまたはレプリケーションインスタンスでリソースのボトルネックが発生している • レプリケーションインスタンスとターゲットの間でネットワークの問題がある
CPUUtilization	複数のコアにわたるタスクによって使用されている CPU の割合。タスク CPUUtilization のセマンティクスは、レプリケーション CPUUtilizaiton とは少し異なる。単一の vCPU がフルで使用されている場合は 100% を示す。ただし、複数の vCPU が使用されている場合は、値が 100% を超える可能性がある。 単位: 割合 (%)
SwapUsage	タスクで使用されるスワップの量 単位: バイト

メトリクス	説明
MemoryUsage	タスクが使用した制御グループ (cgroup) の memory.usage_in_bytes。DMS は cgroup を使用して、メモリや CPU などのシステムリソースの使用を制御する。このメトリクスは、タスクに割り当てられた cgroup 内のタスクのメモリ使用量をメガバイト単位で示す。cgroup の制限は、DMS レプリケーションインスタンスクラスで利用できるリソースに基づいている。memory.usage_in_bytes は、メモリの常駐セット サイズ (RSS)、キャッシュ、スワップコンポーネントで構成される。オペレーティングシステムは、必要に応じてキャッシュメモリを再利用できる。レプリケーションインスタンスメトリクス Available Memory もモニタリングすることが推奨される。 AWS DMS は、CloudWatch コンソールの `ReplicationInstanceIdentifer` と `ReplicationTaskIdentifier` を使用する。このメトリクスを表示するには、`ReplicationInstanceIdentifier, ReplicationTaskIdentifier` カテゴリを使用する。

AWS DMS タスクログの表示と管理

Amazon CloudWatch を使用して、AWS DMS 移行プロセス中にタスク情報をログに記録できます。タスク設定を選択すると、ログ記録が有効になります。詳細については、「ロギングタスク設定」を参照してください。

実行されたタスクのログを確認するには、次のステップを実行します。

1. AWS DMS コンソールを開き、ナビゲーションペインで [データベース移行タスク] を選択します。データベース移行タスクのダイアログが開きます。
2. タスク名を選択します。概要の詳細ダイアログが表示されます。
3. [移行タスクのログ] セクションに移動して、[CloudWatch ログを表示] を選択します。

また、タスクログに関する情報は、AWS CLI または AWS DMS API を使用して確認できます。このためには、`describe-replication-instance-task-logs` AWS CLI コマンドまたは AWS DMS API アクション `DescribeReplicationInstanceTaskLogs` を使用します。

例えば、次の AWS CLI コマンドは、タスクログのメタデータを JSON 形式で表示します。

```
$ aws dms describe-replication-instance-task-logs \
   --replication-instance-arn arn:aws:dms:us-east-1:237565436:rep:CDSFSFSFFFSSUFCAY
```

コマンドからの応答の例は次のとおりです。

```
{
    "ReplicationInstanceTaskLogs": [
        {
            "ReplicationTaskArn": "arn:aws:dms:us-
east-1:237565436:task:MY34U6Z4MSY52GRTIX3O4AY",
            "ReplicationTaskName": "mysql-to-ddb",
            "ReplicationInstanceTaskLogSize": 3726134
        }
    ],
    "ReplicationInstanceArn": "arn:aws:dms:us-east-1:237565436:rep:CDSFSFSFFFSSUFCAY"
}
```

この応答には、レプリケーションインスタンスに関連付けられた単一のタスクログ (mysql-to-ddb) があります。このログのサイズは、3,726,124 バイトです。

describe-replication-instance-task-logs が返す情報を使用して、タスクログに関する問題の診断とトラブルシューティングを行うことができます。例えば、タスクの詳細なデバッグログを有効にすると、タスクログはすぐに増大するため、レプリケーションインスタンスが使用できるストレージが消費され、インスタンスのステータスが storage-full に変わる可能性があります。タスクログを記述することにより、必要でなくなったタスクログを判断できます。これにより、不要なタスクログを削除して、ストレージ領域を解放できます。

タスクのタスクログを削除するには、タスク設定 DeleteTaskLogs を true に設定します。例えば、次の JSON コードは、例えば、次の JSON は、AWS CLI の modify-replication-task コマンドまたは AWS DMS API の ModifyReplicationTask アクションを使用して、タスクを変更する際にタスクログを削除します。

```
{
   "Logging": {
       "DeleteTaskLogs":true
```

```
    }
}
```

AWS CloudTrail を使用した AWS DMS API コールのログ記録

AWS DMS は、AWS DMS のユーザー、ロール、または AWS サービスによって実行されたアクションの記録を提供するサービスである AWS CloudTrail と統合されています。CloudTrail は、AWS DMS コンソールからのコールや AWS DMS API オペレーションへのコード呼び出しなどの AWS DMS のすべての API コールをイベントとして取得します。証跡を作成すると、AWS DMS のイベントを含む CloudTrail イベントの Amazon S3 ケットへの継続的配信を有効にできます。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確認できます。CloudTrail が収集した情報を使用して、AWS DMS 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。

CloudTrail の詳細については、「AWS CloudTrailユーザーガイド」を参照してください。

CloudTrail での AWS DMS の情報

アカウントを作成すると AWS アカウントで CloudTrail が有効になります。AWS DMS でアクティビティが発生すると、そのアクティビティは [イベント履歴] のその他の AWS サービスイベントとともに CloudTrail イベントに記録されます。AWS アカウントでの最近のイベントは、表示、検索、ダウンロードできます。詳細については、「CloudTrail イベント履歴でイベントを確認する」を参照してください。

AWS アカウント内のイベント (AWS DMS のイベントを含む) を継続的に記録するには、証跡を作成します。証跡を使用すると、CloudTrail はログファイルを Amazon S3 バケットに配信できるようになります。デフォルトでは、コンソールで証跡を作成すると、証跡はすべての AWS リージョンに適用されます。証跡は、AWS パーティション内のすべての AWS リージョンからのイベントを記録し、指定した Amazon S3 バケットにログ ファイルを配信します。また、CloudTrail ログに収集されたイベントデータをさらに分析して、これに基づいて動作するようにその他の AWS サービスを設定することもできます。詳細については、次を参照してください。

- 証跡の作成の概要
- CloudTrail でサポートされるサービスと統合
- CloudTrail の Amazon SNS 通知の設定
- CloudTrail ログファイルの複数の AWS リージョンからの受け取りと複数のアカウントから CloudTrail ログファイルを受け取る

すべての AWS DMS アクションは CloudTrail によってログに記録され、「AWS Database Migration Service API リファレンス」に文書化されています。例えば、CreateReplicationInstance、TestConnection、StartReplicationTask の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

すべてのイベントまたはログエントリには、リクエストの生成者に関する情報が含まれています。アイデンティティの情報は次の判断に役立ちます。

- リクエストがルートまたは IAM ユーザーの認証情報を使用して行われたか。
- 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一時的なセキュリティ認証情報を使用して行われたか。
- リクエストが別の AWS サービスによって行われたか。

詳細については、「CloudTrail userIdentity エレメント」 を参照してください。

AWS DMS ログファイルエントリの理解

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 ログファイルには単一または複数のログエントリが含まれています。イベントは、任意のソースからの単一のリクエストを示し、リクエストしたアクション、アクションの日時、リクエストのパラメータなどに関する情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けされたスタックトレースではないため、特定の順序で表示されるわけではありません。

RebootReplicationInstance アクションを表示する CloudTrail ログエントリの例は次のとおりです。

```
{
"eventVersion": "1.05",
"userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAIOSFODNN7EXAMPLE:johndoe",
    "arn": "arn:aws:sts::123456789012:assumed-role/admin/johndoe",
    "accountId": "123456789012",
    "accessKeyId": "ASIAYFI33SINADOJJEZW",
    "sessionContext": {
        "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2018-08-01T16:42:09Z"
```

```
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AKIAIOSFODNN7EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/admin",
                "accountId": "123456789012",
                "userName": "admin"
            }
        }
    },
    "eventTime": "2018-08-02T00:11:44Z",
    "eventSource": "dms.amazonaws.com",
    "eventName": "RebootReplicationInstance",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.198.64",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "forceFailover": false,
        "replicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:EX4MBJ2NMRDL3BMAYJOXUGYPUE"
    },
    "responseElements": {
        "replicationInstance": {
            "replicationInstanceIdentifier": "replication-instance-1",
            "replicationInstanceStatus": "rebooting",
            "allocatedStorage": 50,
            "replicationInstancePrivateIpAddresses": [
                "172.31.20.204"
            ],
            "instanceCreateTime": "Aug 1, 2018 11:56:21 PM",
            "autoMinorVersionUpgrade": true,
            "engineVersion": "2.4.3",
            "publiclyAccessible": true,
            "replicationInstanceClass": "dms.t2.medium",
            "availabilityZone": "us-east-1b",
            "kmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
f7bc0f8e-1a3a-4ace-9faa-e8494fa3921a",
            "replicationSubnetGroup": {
                "vpcId": "vpc-1f6a9c6a",
                "subnetGroupStatus": "Complete",
                "replicationSubnetGroupArn": "arn:aws:dms:us-
east-1:123456789012:subgrp:EDHRVRBAAAPONQAIYWP4NUW22M",
                "subnets": [
                    {
```

```
                    "subnetIdentifier": "subnet-cbfff283",
                    "subnetAvailabilityZone": {
                        "name": "us-east-1b"
                    },
                    "subnetStatus": "Active"
                },
                {
                    "subnetIdentifier": "subnet-d7c825e8",
                    "subnetAvailabilityZone": {
                        "name": "us-east-1e"
                    },
                    "subnetStatus": "Active"
                },
                {
                    "subnetIdentifier": "subnet-6746046b",
                    "subnetAvailabilityZone": {
                        "name": "us-east-1f"
                    },
                    "subnetStatus": "Active"
                },
                {
                    "subnetIdentifier": "subnet-bac383e0",
                    "subnetAvailabilityZone": {
                        "name": "us-east-1c"
                    },
                    "subnetStatus": "Active"
                },
                {
                    "subnetIdentifier": "subnet-42599426",
                    "subnetAvailabilityZone": {
                        "name": "us-east-1d"
                    },
                    "subnetStatus": "Active"
                },
                {
                    "subnetIdentifier": "subnet-da327bf6",
                    "subnetAvailabilityZone": {
                        "name": "us-east-1a"
                    },
                    "subnetStatus": "Active"
                }
            ],
            "replicationSubnetGroupIdentifier": "default-vpc-1f6a9c6a",
```

```
                "replicationSubnetGroupDescription": "default group created by console
 for vpc id vpc-1f6a9c6a"
            },
            "replicationInstanceEniId": "eni-0d6db8c7137cb9844",
            "vpcSecurityGroups": [
                {
                    "vpcSecurityGroupId": "sg-f839b688",
                    "status": "active"
                }
            ],
            "pendingModifiedValues": {},
            "replicationInstancePublicIpAddresses": [
                "18.211.48.119"
            ],
            "replicationInstancePublicIpAddress": "18.211.48.119",
            "preferredMaintenanceWindow": "fri:22:44-fri:23:14",
            "replicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:EX4MBJ2NMRDL3BMAYJOXUGYPUE",
            "replicationInstanceEniIds": [
                "eni-0d6db8c7137cb9844"
            ],
            "multiAZ": false,
            "replicationInstancePrivateIpAddress": "172.31.20.204",
            "patchingPrecedence": 0
        }
    },
    "requestID": "a3c83c11-95e8-11e8-9d08-4b8f2b45bfd5",
    "eventID": "b3c4adb1-e34b-4744-bdeb-35528062a541",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

AWS DMS コンテキストのログ記録

AWS DMS は、コンテキストのログ記録を使用して、実行中の移行に関する情報を提供します。コンテキストのログ記録がタスクの CloudWatch ログに書き込む情報は次のとおりです。

- ソースデータベースとターゲットデータベースへのタスクの接続に関する情報
- レプリケーションタスクの動作。このタスクログを使用して、レプリケーションの問題を診断できます。

- AWS DMS がソースデータベースとターゲットデータベースで実行する SQL ステートメント。これはデータを伴っていません。この SQL ログを使用して、予期しない移行の動作を診断できます。
- 各 CDC イベントのストリーム位置の詳細

コンテキストのログ記録は、AWS DMS バージョン 3.5.0 以降でのみ利用できます。

AWS DMS はデフォルトでコンテキストのログ記録を有効にします。コンテキストのログ記録を制御するには、`EnableLogContext` タスク設定を `true` または `false` に設定するか、コンソールでタスクを変更します。

AWS DMS は、コンテキストログの情報を 3 分ごとに CloudWatch ログのレプリケーションタスクに書き込みます。レプリケーションインスタンスにアプリケーションログのための十分なスペースがあることを確認します。タスクログの管理の詳細については、「AWS DMS タスクログの表示と管理」を参照してください。

トピック

オブジェクトタイプ

AWS DMS は、次のオブジェクトタイプについて CloudWatch でコンテキストのログ記録を生成します。

オブジェクトタイプ	説明
`TABLE_NAME`	このログエントリには、現在のタスクマッピング ルールの範囲内にあるテーブルに関する情報が含まれている。このようなエントリを使用して、移行中の特定の期間のテーブルイベントを調べることができる。
`SCHEMA_NAME`	このログエントリには、現在のタスクマッピング ルールで使用されるスキーマに関する情

オブジェクトタイプ	説明
	報が含まれている。これらのエントリを使用して、AWS DMS が移行中の特定の期間に使用しているスキーマを判断できる。
TRANSACTION_ID	このログエントリには、ソースデータベースからキャプチャされた各 DML/DDL 変更のトランザクション ID が含まれている。このようなエントリを使用して、特定のトランザクション中にどのような変更が発生したかを判断できる。
CONNECTION_ID	このエントリには接続 ID が含まれる。このようなエントリを使用して、AWS DMS が各移行ステップで使用する接続を決定できる。
STATEMENT	このエントリには、各移行変更をフェッチ、処理、適用するために使用される SQL コードが含まれている。
STREAM_POSITION	このエントリには、ソースデータベースの各移行アクションのトランザクションログファイル内の位置が含まれる。このエントリの形式は、ソースデータベースエンジンタイプによって異なる。このような情報を使用して、CDC のみのレプリケーションを設定する際に復旧チェックポイントの開始位置を決定することもできる。

ログ記録の例

このセクションでは、レプリケーションのモニタリングとレプリケーションの問題の診断に使用できるログレコードの例を説明します。

接続ログの例

このセクションでは、接続 ID を含むログのサンプルを記載しています。

```
2023-02-22T10:09:29 [SOURCE_CAPTURE  ]I:  Capture record 1 to internal
 queue from Source  {operation:START_REGULAR (43), connectionId:27598,
 streamPosition:0000124A/6800A778.NOW}  (streamcomponent.c:2920)

2023-02-22T10:12:30 [SOURCE_CAPTURE  ]I:  Capture record 0 to internal queue from
 Source  {operation:IDLE (51), connectionId:27598}  (streamcomponent.c:2920)

2023-02-22T11:25:27 [SOURCE_CAPTURE  ]I:  Capture record 0 to internal queue
 from Source  {operation:IDLE (51), columnName:region, connectionId:27598}
 (streamcomponent.c:2920)
```

タスク動作ログの例

このセクションでは、レプリケーションタスクログの動作に関するログのサンプルを記載しています。このような情報を使用して、IDLE ステータスのタスクなど、レプリケーションの問題を診断できます。

次の SOURCE_CAPTURE ログは、ソースデータベースのログファイルから読み取ることができるイベントがないことを示しており、ターゲットデータベースに適用する AWS DMS CDC コンポーネントから受信したイベントがないことを示す TARGET_APPLY レコードが含まれています。上記のイベントには、以前に適用したイベント関連のコンテキストの詳細も含まれています。

```
2023-02-22T11:23:24 [SOURCE_CAPTURE  ]I:  No Event fetched from wal log
 (postgres_endpoint_wal_engine.c:1369)
2023-02-22T11:24:29 [TARGET_APPLY    ]I:  No records received to load
 or apply on target , waiting for data from upstream. The last context
 is  {operation:INSERT (1), tableName:sales_11, schemaName:public,
 txnId:18662441, connectionId:17855, statement:INSERT INTO
 "public"."sales_11"("sales_no","dept_name","sale_amount","sale_date","region") values
 (?,?,?,?,?),
```

SQL ステートメントログの例

このセクションでは、ソースデータベースとターゲットデータベースで実行される SQL ステートメントに関するログサンプルを記載しています。ログに表示される SQL ステートメントには SQL ステートメントのみが表示され、データは表示されません。次の TARGET_APPLY ログは、ターゲット上で実行された INSERT ステートメントを示しています。

```
2023-02-22T11:26:07 [TARGET_APPLY    ]I:  Applied record 2193305 to
 target  {operation:INSERT (1), tableName:sales_111, schemaName:public,
```

```
txnId:18761543, connectionId:17855, statement:INSERT INTO
"public"."sales_111"("sales_no","dept_name","sale_amount","sale_date","region") values
(?,?,?,?,?),
```

制限

AWS DMS コンテキストのログ記録には、次の制限が適用されます。

- AWS DMS はすべてのエンドポイントタイプについて最小限のログを作成します。ただし、広範なエンジン固有のコンテキストログは次のエンドポイントタイプでのみ利用できます。このようなエンドポイントタイプを使用する場合は、コンテキストのログ記録を有効にすることをお勧めします。
  - MySQL
  - PostgreSQL
  - Oracle
  - Microsoft SQL Server
  - MongoDB と Amazon DocumentDB
  - Amazon S3

AWS Database Migration Service での Amazon EventBridge イベントと通知の使用

Amazon EventBridge を使用すると、AWS DMS イベント (レプリケーションインスタンスの作成や削除など) が発生した際に通知が提供されます。EventBridge はイベントを受信し、イベントルールで定義されたイベントの通知を送信します。通知は、AWS リージョンの Amazon EventBridge でサポートされている任意の形式で使用できます。Amazon EventBridge の詳細については、「Amazon EventBridge ユーザーガイド」の「Amazon EventBridge とは」を参照してください。

> **Note**
>
> Amazon EventBridge イベントの利用は、AWS DMS バージョン 3.4.5 以降でサポートされています。

EventBridge は、AWS DMS 環境の変化を示すイベントを受信し、ルールを適用してイベントを通知メカニズムに送信します。ルールは、イベントパターンと呼ばれるイベントの構造に基づいて、イベントと通知メカニズムを照合します。

AWS DMS は、イベントをグループ分けして、イベントルールを適用できます。これにより、そのカテゴリのイベントが発生した際に、通知を受け取ることができます。例えば、特定のレプリケーションインスタンスの作成カテゴリに EventBridge イベント ルールを適用するとします。これにより、レプリケーションインスタンスに影響を与える作成関連のイベントが発生する都度通知が届きます。レプリケーションインスタンスの設定変更カテゴリにルールを適用すると、レプリケーションインスタンスの設定が変更された際に通知が送信されます。AWS DMS が提供するイベントカテゴリのリストについては、次の AWS DMS イベントカテゴリとイベントメッセージを参照してください。

> **Note**
>
> events.amazonaws.com からのパブリッシュを許可するには、必ず Amazon SNS トピックのアクセスポリシーを更新します。詳細については、「Amazon EventBridge ユーザーガイド」の「Amazon EventBridge のリソースベースのポリシーを使用する」を参照してください 。
> イベントサブスクリプションを Amazon EventBridge に移動する方法の詳細については、「migrate active event subscriptions from DMS to Amazon EventBridge」を参照してください。

Amazon SNS でのテキストメッセージの使用の詳細については、「Amazon SNS を使用した SMS 通知の送信と受信」を参照してください。

AWS DMS のための Amazon EventBridge イベントルールの使用

Amazon EventBridge は、EventBridge イベントルールの作成時に指定したアドレスにイベント通知を送信します。複数の異なるルールを作成することもできます。例えば、すべてのイベント通知を受信する単一のルールと、本番環境の DMS リソースの重大なイベントのみを含む別のルールを作成できます。EventBridge でイベント通知を有効または無効にすることもできます。

AWS DMS イベントに対応する Amazon EventBridge ルールを作成するには

- 「Amazon EventBridge ユーザーガイド」の「イベントに反応する Amazon EventBridge ルールの作成」で説明されている手順を実行して、AWS DMS イベントのためのルールを次のとおり作成します。

  a. EventBridge がルールのイベントパターンに一致するイベントを受信した際に実行する通知アクションを指定します。イベントが一致すると、EventBridge はイベントを送信し、ルールで定義されたアクションを呼び出します。
  b. [サービスプロバイダー] で、AWS を選択します。
  c. [サービス名] では、[Database Migration Service (DMS)] を選択します。

これで、イベント通知を受け取ることができます。

次の JSON の例は、AWS DMS サービスのEventBridge イベントモデルを説明しています。

```
{
   "version":"0",
   "id":"11a11b11-222b-333a-44d4-01234a5b67890",
   "detail-type":"DMS Replication Task State Change",
   "source":"aws.dms",
   "account":"0123456789012",
   "time":"1970-01-01T00:00:00Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:dms:us-east-1:012345678901:task:AAAABBBB0CCCCDDDDEEEEE1FFFF2GGG3FFFFFF3"
   ],
   "detail":{
      "type":"REPLICATION_TASK",
```

```
      "category":"StateChange",
      "eventType":"REPLICATION_TASK_STARTED",
      "eventId":"DMS-EVENT-0069",
      "resourceLink":"https://console.aws.amazon.com/dms/v2/home?region=us-
east-1#taskDetails/taskName",
      "detailMessage":"Replication task started, with flag = fresh start"
   }
}
```

通知できるカテゴリとイベントのリストについては、次のセクションを参照してください。

AWS DMS のイベントカテゴリとイベントメッセージ

AWS DMS は、識別できるカテゴリ内で多数のイベントを生成します。各カテゴリは、レプリケーションインスタンスまたはレプリケーションタスクのソースタイプに適用されます。

トピック

- ReplicationInstance イベントメッセージ
- ReplicationTask イベントのメッセージ
- レプリケーションイベントのメッセージ

ReplicationInstance イベントメッセージ

ReplicationInstance ソースタイプのカテゴリとイベントは、次の表のとおりです。

カテゴリ	イベント ID	説明
作成	DMS-EVENT-0067	レプリケーションインスタンス作成中。
削除	DMS-EVENT-0066	レプリケーションインスタンス削除中。
設定変更	DMS-EVENT-0012	このレプリケーションインスタンスのレプリケーションインスタンスクラスが変更されている。

カテゴリ	イベント ID	説明
設定変更	DMS-EVENT-0018	レプリケーションインスタンスのストレージが増加している。
設定変更	DMS-EVENT-0024	レプリケーションインスタンスはマルチ AZ 配置に移行中。
設定変更	DMS-EVENT-0030	レプリケーションインスタンスはシングル AZ 配置に移行中。
メンテナンス	DMS-EVENT-0026	レプリケーションインスタンスのオフラインメンテナンス実行中。このレプリケーションインスタンスは現在使用できない。
作成	DMS-EVENT-0005	レプリケーションインスタンスが作成された。
削除	DMS-EVENT-0003	レプリケーションインスタンスが削除された。
設定変更	DMS-EVENT-0014	このレプリケーションインスタンスのレプリケーションインスタンスクラスが変更された。
設定変更	DMS-EVENT-0017	レプリケーションインスタンスのストレージが増加した。
設定変更	DMS-EVENT-0025	レプリケーションインスタンスのマルチ AZ 配置への移行が完了した。
設定変更	DMS-EVENT-0029	レプリケーションインスタンスのシングル AZ 配置への移行が完了した。
メンテナンス	DMS-EVENT-0047	レプリケーションインスタンスの管理ソフトウェアが更新された。

カテゴリ	イベント ID	説明
メンテナンス	DMS-EVENT-0027	レプリケーションインスタンスのオフラインメンテナンスが完了した。レプリケーションインスタンスは現在使用できる。
メンテナンス	DMS-EVENT-0068	レプリケーションインスタンスがアップグレードできない状態。
フェイルオーバー	DMS-EVENT-0034	フェイルオーバーのリクエスト頻度が多すぎると、通常のフェイルオーバーイベントの代わりにこのイベントが発生する。
失敗	DMS-EVENT-0031	レプリケーションインスタンスが %s 状態となった。
失敗	DMS-EVENT-0036	ネットワークに互換性がないため、レプリケーションインスタンスが失敗した。
失敗	DMS-EVENT-0037	サービスがデータボリュームの暗号化に使用される KMS キーにアクセスできない場合。
失敗		レプリケーションインスタンスが非互換パラメータに設定された。
フェイルオーバー		ユーザーのリクエストによるフェイルオーバーを開始するために安全な状態を待機している間にタイムアウトになった。
フェイルオーバー	DMS-EVENT-0013	マルチ AZ レプリケーションインスタンスのフェイルオーバーが開始された。

カテゴリ	イベント ID	説明
フェイルオーバー	DMS-EVENT-0049	マルチ AZ レプリケーションインスタンスのフェイルオーバーが完了した。
フェイルオーバー	DMS-EVENT-0050	マルチ AZ アクティベーションが開始された。
フェイルオーバー	DMS-EVENT-0051	マルチ AZ アクティベーションが完了した。
StateChange		一般クエリログとスロークエリログが自動的に %s としてローテーションされた。
StateChange		AWS DMS がアプリケーションインスタンス %s の KMS 暗号化キーにアクセスできない。キーが無効になっているか、AWS DMS がアクセスできないことが原因と考えられる。この状態が続くと、アプリケーションはアクセスできない状態になる。詳細については、AWS DMS ドキュメントのトラブルシューティングセクションを参照。
StateChange		AWS DMS は、アプリケーションインスタンス %s の KMS 暗号化キーに正常にアクセスできるようになった。
StateChange		Amazon DMS、アプリケーションインスタンス %s の KMS 暗号化キーにアクセスできなかった。このアプリケーションはアクセスできない状態になる。詳細については、Amazon DMS ドキュメントのトラブルシューティングセクションを参照。

カテゴリ	イベント ID	説明
StateChange		レプリケーションインスタンス作成の一環として HM でアプリが再起動する。
StateChange		レプリケーションインスタンス削除の一環としての HM でアプリケーションがシャットダウンする
フェイルオーバー	DMS-EVENT-0015	マルチ AZ のスタンドバイへのフェイルオーバーが完了した。
LowStorage	DMS-EVENT-0007	レプリケーションインスタンスの空きストレージが少なくなっている。
LowStorage		割り当てられた使用できる inode がなくなった。解決するには、ストレージをスケールする

ReplicationTask イベントのメッセージ

ReplicationTask ソースタイプのカテゴリとイベントは、次の表のとおりです。

カテゴリ	イベント ID	説明
失敗	DMS-EVENT-0078	レプリケーションタスクが失敗した。
失敗	DMS-EVENT-0082	タスクデータを消去する呼び出しが失敗した。
状態変化	DMS-EVENT-0081	テーブルの詳細の再ロードがリクエストされた。
状態変化		レプリケーションタスクがコピーされた。

カテゴリ	イベント ID	説明
状態変化		レプリケーションタスクのコピーが失敗した。
状態変化		レプリケーションタスクは移動された。
状態変化		レプリケーションタスクの移動が失敗した。
状態変化		ターゲットタスクの作成が失敗した。
状態変化		レプリケーションタスク評価の実行が開始された。
状態変化		レプリケーションタスク評価の実行が正常に完了した。
状態変化		レプリケーションタスク評価の実行が失敗した。
StateChange		レプリケーションタスク評価の実行が警告付きで完了した·
StateChange		レプリケーションタスク評価の実行がエラーで完了した。
StateChange		レプリケーションタスク評価の実行 %s がキャンセルされた。
StateChange		レプリケーションタスク評価の実行 %s が削除された。
StateChange		レプリケーションタスク評価の実行でリソースをプロビジョンできなかった。
StateChange		レプリケーションタスクが失敗した。

カテゴリ	イベント ID	説明
作成		レプリケーションタスクが作成された。
ConfigurationChange		レプリケーションタスクが変更された。
失敗		レプリケーションタスクが失敗した。
StateChange	DMS-EVENT-0091	読み込みが一時停止した。スワップファイル限界に達した。
StateChange	DMS-EVENT-0092	読み取りが一時停止した。ディスク使用量の限界に達した。
StateChange	DMS-EVENT-0093	読み取りが一時停止した。ディスク使用量の限界に達した。
StateChange	DMS-EVENT-0093	読み取りを再開した。
StateChange	DMS-EVENT-0069	レプリケーションタスクは taskType %s、startType %s で開始された
StateChange	DMS-EVENT-0079	レプリケーションが停止した。
削除	DMS-EVENT-0073	レプリケーションタスクが削除された。

レプリケーションイベントのメッセージ

Replication ソースタイプのカテゴリとイベントは、次の表のとおりです。

カテゴリ	説明
状態変化	DMS レプリケーションのスケールアップイベント

カテゴリ	説明
状態変化	DMS レプリケーションのスケールダウンイベント
状態変化	DMS レプリケーションのスケーリングイベントが完了した。
状態変化	DMS レプリケーションが作成された。
状態変化	DMS レプリケーションは初期化中。
状態変化	DMS レプリケーションは、メタデータ収集のためのリソースを準備している。
状態変化	DMS レプリケーションに関連する接続がテスト中。
状態変化	DMS レプリケーションはメタデータをフェッチしている。
状態変化	DMS レプリケーションはキャパシティを計算している。
状態変化	DMS レプリケーションはキャパシティをプロビジョンしている。
状態変化	DMS レプリケーションがプロビジョンされた。
状態変化	DMS レプリケーションが開始された。
状態変化	DMS レプリケーションが実行中。
状態変化	DMS レプリケーションは停止中。
状態変化	DMS レプリケーションが停止した。
状態変化	DMS レプリケーションは変更中。
状態変化	DMS レプリケーションは削除中。
状態変化	DMS レプリケーションはキャパシティのプロビジョニングを解除している。
状態変化	DMS レプリケーションはプロビジョニング解除された。
失敗	DMS レプリケーションが失敗した。

AWS Database Migration Service での Amazon SNS イベントと通知の使用

AWS DMS 3.4.5 リリース以降のバージョンでは、AWS DMS イベントが発生した際の通知には、Amazon EventBridge を使用することをお勧めします。AWS DMS での EventBridge イベントの使用の詳細については、「AWS Database Migration Service での Amazon EventBridge イベントと通知の使用」を参照してください。

イベントサブスクリプションの Amazon EventBridge への移動

次の AWS CLI コマンドを使用して、アクティブなイベントサブスクリプションを DMS から Amazon EventBridge に一度に最大 10 件移行できます。

```
update-subscriptions-to-event-bridge [--force-move | --no-force-move]
```

デフォルトでは、レプリケーションインスタンスが AWS DMS 3.4.5 以降を使用している場合にのみ、AWS DMS はアクティブなイベントサブスクリプションを移行します。このデフォルトの動作を上書きするには、`--force-move` オプションを使用します。ただし、レプリケーションインスタンスがアップグレードされていない場合、イベントのタイプによっては Amazon EventBridge を使用しては利用できない場合があります。

`update-subscriptions-to-event-bridge` CLI コマンドを実行する AWS Identity and Access Management (IAM) ユーザには、次のポリシーのアクセス許可が必要です。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "SNS:GetTopicAttributes",
              "SNS:SetTopicAttributes",
              "events:PutTargets",
              "events:EnableRule",
              "events:PutRule"
          ],
          "Resource": "*"
      }
  ]
```

```
}
```

EventBridge へのサブスクリプションの移動の詳細については、「AWS Database Migration Service API リファレンス」の「UpdateSubscriptionsToEventBridge」を参照してください。

Amazon SNS イベントと通知の使用

AWS DMS 3.4.5 以前のバージョンでサポートされているイベントと通知は、次のとおりです。

AWS Database Migration Service (AWS DMS) は、Amazon Simple Notification Service (Amazon SNS) を使用して、レプリケーションインスタンスの作成や削除などの AWS DMS イベントが発生した際に通知を送信できます。このような通知は、E メールメッセージ、テキストメッセージ、HTTP エンドポイントへの呼び出しなど、AWS リージョンの Amazon SNS でサポートされている任意の形式で使用できます。

AWS DMS は、ユーザーがイベントをサブスクライブできるカテゴリにイベントをグループ分けします。これにより、そのカテゴリのイベントが発生した際に、通知を受け取ることができます。例えば、特定のレプリケーションインスタンスの作成カテゴリをサブスクライブすると、レプリケーションインスタンスに影響を及ぼす作成に関連したイベントが発生する都度、通知が届きます。レプリケーションインスタンスの設定変更カテゴリにサブスクライブすると、レプリケーションインスタンスの設定が変更された際に通知が送信されます。イベント通知サブスクリプションが変更された場合にも通知が送信されます。AWS DMS が提供するイベントカテゴリのリストについては、次の「SNS 通知の AWS DMS イベントカテゴリとイベントメッセージ」を参照してください。

AWS DMS は、イベントサブスクリプションの作成時に指定したアドレスにイベント通知を送信します。すべてのイベント通知を受信する単一のサブスクリプションと、本番環境の DMS リソースの重大なイベントのみを含む別のサブスクリプションなど、複数の異なるサブスクリプションを作成することができます。AWS DMS コンソールで [有効]オプションをオフにするか、AWS DMS API を使用して `Enabled` パラメータを [false] に設定することで、サブスクリプションを削除せずに通知を容易に無効にできます。

> **Note**
>
> 現時点で、SMS テキストメッセージを使用した AWS DMS イベント通知は、Amazon SNS がサポートされているすべての AWS リージョンの AWS DMS リソースで利用できます。Amazon SNS が SMS メッセージングをサポートする AWS リージョンと国のリストについては、「サポートされている国と地域」を参照してください。

SNS でのテキストメッセージの使用の詳細については、「Amazon SNS を使用した SMS 通知の送信と受信」を参照してください。
AWS DMS イベント通知は、CloudWatch または EventBridge の CloudTrail イベントとは異なります。CloudTrail イベント通知は、任意の API 呼び出しにより生成できます。DMS の場合は、DMS イベントが発生した場合にのみ通知を送信します。

AWS DMS は、サブスクリプション識別子を使用して各サブスクリプションを識別します。複数の AWS DMS のイベントサブスクリプションを同じ Amazon SNS トピックに発行できます。イベント通知を使用すると Amazon SNS の料金が適用されます。Amazon SNS の請求詳細については「Amazon SNS の料金」を参照してください。

Amazon SNS を使用して AWS DMS イベントをサブスクライブするには、次のプロセスを使用します。

1. Amazon SNS トピックを作成します。トピックでは、受信する通知タイプと、通知の宛先となるアドレスまたは番号を指定します。
2. AWS Management Console、AWS CLI、または AWS DMS API を使用して、AWS DMS イベント通知サブスクリプションを作成します。
3. AWS DMS は、サブスクリプションを送信したアドレスに承認メールまたは SMS メッセージを送信します。サブスクリプションを確認するには、承認メールまたは SMS メッセージに記載のリンクをクリックします。
4. サブスクリプション確認後、AWS DMS コンソールの [イベントサブスクリプション] セクションで、サブスクリプションのステータスが更新されます。
5. その後、イベント通知の送信が開始します。

通知できるカテゴリとイベントのリストについては、次のセクションを参照してください。AWS DMS イベントサブスクリプションのサブスクライブと使用の詳細については、「SNS を使用した AWS DMS イベント通知のサブスクリプション」を参照してください。

SNS 通知の AWS DMS イベントカテゴリとイベントメッセージ

Important

AWS DMS 3.4.5 リリース以降のバージョンでは、AWS DMS イベントが発生した際の通知には、Amazon EventBridge を使用することをお勧めします。AWS DMS での EventBridge

イベントの使用の詳細については、「AWS Database Migration Service での Amazon EventBridge イベントと通知の使用」を参照してください。

AWS DMS は、AWS DMS コンソールまたは AWS DMS API を使用してサブスクライブできるカテゴリに多数のイベントを生成します。各カテゴリはソースタイプに適用されます。AWS DMS は現時点でレプリケーションインスタンスとレプリケーションタスクのソースタイプをサポートしています。

レプリケーションインスタンスのソースタイプのカテゴリとイベントは、次の表のとおりです。

カテゴリ	DMS イベント ID	説明
設定変更	DMS-EVENT-0012	このレプリケーションインスタンスのレプリケーションインスタンスクラスが変更されている。
設定変更	DMS-EVENT-0014	このレプリケーションインスタンスのレプリケーションインスタンスクラスが変更された。
設定変更	DMS-EVENT-0018	レプリケーションインスタンスのストレージが増加している。
設定変更	DMS-EVENT-0017	レプリケーションインスタンスのストレージが増加した。
設定変更	DMS-EVENT-0024	レプリケーションインスタンスはマルチ AZ 配置に移行中。
設定変更	DMS-EVENT-0025	レプリケーションインスタンスはマルチ AZ 配置への移行を完了した。
設定変更	DMS-EVENT-0030	レプリケーションインスタンスはシングル AZ 配置に移行中。
設定変更	DMS-EVENT-0029	レプリケーションインスタンスのシングル AZ 配置への移行が完了した。
作成	DMS-EVENT-0067	レプリケーションインスタンス作成中。
作成	DMS-EVENT-0005	レプリケーションインスタンスが作成された。

カテゴリ	DMS イベント ID	説明
削除	DMS-EVENT-0066	レプリケーションインスタンス削除中。
削除	DMS-EVENT-0003	レプリケーションインスタンスが削除された。
メンテナンス	DMS-EVENT-0047	レプリケーションインスタンスの管理ソフトウェアが更新された。
メンテナンス	DMS-EVENT-0026	レプリケーションインスタンスのオフラインメンテナンス実行中。このレプリケーションインスタンスは現在使用できない。
メンテナンス	DMS-EVENT-0027	レプリケーションインスタンスのオフラインメンテナンスが完了した。レプリケーションインスタンスは現在使用できる。
メンテナンス	DMS-EVENT-0068	レプリケーションインスタンスがアップグレードできない状態。
LowStorage	DMS-EVENT-0007	レプリケーションインスタンスは、割り当てられたストレージの 90% 以上を使用している。空きストレージ領域メトリクスを使用して、レプリケーションインスタンスのストレージ容量をモニタリングできる。
フェイルオーバー	DMS-EVENT-0013	マルチ AZ レプリケーションインスタンスのフェイルオーバーが開始された。
フェイルオーバー	DMS-EVENT-0049	マルチ AZ レプリケーションインスタンスのフェイルオーバーが完了した。
フェイルオーバー	DMS-EVENT-0015	スタンドバイへのマルチ AZ フェイルオーバーが完了した。
フェイルオーバー	DMS-EVENT-0050	マルチ AZ アクティベーションが開始された。

カテゴリ	DMS イベント ID	説明
フェイルオーバー	DMS-EVENT-0051	マルチ AZ アクティベーションが完了した。
フェイルオーバー	DMS-EVENT-0034	フェイルオーバーのリクエスト頻度が多すぎると、通常のフェイルオーバーイベントの代わりにこのイベントが発生する。
失敗	DMS-EVENT-0031	レプリケーションインスタンスでストレージ障害が発生した。
失敗	DMS-EVENT-0036	ネットワークに互換性がないため、レプリケーションインスタンスが失敗した。
失敗	DMS-EVENT-0037	サービスは、データ ボリュームの暗号化に使用される AWS KMS キーにアクセスできない。

レプリケーションタスクのソースタイプのカテゴリとイベントは、次の表のとおりです。

カテゴリ	DMS イベント ID	説明
状態変化	DMS-EVENT-0069	レプリケーションタスクが開始された。
状態変化	DMS-EVENT-0081	テーブルの詳細の再ロードがリクエストされた。
状態変化	DMS-EVENT-0079	レプリケーションタスクが停止された。
状態変化	DMS-EVENT-0091	読み込みが一時停止した。スワップファイル限界に達した。
状態変化	DMS-EVENT-0092	読み取りが一時停止した。ディスク使用量の限界に達した。
状態変化	DMS-EVENT-0093	読み取りを再開した。
失敗	DMS-EVENT-0078	レプリケーションタスクが失敗した。

カテゴリ	DMS イベント ID	説明
失敗	DMS-EVENT-0082	タスクを削除する呼び出しがタスクデータのクリーンアップに失敗した。
設定変更	DMS-EVENT-0080	レプリケーションタスクが変更された。
削除	DMS-EVENT-0073	レプリケーションタスクが削除された。
作成	DMS-EVENT-0074	レプリケーションタスクが作成された。

状態変更カテゴリの AWS DMS イベントサブスクリプションの例は次のとおりです。

```
        Resources:
            DMSEvent:
                Type: AWS::DMS::EventSubscription
                Properties:
                    Enabled: true
                    EventCategories: State Change
                    SnsTopicArn: arn:aws:sns:us-east-1:123456789:testSNS
                    SourceIds: []
                    SourceType: replication-task
```

SNS を使用した AWS DMS イベント通知のサブスクリプション

> **Important**
>
> AWS DMS 3.4.5 リリース以降のバージョンでは、AWS DMS イベントが発生した際の通知には、Amazon EventBridge を使用することをお勧めします。AWS DMS での EventBridge イベントの使用の詳細については、「AWS Database Migration Service での Amazon EventBridge イベントと通知の使用」を参照してください。

AWS DMS イベント通知サブスクリプションを作成すると、AWS DMS イベントが発生した際に通知を受け取ることができます。サブスクリプションを作成する最も簡単な方法は、AWS DMS コンソールを使用することです。通知サブスクリプションでは、通知を送信する方法と送信先を選択します。通知を受け取るソースタイプを指定します。現時点で、AWS DMS はレプリケーションインス

タンスとレプリケーションタスクのソースタイプをサポートしています。選択したソースタイプに応じてイベントカテゴリを選択して、イベント通知を受け取るソースを特定します。

AWS Management Console を使用する場合

> **Important**
>
> AWS DMS 3.4.5 リリース以降のバージョンでは、AWS DMS イベントが発生した際の通知には、Amazon EventBridge を使用することをお勧めします。AWS DMS での EventBridge イベントの使用の詳細については、「AWS Database Migration Service での Amazon EventBridge イベントと通知の使用」を参照してください。

コンソールを使用して Amazon SNS で AWS DMS イベント通知をサブスクライブするには

1. AWS Management Console にサインインして、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。

   IAM ユーザーとしてサインインしている場合は、AWS DMS にアクセスするための適切なアクセス許可があることを確認します。
2. ナビゲーションペインで [イベントサブスクリプション] を選択します。
3. [イベントサブスクリプション] ページで、[イベントサブスクリプションの作成] をクリックします。
4. [イベントサブスクリプションの作成] ページで、次のとおり設定します。

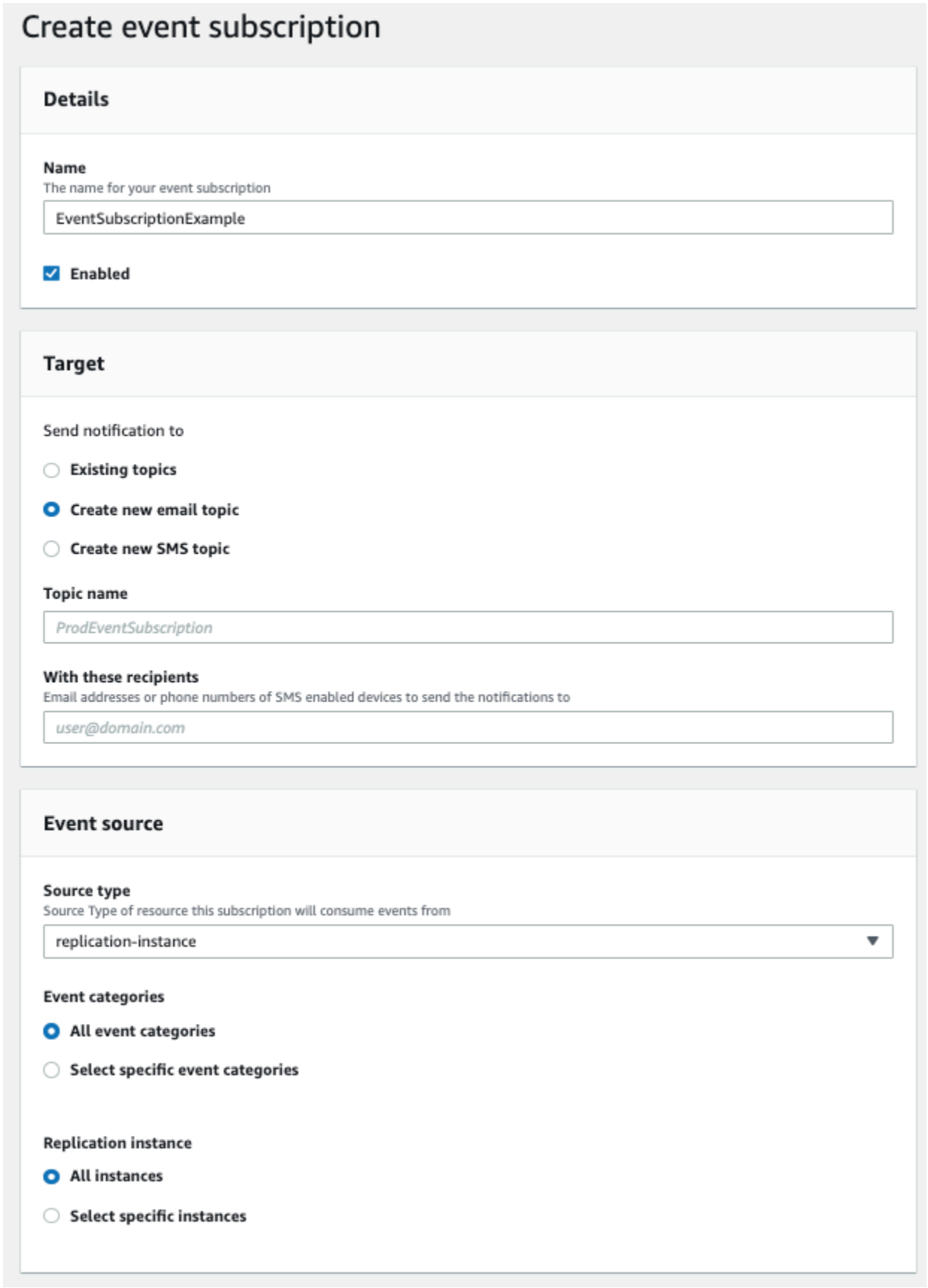
   a. [詳細] の下にある [名前] には、イベント通知サブスクリプション名を入力します。

   b. サブスクリプションを有効にするには、[有効] を選択します。サブスクリプションを作成しても、まだ通知を送信しない場合は、[有効] は選択しません。

   c. 通知を送信するには、[ターゲット] の下で、[既存のトピック]、[新しいメールトピックの作成]、または [新しい SMS トピックの作成] を選択します。通知の送信先として、既存の Amazon SNSトピックを使用するか、トピックを作成する必要があります。トピックを作成する場合は、通知の送信先の E メールアドレスを入力します。

   d. [イベントソース] の下の [ソースタイプ] で、ソースタイプを選択します。オプションは、[replication-instance] と [replication-task] のみです。

   e. 選択したソースタイプに応じて、イベント通知を受信するイベントカテゴリとソースを選択します。


Create event subscription
Details
Name
The name for your event subscription
EventSubscriptionExample
Enabled
Target
Send notification to
Existing topics
Create new email topic
Create new SMS topic
Topic name
ProdEventSubscription
With these recipients
Email addresses or phone numbers of SMS enabled devices to send the notifications to
user@domain.com
Event source
Source type
Source Type of resource this subscription will consume events from
replication-instance
Event categories
All event categories
Select specific event categories
Replication instance
All instances
Select specific instances

f. [イベントサブスクリプションの作成] を選択します。

AWS DMS コンソールにサブスクリプションが作成中であることが表示されます。

> **Note**
>
> AWS DMS API と CLI を使用して、Amazon SNS イベント通知サブスクリプションを作成することもできます。詳細については、「AWS DMS API リファレンス」の「CreateEventSubscription」と「AWS DMS CLI リファレンス」ドキュメントの「create-event-subscription」を参照してください。

SNS トピックのアクセスポリシーの検証

SNS アクセスポリシーには、AWS DMS が SNS トピックにイベントを公開できるアクセス許可が必要です。次の手順の説明のとおり、アクセスポリシーを検証し、更新できます。

アクセスポリシーを検証するには

1. [Amazon SNS] コンソールを開きます。
2. ナビゲーションペインで [トピック] をクリックして、DMS 通知を受け取るトピックを選択します。
3. [アクセスポリシー] タブをクリックします。

SNS アクセスポリシーで AWS DMS が SNS トピックにイベントをパブリッシュすることを許可していない場合は、ポリシーを更新できます。

アクセスポリシーを更新するには

1. トピックページの [詳細] セクションで、[編集] を選択します。
2. [アクセスポリシー] セクションを展開して、次のポリシーを JSON エディタにアタッチします。

```
{
      "Sid": "dms-allow-publish",
      "Effect": "Allow",
      "Principal": {
        "Service": "dms.amazonaws.com"
```

```
      },
      "Action": "sns:Publish",
      "Resource": "your-SNS-topic-ARN"
    }
```

イベントをトピックにパブリッシュする DMS EventSubscription ARN である aws:SourceArn 条件を指定して、SNS トピックへのアクセスをさらに制限することをお勧めします。

```
...
"Resource": "your-SNS-topic-ARN"
"Condition": {
    "StringEquals": {
       "aws:SourceArn": "arn:partition:dms:your-AWS-region:your-AWS-account-
ID:es:your-dms-es-arn or *"
 }
```

3. [変更の保存] をクリックします。

AWS DMS データ検証

トピック

AWS DMS では、データがソースからターゲットに正確に移行されたことを確認するため、データを検証できます。有効な場合、テーブルに対して全ロードが実行された後で、検証がすぐに開始します。検証では、CDC が有効なタスクの増分変更が発生したときに比較されます。

データの検証中に AWS DMS はソースの各行をターゲットの対応する行と比較し、それらの行に同じデータが含まれていることを確認し、食い違いがあればレポートします。AWS DMS は、これを達成するため、適切なクエリを実行してデータを取得します。これらのクエリは、ソースとターゲットで追加リソースと、追加ネットワークリソースを消費します。

CDC のみのタスクで検証が有効になっている場合、新しいデータの検証を開始する前に、テーブル内のすべての既存データが検証されます。

データ検証は、次のソースデータベースで AWS DMS がソースエンドポイントとしてサポートしている場合は常に機能します。

- Oracle
- PostgreSQL 互換データベース (PostgreSQL または Aurora PostgreSQL、Aurora Serverless for PostgreSQL)
- MySQL 互換データベース (MySQL または MariaDB、Aurora MySQL、Aurora Serverless for MySQL)
- Microsoft SQL Server

- IBM Db2 LUW

データ検証は、次のターゲットデータベースで AWS DMS がターゲットエンドポイントとしてサポートしている場合は常に機能します。

- Oracle
- PostgreSQL 互換データベース (PostgreSQL または Aurora PostgreSQL、Aurora Serverless for PostgreSQL)
- MySQL 互換データベース (MySQL または MariaDB、Aurora MySQL、Aurora Serverless for MySQL)
- Microsoft SQL Server
- IBM Db2 LUW
- Amazon Redshift
- Amazon S3。Amazon S3 ターゲットデータの検証については、「Amazon S3 ターゲットのデータ検証」を参照してください。

サポートされているエンドポイントの詳細については、「AWS DMS エンドポイントの使用」をご参照ください。

データの検証には、移行自体に必要な時間以外にも、さらに時間がかかります。必要な追加の時間は、移行されたデータの量によって異なります。

これらの設定の詳細については、「 データ検証タスクの設定」をご参照ください。

JSON ファイル内の `ValidationSettings` タスク設定の例については、タスク設定例 を参照してください。

レプリケーションタスクの統計

データ検証が有効になっている場合、AWS DMS はテーブルレベルで以下の統計情報を提供します。

- [ValidationState] (検証状態) — テーブルの検証状態。このパラメータには以下の値があります。
  - Not enabled — 移行タスクでテーブルに対して検証が有効化されていません。
  - Pending records — テーブル内の一部のレコードが、検証を待機しています。

  - [Mismatched records] (不一致レコード) – テーブル内の一部のレコードが、ソースとターゲット間で一致しません。さまざまな理由により、不一致が発生することがあります。詳細については、ターゲットエンドポイントの `awsdms_control.awsdms_validation_failures_v1` を確認してください。
  - Suspended records – テーブル内に検証できないレコードがあります。
  - No primary key – テーブルにプライマリキーがないため、検証できません。
  - [Table error] (テーブルエラー)– テーブルがエラー状態で一部のデータが移行されなかったため、テーブルは検証されませんでした。
  - [Validated] (検証済み) – テーブル内のすべての行が検証されます。テーブルが更新された場合、ステータスは [Validated] から変わる可能性があります。
  - Error – 予期しないエラーが発生したため、テーブルを検証できません。
  - [Pending validation] (検証保留中) — テーブルは検証を待っています。
  - [Preparing table] (テーブルの準備) — 移行タスクで有効になっているテーブルの検証を準備します。
  - [Pending revalidation] (保留中の再検証) —テーブルが更新された後で、テーブル内のすべての行が検証を保留します。
- ValidationPending – ターゲットに移行されたが、まだ検証されていないレコードの数。
- ValidationSuspended — AWS DMS が比較することができないレコードの数 たとえば、ソースのレコードが頻繁に更新されている場合、AWS DMS は、ソースとターゲットを比較できません。
- [ValidationFailed] (検証失敗) – データの検証フェーズに合格しなかったレコードの数。

JSON ファイル内の `ValidationSettings` タスク設定の例については、タスク設定例 を参照してください。

データ検証情報を表示するには、コンソールまたは AWS CLI、AWS DMS API を使用できます。

- コンソールで、タスクを作成または変更するときにタスクの検証を選択できます。コンソールを使用してデータ検証レポートを表示するには、[Tasks] ページでタスクを選択し、詳細セクションの [Table statistics] タブを選択します。
- CLI を使用して、タスク作成時または変更時にデータ検証を開始する場合は、`EnableValidation` パラメータを `true` に設定します。以下の例では、タスクを作成し、データ検証を有効にします。

```
create-replication-task
  --replication-task-settings '{"ValidationSettings":{"EnableValidation":true}}'
```

```
  --replication-instance-arn arn:aws:dms:us-east-1:5731014:
     rep:36KWVMB7Q
  --source-endpoint-arn arn:aws:dms:us-east-1:5731014:
     endpoint:CSZAEFQURFYMM
  --target-endpoint-arn arn:aws:dms:us-east-1:5731014:
     endpoint:CGPP7MF6WT4JQ
  --migration-type full-load-and-cdc
  --table-mappings '{"rules": [{"rule-type": "selection", "rule-id": "1",
     "rule-name": "1", "object-locator": {"schema-name": "data_types", "table-name":
 "%"},
     "rule-action": "include"}]}'
```

describe-table-statistics コマンドを使用して、JSON 形式でデータ検証レポートを受け取ります。以下のコマンドでは、データ検証レポートを表示します。

```
aws dms  describe-table-statistics --replication-task-arn arn:aws:dms:us-
east-1:5731014:
rep:36KWVMB7Q
```

このレポートは以下の例のようになります。

```
{
    "ReplicationTaskArn": "arn:aws:dms:us-west-2:5731014:task:VFPFTYKK2RYSI",
    "TableStatistics": [
        {
            "ValidationPendingRecords": 2,
            "Inserts": 25,
            "ValidationState": "Pending records",
            "ValidationSuspendedRecords": 0,
            "LastUpdateTime": 1510181065.349,
            "FullLoadErrorRows": 0,
            "FullLoadCondtnlChkFailedRows": 0,
            "Ddls": 0,
            "TableName": "t_binary",
            "ValidationFailedRecords": 0,
            "Updates": 0,
            "FullLoadRows": 10,
            "TableState": "Table completed",
            "SchemaName": "d_types_s_sqlserver",
            "Deletes": 0
        }
}
```

- AWS DMS API を使用し、[CreateReplicationTask] (レプレリケーションタスク作成) アクションを使ってタスクを作成し、次に、`EnableValidation` パラメータを true に設定して、タスクによって移行されたデータを検証します。DescribeTableStatistics アクションを使用して、JSON 形式でデータ検証レポートを受け取ります。

Amazon CloudWatch を使用したレプリケーション タスクの統計

Amazon CloudWatch が有効な場合、AWS DMS には、次のレプリケーション タスクの統計情報が表示されます。

- ValidationSucceededRecordCount - AWS DMSが検証した 1 分あたりの行数。
- ValidationAttemptedRecordCount - 検証が試行された行の 1 分あたりの数。
- ValidationFailedOverallCount - 検証が失敗した行の数。
- ValidationSuspendedOverallCount - 検証が停止された行の数。
- ValidationPendingOverallCount - 検証がまだ保留中の行の数。
- ValidationBulkQuerySourceLatency — AWS DMS は、特に多くの変更がある場合、全ロードまたは継続的レプリケーション中の特定のシナリオで、データ検証を一括実行できます。このメトリックスは、ソースエンドポイントから大量のデータを読み取るために必要なレイテンシーを示します。
- ValidationBulkQueryTargetLatency — AWS DMS は、特に多くの変更がある場合、ロードまたは継続的レプリケーション中の特定のシナリオで、データ検証を一括実行できます。このメトリックスは、ターゲットエンドポイントの大量のデータを読み取るために必要なレイテンシーを示します。
- ValidationItemQuerySourceLatency - 継続的レプリケーションでは、データ検証によって継続的変更を識別し、それらの変更を検証できます。このメトリックスは、変更をソースから読み取る際のレイテンシーを示します。検証中にエラーが発生した場合、検証では必要な数以上のクエリを実行できます。
- ValidationItemQueryTargetLatency - 継続的レプリケーションでは、データ検証によって継続的変更を識別し、それらの変更を行ごとに検証できます。このメトリックスは、変更をターゲットから読み取る際のレイテンシーを示します。検証中にエラーが発生した場合、検証では必要な数以上のクエリが実行される場合があります。

CloudWatch が有効な統計情報からデータ検証情報を収集するには、タスクを作成または変更するとき、[Enable CloudWatch logs] (CloudWatch ログを有効にする) コンソールを使用します。次に、

データ検証情報を確認し、データがソースからターゲットに正確に移行されたことを確認するため、以下を行います。

1. [Database migration tasks] (データベース移行タスク) ページでタスクを選択します。
2. [CloudWatch metrics] (CloudWatch メトリクス) タブ。
3. CloudWatch が有効な統計情報からデータ検証情報を収集するには、タスクを作成または変更するとき、[Enable CloudWatch logs] (CloudWatch ログを有効にする) コンソールを使用します。

タスク実行中のテーブル再検証

タスクを実行中に、AWS DMS がデータ検証を実行するようにリクエストできます。

AWS Management Console

1. AWS Management Console にサインインし、AWS DMS コンソール (https://console.aws.amazon.com/dms/v2/) を開きます。

   AWS Identity and Access Management (IAM) ユーザーとしてサインインしている場合は、AWS DMS にアクセスするための適切なアクセス許可があることを確認します。必要な許可は AWS DMSの使用に必要な IAM アクセス許可 をご参照ください。
2. ナビゲーションペインから [Tasks] を選択します。
3. 再検証するテーブルを含む実行中のタスクを選択します。
4. [テーブル統計] タブを選択します。
5. 再検証するテーブルを選択します (一度に最大 10 個のテーブルを選択できます)。タスクがすでに実行されていない場合は、テーブルを再検証できません。
6. [Revalidate (再検証)] を選択します。

JSON エディタを使用して検証ルールを変更する

AWS DMS コンソールから JSON エディタを使用して検証ルールをタスクに追加するには、次の操作を行います：

1. [Database migration tasks] (データベース移行タスク) を選択します。
2. 移行タスクの一覧からタスクを選択します。

3. タスクが実行中の場合には、[Actions] (アクション) ドロップダウンメニューから[Stop] (停止) を選択します。
4. タスクが停止したら、タスクを変更するには、[Actions] (アクション) ドロップダウンメニューから[Modify] (変更) を選択します。
5. [Table mappings] (テーブルマッピング) セクションで、[JSON editor] (JSON エディター) を選択し、検証ルールをテーブルマッピングに追加します。

たとえば、次の検証ルールを追加して、ソースで置換関数を実行できます。この場合、検証ルールがヌルバイトを検出すると、それをスペースとして検証します。

```
{
 "rule-type": "validation",
 "rule-id": "1",
 "rule-name": "1",
 "rule-target": "column",
 "object-locator": {
  "schema-name": "Test-Schema",
  "table-name": "Test-Table",
  "column-name": "Test-Column"
 },
 "rule-action": "override-validation-function",
 "source-function": "REPLACE(${column-name}, chr(0), chr(32))",
 "target-function": "${column-name}"
}
```

検証のみのタスク

検証専用タスクを作成して、移行やデータレプリケーションを実行せずにデータをプレビューしたり検証したりできます。検証のみのタスクを作成するには、EnableValidation 設定と ValidationOnly 設定を true に設定します。ValidationOnly を有効にすると、追加の要件が適用されます。詳細については、「データ検証タスクの設定」を参照してください。

フルロードのみの移行タイプで多くの障害がレポートされた場合、検証のみのタスクでは CDC の同等のタスクよりもはるかに速く完了します ただし、ソースエンドポイントまたはターゲットエンドポイントへの変更は、フルロードモードの障害としてレポートされることが不利な点となる可能性があります。

CDC 検証のみのタスクは、平均レイテンシーに基づいて検証を遅らせ、障害を報告する前に複数回再試行します。データ比較の大部分が障害付きで完了する場合、CDC モードの検証のみタスクは非常に遅く、不利となる可能性があります。

検証のみのタスクは、特に CDC の場合、レプリケーションタスクと同じ方向に設定する必要があります。これは、CDC 検証のみのタスクが、ソースの変更ログに基づいて変更され、再検証が必要な行を検出するためです。ターゲットがソースとして指定されている場合、ターゲットは DMS がターゲットに送信した変更についてのみ認識するため、レプリケーションエラーが検出される保証はありません。

フルロード検証のみ

AWS DMS バージョン 3.4.6 以降では、フルロード検証のみのタスクは、1 回のパスでソーステーブルとターゲットテーブルのすべての行を迅速に比較し、障害があればすぐにレポートしてからシャットダウンします。このモードでは障害が発生しても検証が中断されることはなく、速度が最適化されます。ただし、ソースまたはターゲットエンドポイントへの変更は失敗として報告されます。

> **Note**
>
> この検証動作は AWS DMS バージョン 3.4.6 以降では、検証が有効になっているフルロード移行タスクにも適用されます。

CDC 検証のみ

CDC 検証のみのタスクでは、新たな開始時にソーステーブルとターゲットテーブル間の既存のすべての行を検証します。さらに、CDC 検証のみのタスクは継続的に実行され、継続的なレプリケーションの変更は再検証され、各パスで報告される失敗の数は制限され、不一致の行は失敗する前に再試行されます。誤検出を防ぐように最適化されています。

`FailureMaxCount` または `TableFailureMaxCount` のしきい値を超えると、テーブル (またはタスク全体) の検証が中断されます。これは、検証が有効になっている CDC またはフルロード + CDC 移行タスクにも適用されます。また、検証が有効になっている CDC タスクでは、ソースとターゲットの平均レイテンシーにより、変更された各行の再検証が遅延します。

ただし、CDC 検証のみのタスクではデータは移行されず、レイテンシーもありません。デフォルトでは、`ValidationQueryCdcDelaySeconds` は 180 に設定されます。レイテンシーが高い環境のアカウントではこの量を増やし、誤検出を防ぐこともできます。

検証のみのユースケース

移行またはレプリケーションタスクのデータ検証部分を別の検証のみのタスクに分割するユースケースには、次がありますが、これらに限定されません。

- 検証がいつ行われるかを正確に制御する — 検証クエリを使用すると、ソースエンドポイントとターゲットエンドポイントの両方に追加の負荷がかかります。そのため、最初に 1 つのタスクでデータを移行またはレプリケートして、次に別のタスクで結果を検証すると、有益な場合があります。
- レプリケーションインスタンスの負荷を軽減する — データ検証を分割して独自のインスタンスで実行すると、利点が得られる場合があります。
- 特定の時点で一致しない行の数を迅速に取得する — 例えば、ターゲットエンドポイントへのメンテナンスウィンドウの本番環境のカットオーバーの直前または最中に、フルロード検証のみのタスクを作成すると、質問への回答を得られます。
- CDC コンポーネントを使用した移行タスクで検証の失敗が予想される場合 — 例えば、Oracle の `varchar2` を PostgreSQL の `jsonb` に移行する場合、CDC 検証は、このような失敗した行の再試行を引き続き行い、毎回レポートされる障害の数を制限します。ただし、フルロード検証のみのタスクを作成すると、より迅速に回答を得ることができます。
- 検証に失敗したテーブルを読み取るデータ修復スクリプトまたはユーティリティを開発する — (トラブルシューティング も参照)。フルロード検証のみのタスクでは、データ修復スクリプトが対応できるように障害を迅速にレポートします。

JSON ファイル内の `ValidationSettings` タスク設定の例については、「タスク設定例」を参照してください)。

トラブルシューティング

検証中に、AWS DMS はターゲットエンドポイント `awsdms_control.awsdms_validation_failures_v1` に新しいテーブルを作成します。レコードが ValidationSuspended または ValidationFailed 状態になった場合、AWS DMS は診断情報を `awsdms_control.awsdms_validation_failures_v1` に書き込みます。このテーブルをクエリすることで、検証エラーをトラブルシューティングすることができます。

ターゲット上でテーブルが作成されるデフォルトスキーマの変更については、「制御テーブルタスク設定」をご参照ください。

以下に、`awsdms_control.awsdms_validation_failures_v1` テーブルの説明を示します。

列名	データ型	説明
TASK_NAME	VARCHAR(128) NOT NULL	AWS DMS タスク識別子。
TABLE_OWNER	VARCHAR(128) NOT NULL	テーブルのスキーマ (所有者)。
TABLE_NAME	VARCHAR(128) NOT NULL	テーブル名。
FAILURE_TIME	DATETIME(3) NOT NULL	イベントが失敗した時刻。
KEY_TYPE	VARCHAR(128) NOT NULL	将来の使用のために予約済み (値は常に [Row](行))
KEY	TEXT NOT NULL	これは行レコードタイプのプライマリキーです。
FAILURE_TYPE	VARCHAR(128) NOT NULL	検証エラーの重大度。RECORD_DIFF または MISSING_SOURCE 、 MISSING_TARGET のいずれかになります。
DETAILS	VARCHAR(8000) NOT NULL	所与のキーと一致しないすべてのソース/ターゲット列の値を含む JSON 形式の文字列です。

以下のクエリでは、awsdms_control.awsdms_validation_failures_v1 テーブルに対してクエリを実行して、タスクのすべての失敗を表示します。タスク名は、タスクの外部リソース ID である必要があります。タスクの外部リソース ID は、タスク ARN の最後の値です。たとえば、ARN 値が arn:aws:dms:us-west-2:5599:task: VFPFKH4FJR3FTYKK2RYSI のタスクの場合、タスクの外部リソース ID は VFPFKH4FJR3FTYKK2RYSI となります。

```
select * from awsdms_validation_failures_v1 where TASK_NAME = 'VFPFKH4FJR3FTYKK2RYSI'

TASK_NAME       VFPFKH4FJR3FTYKK2RYSI
TABLE_OWNER     DB2PERF
TABLE_NAME      PERFTEST
FAILURE_TIME    2020-06-11 21:58:44
```

```
KEY_TYPE        Row
KEY             {"key":  ["3451491"]}
FAILURE_TYPE    RECORD_DIFF
DETAILS         [[{'MYREAL': '+1.10106036e-01'}, {'MYREAL': '+1.10106044e-01'}],]
```

DETAILS フィールドを参照し、一致しない列を特定します。失敗したレコードのプライマリ キーを入手したら、ソース エンドポイントとターゲット エンドポイントをクエリし、レコードの不一致部分を確認できます。

Redshift 検証パフォーマンス

Amazon Redshift は、いくつかの点 (列指向ストレージ、MPP、データ圧縮、その他の要因など) でリレーショナルデータベースとは異なります。これらの違いにより、Redshift はリレーショナルデータベースとは異なるパフォーマンスプロファイルを持ちます。

フルロードレプリケーションフェーズでは、検証で範囲クエリを使用し、データサイズは `PartitionSize` 設定によって決まります。これらの範囲ベースのクエリは、ソーステーブルからすべてのレコードを選択します。

継続的なレプリケーションの場合、クエリは範囲ベースのフェッチと個別レコードのフェッチを切り替えます。クエリタイプは、次のような複数の要因に基づいて動的に決定されます。

- クエリボリューム
- ソーステーブルの DML クエリのタイプ
- タスクレイテンシー
- レコード総数
- 検証設定 (`PartitionSize` など)

検証クエリにより、Amazon Redshift クラスターに追加の負荷がかかる場合があります。上記の要因はユースケースによって異なるため、検証クエリのパフォーマンスを確認してクラスターとテーブルを相応に調整する必要があります。パフォーマンスの問題を軽減するためのオプションには、次のようなものがあります。

- `PartitionSize` および `ThreadCount` の設定を減らして、フルロード検証中のワークロードを軽減します。これにより、データ検証が遅くなることに注意してください。

- Redshift はプライマリキーを強制しませんが、AWS DMS はデータ検証でプライマリキーを使用してターゲットのレコードを一意に識別します。可能であれば、ソートキーをミラーリングするようにプライマリキーを設定し、フルロード検証クエリの実行を迅速化します。

制限事項

- データ検証では、テーブルにプライマリキーまたは一意のインデックスがなければなりません。
  - プライマリキー列の型を CLOB、BLOB、または BYTE に設定することはできません。
  - 型が VARCHAR または CHAR であるプライマリキー列の場合、長さは 1024 未満にする必要があります。データ型の長さを指定する必要があります。無制限のデータ型をデータ検証のプライマリキーとして使用することはできません。
  - NOVALIDATE 句を使用して作成された Oracle キーは、プライマリ キーまたは一意のインデックスと見なされません。
  - プライマリキーがなく一意キーのみを持つ Oracle テーブルの場合、一意制約のある列にも NOT NULL 制約が必要です。
- NULL PK/UK 値の検証はサポートされていません。
- ターゲット PostgreSQL インスタンスのプライマリキー列の照合が「C」に設定されていない場合、プライマリキーと Oracle ではソート順が異なります。PostgreSQL と Oracle でソート順序が異なる場合、データ検証でレコードの検証に失敗します。
- データ検証では、ソースデータベースとターゲットデータベースに対して追加のクエリが生成されます。両方のデータベースに、この追加の負荷を処理するための十分なリソースがあることを確認する必要があります。これは特に Redshift ターゲットに当てはまります。詳細については、「Redshift 検証パフォーマンス」を参照してください。
- 複数のデータベースを 1 つに統合する場合、データ検証はサポートされません。
- ソースまたはターゲット Oracle エンドポイントの場合、AWS DMS は DBMS_CRYPTO を使用して LOB を検証します。Oracle エンドポイントで LOB を使用する場合は、Oracle エンドポイントにアクセスするために使用されるユーザーアカウントに、dbms_crypto での実行権限を付与する必要があります。これを行うには、以下のステートメントを実行します。

```
grant execute on sys.dbms_crypto to dms_endpoint_user;
```

- 検証中にターゲットデータベースが AWS DMS 外部で変更された場合、不整合は正確に報告されない可能性があります。これは、AWS DMS によってターゲットテーブルで検証が実行されている際に、いずれかのアプリケーションがそのテーブルにデータを書き込む場合に発生する可能性があります。

- 1 つまたは複数の行が検証中に継続的に変更される場合、AWS DMS はそれらの行を検証することができません。
- AWS DMS が 10,000 以上の失敗または停止されたレコードを検出した場合、検証は中止されます。先に進む前に、データの根本的な問題を解決する必要があります。
- AWS DMS は、ビューのデータ検証をサポートしていません。
- AWS DMS は、文字置換タスク設定が使用されている場合のデータ検証をサポートしません。
- AWS DMS は、Oracle LONG タイプの検証をサポートしていません。
- AWS DMS は、異種移行中の Oracle Spatial タイプの検証をサポートしていません。

S3 ターゲット検証を使用する際の制限については、「ターゲットの S3 の検証を使用する場合の制限」を参照してください。

Amazon S3 ターゲットのデータ検証

AWS DMS は、Amazon S3 ターゲットでレプリケートされたデータの検証をサポートしています。AWS DMS はレプリケートしたデータを Amazon S3 にフラットファイルとして保存するため、データを検証するには Amazon Athena `CREATE TABLE AS SELECT` (CTAS) クエリを使用します。

Amazon S3 に保存されているデータに対するクエリは、多大なコンピューティングが必要となるため、AWS DMS は、変更データキャプチャ (CDC) 中の Amazon S3 データの検証は、1 日 1 回のみ、UTC の午前 0 時 (00:00) に実行します。AWS DMS が毎日実行する検証は、間隔検証と呼ばれます。間隔検証中、AWS DMS は過去 24 時間にターゲットの Amazon S3 バケットに移行されたすべての変更レコードを検証します。間隔検証の制限の詳細については、「ターゲットの S3 の検証を使用する場合の制限」を参照してください。

Amazon S3 のターゲット検証では Amazon Athena を使用するため、追加料金が適用されます。詳細については、Amazon Athena 料金 を参照してください。

> **Note**
>
> S3 ターゲットの検証には、AWS DMS バージョン 3.5.0 以降が必要です。

トピック

- S3 ターゲット検証の前提条件

- ターゲットの S3 の検証を使用するためのアクセス許可
- ターゲットの S3 の検証を使用する場合の制限
- S3 ターゲット検証での検証のみのタスクの使用

S3 ターゲット検証の前提条件

S3 ターゲット検証を使用する前に、次の設定とアクセス許可を確認します。

- エンドポイントの S3Settings の DataFormat 値を parquet に設定します。詳細については、「S3 の parquet 設定」を参照してください。
- 移行タスクの作成に使用するユーザーアカウントに割り当てられたロールに、適切なアクセス許可のセットが付与されていることを確認します。次の「アクセス許可」を参照してください。

継続的なレプリケーション (CDC) を使用するタスクの場合は、次の設定を確認します。

- 補足ログを有効にすると、CDC データに完全な記録を残せます。補足ロギングを有効にする方法については、このガイドの「トラブルシューティングと診断サポート」セクションの「Oracle ソース エンドポイントにサプリメンタル ログを自動的に追加する」を参照してください。
- ターゲットエンドポイントの TimestampColumnName パラメータを設定します。タイムスタンプ列名には制限はありません。詳細については、[S3Settings] (S3設定)をご参照ください。
- ターゲットでの日付ベースのフォルダ分割を設定します。詳細については、「日付ベースのフォルダパーティション分割を使用する」を参照してください。

ターゲットの S3 の検証を使用するためのアクセス許可

ターゲットの S3 の検証を使用するためのアクセス許可を設定するには、移行タスクの作成に使用するユーザーアカウントに割り当てられたロールに、次のとおりのアクセス許可のセットが付与されていることを確認します。サンプル値は実際の値に置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "athena:StartQueryExecution",
```

```
                "athena:GetQueryExecution",
                "athena:CreateWorkGroup"
            ],
            "Resource": "arn:aws:athena:<endpoint_region_code>:<account_id>:workgroup/
dms_validation_workgroup_for_task_*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:DeleteDatabase",
                "glue:GetDatabase",
                "glue:GetTables",
                "glue:CreateTable",
                "glue:DeleteTable",
                "glue:GetTable"
            ],
            "Resource": [
                "arn:aws:glue:<endpoint_region_code>:<account_id>:catalog",
                "arn:aws:glue:<endpoint_region_code>:<account_id>:database/
aws_dms_s3_validation_*",
                "arn:aws:glue:<endpoint_region_code>:<account_id>:table/
aws_dms_s3_validation_*/*",
                "arn:aws:glue:<endpoint_region_code>:<account_id>:userDefinedFunction/
aws_dms_s3_validation_*/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucketMultipartUploads",
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket_name>",
                "arn:aws:s3:::<bucket_name>/*"
            ]
        }
    ]
}
```

ターゲットの S3 の検証を使用する場合の制限

ターゲットの S3 の検証を使用する場合に適用される追加の制限は、次のとおりです。すべての検証に適用される制限については、「制限事項」を参照してください。

- `DatePartitionSequence` 値には Day コンポーネントが必要です。ターゲットの S3 の検証は、YYYYMM 形式をサポートしていません。
- CDC 中に間隔検証を実行すると、`awsdms_validation_failures_v1` テーブルに誤った検証エラーが表示されることがあります。このようなエラーは、AWS DMS が間隔検証中に取得した変更を翌日のパーティションフォルダに移行するために発生します。このような変更は通常、当日のパーティションフォルダに書き込まれます。このような誤ったエラーは、動的ソースデータベースから Amazon S3 などの静的ターゲットへのレプリケーションの検証の制限です。このような誤ったエラーを調べるには、検証期間の終わり (UTC 00:00) 付近のレコードを確認します。エラーは通常、この時間帯に表示されます。

  誤ったエラーの数を最小限に抑えるには、タスクの `CDCLatencySource` が低く設定されていることを確認します。レイテンシーのモニタリングの詳細については、「レプリケーションのタスクメトリクス」を参照してください。
- `failed` または `stopped` の状態のタスクは前日の変更を検証しません。予期しない障害による検証エラーを最小限に抑えるには、テーブルマッピング、ソースエンドポイント、ターゲットエンドポイントに同じ検証のみのタスクを個別に作成します。データ検証の詳細については、「S3 ターゲット検証での検証のみのタスクの使用」を参照してください。
- テーブル統計の [検証ステータス] 列には、最新の間隔検証の状態が反映されます。そのため、不一致のあるテーブルが翌日の間隔検証後に検証済みとして表示される可能性があります。ターゲットの Amazon S3 バケット内の `s3_validation_failures folder` フォルダを調べて、1 日以上前に発生した不一致がないかを確認します。
- S3 検証では、Amazon Athena のバケット化されたテーブル機能を使用します。これにより、S3 検証でターゲットテーブルデータのバケット化されたコピーを作成できます。つまり、テーブルデータのコピーは、DMS 検証の内部パーティショニングに一致するサブセットに分割されます。Athena のバケット化されたテーブルには、100,000 個のバケットという制限があります。この制限を超えることを S3 検証が検証しようとするテーブルは、検証に失敗します。S3 検証が作成を試みるバケットの数は、次のようになります。

```
(#records in the table) / (validation partition size setting)
```

この制限を回避するには、S3 検証によって作成されたバケットの数が 100,000 未満になるように、検証パーティションサイズ設定を増やします。バケット化の詳細については、「Amazon Athena ユーザーガイド」の「Athena でのパーティション化とバケット化Amazon Athena」を参照してください。

S3 ターゲット検証での検証のみのタスクの使用

検証のみのタスクでは、移行は実行せず、移行されるデータの検証を実行します。

検証のみのタスクでは、移行タスクが停止しても引き続き検証が実行されるため、AWS DMS は 00:00 UTC 間隔の検証期間を見逃すことがありません。

Amazon S3 ターゲットエンドポイントで検証のみのタスクを使用する場合、次の制限があります。

- 検証のみの設定が有効になっているフルロードタスクの Amazon S3 検証はサポートされています。ただし、その他のエンドポイントのフルロードタスクや検証のみのタスクとは動作が異なります。S3 をターゲットとして使用する場合、このようなタイプのタスクは S3 ターゲットのフルロードデータのみを検証し、CDC 移行の一環として移行されたデータは検証されません。この機能は、フルロードのみのタスクが作成したデータを検証する場合にのみ使用します。アクティブな CDC タスクが実行されているターゲットでこのモードを使用してデータを検証しても、効果的な検証は行われません。
- 検証のみのタスクは、最後の間隔検証期間 (UTC 00:00) 以降の変更のみを検証します。検証のみのタスクでは、前日のフルロードデータや CDC データは検証されません。

AWS Database Migration Service でのリソースへのタグ付け

AWS Database Migration Service (AWS DMS) でタグを使用してリソースにメタデータを追加できます。また、これらのタグを AWS Identity and Access Management (IAM) ポリシーで使用して、AWS DMS リソースへのアクセスを管理したり、AWS DMS リソースに適用できるアクションを制御したりできます。最後に、このようなタグを使用して、同様にタグ付けされたリソースのコストをグループ化してコストを追跡できます。

次のすべての AWS DMS リソースにタグを付けることができます。

- 証明書
- データプロバイダー
- データの移行
- エンドポイント
- イベントサブスクリプション
- インスタンスプロファイル
- 移行プロジェクト
- レプリケーションインスタンス
- レプリケーションサブネットグループ
- レプリケーションタスク

AWS DMS タグは、AWS DMS リソースを定義して関連付ける名前と値のペアです。このペアの名前はキーと呼ばれます。キーの値の指定はオプションです。タグを使用して、AWS DMS リソースに任意の情報を割り当てることができます。例えば、タグキーを使用してカテゴリを定義し、タグ値をそのカテゴリ内の項目にすることができます。例えば、「project」というタグキーと「Salix」というタグ値を定義して、AWS DMS リソースが Salix プロジェクトに割り当てられていることを示すことができます。また、タグを使用して environment=test や environment=production などのキーを使って、AWS DMS リソースがテスト用なのか本番稼働用なのかを示すこともできます。AWS DMS リソースに関連付けられたメタデータの追跡が容易になるように、一貫性あるタグキーのセットを使用することをお勧めします。

タグを使用して AWS の請求書を整理すると、独自のコスト構造を反映できます。これを行うには、タグキー値が含まれた AWS アカウント の請求書を取得するようにサインアップする必要がありま

す。その後、結合したリソースのコストを確認するには、同じタグキー値を持つリソース別に請求情報を整理します。例えば、複数のリソースに特定のアプリケーション名をタグ付けし、請求情報を整理して、複数のサービスにわたるそのアプリケーションの合計コストを確認できます。詳細については、「AWS Billing ユーザーガイド」の「コスト配分タグの使用」を参照してください。

各 AWS DMS リソースにはタグセットがあり、AWS DMS リソースに割り当てられているすべてのタグが含まれています。タグセットには、最大 10 個のタグを含めることも、空にすることもできます。AWS DMS リソースに追加したタグのキーがそのリソースの既存のタグのキーと同じ場合、既存の値は新しい値で上書きされます。

AWS は、タグにセマンティックな意義は適用せず、タグは厳密に文字列として解釈されます。AWSDMS は、リソースの作成時に使用した設定に応じて、AWS DMS リソースにタグを設定する場合があります。

AWS DMS タグの特徴は、次のリストの説明のとおりです。

- タグキーは、タグに必須の名前です。文字列値の長さは 1 ~ 128 の Unicode 文字で、プレフィックスとして「aws:」または「dms:」を付けることはできません。文字列には、Unicode 文字、数字、空白、「_」、「.」、「/」、「=」、「+」、「-」(Java 正規表現: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$") のセットのみを含めることができます。

- タグ値は、タグのオプションの文字列値です。文字列値の長さは 1 ~ 256 の Unicode 文字で、プレフィックスとして「aws:」または「dms:」を付けることはできません。文字列には、Unicode 文字、数字、空白、「_」、「.」、「/」、「=」、「+」、「-」(Java 正規表現: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$") のセットのみを含めることができます。

  値はタグセット内で一意である必要はなく、null にすることもできます。例えば、project/Trinity と cost-center/Trinity のタグセット内にキーバリューのペアを使用できます。

AWS CLI または AWS DMS API を使用して、AWS DMS リソースにタグの追加、一覧表示、削除できます。AWS CLI または AWS DMS API を使用する場合は、使用する AWS DMS リソースの Amazon リソースネーム (ARN)を指定する必要があります。ARN の作成の詳細については、「の Amazon リソースネーム (ARN) の構築 AWS DMS」を参照してください。

タグは認証の目的でキャッシュされることに注意します。このため、AWS DMS リソースのタグの追加と更新が利用可能になるまでに数分かかる場合があります。

API

AWS DMS API を使用して AWS DMS リソースのタグを追加、一覧表示、削除できます。

- AWS DMS リソースにタグを追加するには、AddTagsToResource オペレーションを使用します。
- AWS DMS リソースに割り当てられているタグを一覧表示するには、ListTagsForResource オペレーションを使用します。
- AWS DMS リソースからタグを削除するには、RemoveTagsFromResource オペレーションを使用します。

必要な ARN を作成する方法の詳細については、「の Amazon リソースネーム (ARN) の構築 AWS DMS」を参照してください。

AWS DMS API で XML を使用する場合、タグは次のスキーマを使用します。

```
<Tagging>
  <TagSet>
   <Tag>
    <Key>Project</Key>
    <Value>Trinity</Value>
   </Tag>
   <Tag>
    <Key>User</Key>
    <Value>Jones</Value>
   </Tag>
  </TagSet>
</Tagging>
```

使用可能な XML タグと特性の一覧は、次の表のとりです。キーと値では大文字と小文字が区別されることに注意します。例えば、project=Trinity と PROJECT=Trinity は 2 つの別のタグです。

タグ付け要素	説明
TagSet	タグセットは Amazon RDS リソースに割り当てられたすべてのタグのコンテナ。リソースごとに設定できるタグは 1 つのみ。TagSet は、AWS DMS API を介してのみ使用できる。

タグ付け要素	説明
Tag	タグはユーザー定義のキーバリューペア。単一のタグセットには 1 ~ 10 個のタグを含めることができる。
Key	キーはタグの必須の名前。文字列値の長さは 1 ~ 128 の Unicode 文字で、プレフィックスとして「dms:」または「aws:」を付けることはできない。文字列には、Unicode 文字、数字、空白、「_」、「.」、「/」、「=」、「+」、「-」(Java 正規表現: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$") のセットのみを含めることができる。 キーはタグセットに一意である必要がある。例えば、project/Trinity と project/Xanadu のように、キーは同じで値が異なるキーペアをタグセットに含めることはできない。
Value	値は、タグのオプションの値。文字列値の長さは 1 ~ 256 の Unicode 文字で、プレフィックスとして「dms:」または「aws:」を付けることはできない。文字列には、Unicode 文字、数字、空白、「_」、「.」、「/」、「=」、「+」、「-」(Java 正規表現: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$") のセットのみを含めることができる。 値はタグセット内で一意である必要はなく、null にすることもできます。例えば、project/Trinity と cost-center/Trinity のタグセット内にキーバリューのペアを使用できます。

のセキュリティ AWS Database Migration Service

のクラウドセキュリティが最優先事項 AWS です。 AWS のお客様は、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して説明しています。

- クラウドのセキュリティ — 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 は にあります AWS 。 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。AWS 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が定期的にセキュリティの有効性をテストおよび検証しています。に適用されるコンプライアンスプログラムの詳細については AWS DMS、「コンプライアンスAWS プログラムによる対象範囲内の のサービス」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、組織の要件、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、 の使用時に責任共有モデルを適用する方法を理解するのに役立ちます AWS DMS。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するために AWS DMS を設定する方法を示します。また、 AWS DMS リソースのモニタリングや保護に役立つ他の AWS のサービスの使用方法についても説明します。

AWS DMS リソースとデータベース (DBsへのアクセスを管理できます。アクセスの管理に使用する方法は、 で実行する必要があるレプリケーションタスクによって異なります AWS DMS。

- AWS Identity and Access Management (IAM) ポリシーを使用して、 AWS DMS リソースの管理を許可されているユーザーを決定するアクセス許可を割り当て AWS DMS ます。IAM ユーザーとしてサインインする場合は、適切なアクセス許可が必要です。たとえば、IAM を使用して、DB インスタンスのおよびクラスターの作成、記述、変更、削除と、リソースのタグ付け、セキュリティグループの変更をどのユーザーに許可するかを決定できます。IAM とその での使用の詳細については AWS DMS、「」を参照してくださいの Identity and Access Management AWS Database Migration Service。
- AWS DMS は、Transport Layer Security (TLS) とのエンドポイント接続に Secure Sockets Layer (SSL) を使用します。での SSL/TLS の使用の詳細については AWS DMS、「」を参照してくださいでの SSL の使用 AWS Database Migration Service。

- AWS DMS は AWS Key Management Service 、 (AWS KMS) 暗号化キーを使用して、レプリケーション インスタンスで使用されるストレージとそのエンドポイント接続情報を暗号化します。AWS DMS また、 は AWS KMS 、暗号化キーを使用して、Amazon S3 および Amazon Redshift ターゲットエンドポイントの保管中のターゲットデータを保護します。詳細については、「暗号化キーの設定と AWS KMS アクセス許可の指定」を参照してください。
- AWS DMS は、可能な限り最大のネットワークアクセスコントロールを実現するために、常に Amazon VPC サービスに基づいて Virtual Private Cloud (VPC) にレプリケーションインスタンスを作成します。DB インスタンスとインスタンス クラスターには、レプリケーション インスタンスと同じ VPC を使用するか、このレベルのアクセス コントロールに一致させるために追加の VPC を使用します。使用するそれぞれの Amazon VPC は、すべてのポートですべてのトラフィックについて VPC からの送信 (egress) がルールで許可されているセキュリティグループに関連付ける必要があります。このアプローチでは、ソースおよびターゲットデータベースエンドポイントで適切な受信が有効になっている限り、レプリケーション インスタンスからそれらのエンドポイントへの通信が許可されます。

  で使用できるネットワーク設定の詳細については AWS DMS、「」を参照してくださいレプリケーション インスタンスのためのネットワークのセットアップ。VPC での DB インスタンスまたはインスタンス クラスターの作成については、AWS ドキュメントで Amazon データベースのセキュリティとクラスター管理のドキュメントをご参照ください。 AWS DMS でサポートされるネットワーク設定の詳細については、「レプリケーション インスタンスのためのネットワークのセットアップ」をご参照ください。
- データベース移行ログを表示するには、使用している IAM ロールに対する適切な Amazon CloudWatch Logs アクセス許可が必要です。 AWS DMSのログ作成の詳細については、「Amazon CloudWatch を使用したレプリケーションタスクのモニタリング」をご参照ください。

トピック

- でのデータ保護 AWS Database Migration Service
- の Identity and Access Management AWS Database Migration Service
- AWS Database Migration Service のコンプライアンス検証
- AWS Database Migration Service での耐障害性
- AWS Database Migration Service でのインフラストラクチャのセキュリティ
- リソース名とタグを使用したファイングレインアクセスコントロール
- 暗号化キーの設定と AWS KMS アクセス許可の指定
- のネットワークセキュリティ AWS Database Migration Service

- での SSL の使用 AWS Database Migration Service
- データベースのパスワードの変更

でのデータ保護 AWS Database Migration Service

データ暗号化

サポートされている AWS DMS ターゲットエンドポイントのデータリソースの暗号化を有効にすることができます。 は、 AWS DMS とそのすべてのソースエンドポイント AWS DMS とターゲットエンドポイントとの間の接続 AWS DMS も暗号化します。さらに、この暗号化を有効にするために AWS DMS とそのサポートされているターゲットエンドポイントが使用するキーを管理できます。

トピック

- 保管中の暗号化
- 転送中の暗号化
- キー管理

保管中の暗号化

AWS DMS は、サポートされている AWS DMS ターゲットエンドポイントにコピーされる前に、レプリケートされたデータを Amazon S3 にプッシュするために使用するサーバー側の暗号化モードを指定できるようにすることで、保管時の暗号化をサポートします。この暗号化モードは、エンドポイントの追加の接続属性 `encryptionMode` を設定することで指定できます。`encryptionMode` この設定で KMS キー暗号化モードが指定されている場合は、次のターゲットエンドポイントの AWS DMS ターゲットデータを暗号化するためにカスタム AWS KMS キーを作成することもできます。

- Amazon Redshift - `encryptionMode` の設定詳細については、「AWS DMS のターゲットとして Amazon Redshift を使用する場合のエンドポイントの設定」をご参照ください。カスタム AWS KMS 暗号化キーの作成の詳細については、「」を参照してくださいAmazon Redshift ターゲットデータを暗号化する AWS KMS キーの作成と使用。
- Amazon S3 - `encryptionMode` の設定詳細については、「AWS DMS のターゲットとして Amazon S3を使用する場合のエンドポイントの設定」をご参照ください。カスタム AWS KMS 暗号化キーの作成の詳細については、「」を参照してくださいAmazon S3 ターゲットオブジェクトを暗号化する AWS KMS キーの作成。

転送中の暗号化

AWS DMS は、レプリケートするデータがソースエンドポイントからターゲットエンドポイントに安全に移動するようにすることで、転送中の暗号化をサポートします。この措置には、レプリケー

ションパイプラインを経由してデータが移動するときに、レプリケーションタスクが中間ストレージ用に使用するレプリケーション インスタンスの S3 バケットの暗号化も含まれます。ソースエンドポイントとターゲットエンドポイントへのタスク接続を暗号化するには、Secure Socket Layer (SSL) または Transport Layer Security (TLS) AWS DMS を使用します。は、両方のエンドポイントへの接続を暗号化することで、ソースエンドポイントからレプリケーションタスク、およびタスクからターゲットエンドポイントの両方に移動するときに、データの安全性 AWS DMS を確保します。での SSL/TLS の使用の詳細については AWS DMS、「」を参照してください。 [での SSL の使用 AWS Database Migration Service]

AWS DMS は、デフォルトキーとカスタムキーの両方をサポートし、中間レプリケーションストレージと接続情報の両方を暗号化します。これらのキーは、 AWS KMSを使用して管理します。詳細については、「[暗号化キーの設定と AWS KMS アクセス許可の指定]」を参照してください。

キー管理

AWS DMS は、特定のターゲットエンドポイントのレプリケーションストレージ、接続情報、およびターゲットデータストレージを暗号化するためのデフォルトキーまたはカスタムキーをサポートします。これらのキーは、 を使用して管理します AWS KMS。詳細については、「[暗号化キーの設定と AWS KMS アクセス許可の指定]」を参照してください。

インターネットトラフィックのプライバシー

接続には、オンプレミスで実行されているか、クラウド内の AWS サービスの一部として実行されているかにかかわらず、同じ AWS リージョン内の AWS DMS ソースエンドポイントとターゲットエンドポイント間の保護が提供されます。(少なくとも 1 つのエンドポイント、ソース、またはターゲットは、クラウド内の AWS サービスの一部として実行する必要があります)。この保護は、これらのコンポーネントが同じ Virtual Private Cloud (VPC) を共有するか、VPC VPCs がすべて同じ AWS リージョンにある場合、別々の VPCsに存在するかにかかわらず適用されます。でサポートされているネットワーク設定の詳細については、 AWS DMS「」を参照してください[レプリケーション インスタンスのためのネットワークのセットアップ]。これらのネットワーク設定を使用する場合のセキュリティに関する考慮事項については、「[のネットワークセキュリティ AWS Database Migration Service]」をご参照ください。

DMS Fleet Advisor でのデータ保護

DMS Fleet Advisor は、データベースのメタデータを収集して分析し、移行ターゲットの適切なサイズを決定します。DMS Fleet Advisor はテーブル内のデータにアクセスしたり、データを転送したりすることはありません。また、DMS Fleet Advisor はデータベース機能の使用状況は追跡せず、使用統計にもアクセスしません。

DMS Fleet Advisor がデータベースを利用するために使用するデータベースユーザーを作成する際に、データベースへのアクセスを制御します。このようなユーザーに必要なアクセス許可を付与します。DMS Fleet Advisor を使用するには、データベースユーザーに読み取り権限を付与します。DMS Fleet Advisor はデータベースを変更しないため、書き込み権限は必要ありません。詳細については、「AWS DMS Fleet Advisor のデータベースユーザーの作成」を参照してください。

データベースでデータ暗号化を使用できます。 は、DMS Fleet Advisor 内とそのデータコレクター内の接続 AWS DMS も暗号化します。

DMS データコレクターは、データ保護アプリケーションプログラミングインターフェイス (DPAPI) を使用して、お客様の環境とデータベースの認証情報を暗号化、保護、保存します。DMS Fleet Advisor は、このような暗号化されたデータを、DMS データコレクターが動作するサーバー上のファイルに保存します。DMS Fleet Advisor は、このようなサーバーからデータを転送することはありません。DPAPI の詳細については、「方法: データ保護の使用」を参照してください。

DMS データコレクターをインストールすると、このアプリケーションがメトリクスを収集するために実行するすべてのクエリを確認できます。DMS データコレクターをオフラインモードで実行して、収集したデータをサーバー上で確認できます。収集したデータを Amazon S3 バケットで確認することもできます。詳細については、「DMS データコレクターの仕組み」を参照してください。

の Identity and Access Management AWS Database Migration Service

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、誰を認証 (サインイン) し、誰に AWS DMS リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

トピック

対象者

AWS Identity and Access Management (IAM) の使用方法は、 で行う作業によって異なります AWS DMS。

サービスユーザー – AWS DMS サービスを使用してジョブを実行する場合、管理者から必要な認証情報とアクセス許可が与えられます。さらに多くの AWS DMS 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解すると、管理者から適切な権限をリクエストするのに役に立ちます。 AWS DMS機能にアクセスできない場合は、「AWS Database Migration Service ID とアクセスのトラブルシューティング」を参照してください。

サービス管理者 – 社内の AWS DMS リソースを担当している場合は、通常、 へのフルアクセスがあります AWS DMS。サービスユーザーがどの AWS DMS 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で IAM を で使用する方法の詳細については、 AWS DMS「」を参照してくださいが IAM と AWS Database Migration Service 連携する方法。

IAM 管理者 - 管理者は、 AWS DMSへのアクセスを管理するポリシーの書き込み方法の詳細について確認する場合があります。IAM で使用できる AWS DMS アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAWS Database Migration Service アイデンティティベースのポリシーの例。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けて認証（ にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center（ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーティッド ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 ユーザーガイド」の「 へのサインイン AWS アカウント方法AWS サインイン 」を参照してください。

AWS プログラムで にアクセスする場合、 は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号で署名します。 AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、IAM ユーザーガイドの API AWS リクエストの署名を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、 AWS では、多要素認証 (MFA) を使用してアカウントのセキュリティを向上させることをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「IAM ユーザーガイド」の「AWSでの多要素認証 (MFA) の使用」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、IAM ユーザーガイドの「ルートユーザー認証情報が必要なタスク」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションする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「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロール を切り替える AWS Management Console ことで、 で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、 または AWS API AWS CLI オペレーションを呼び出すか、カスタム URL を使

用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「Creating a role for a third-party Identity Provider」(サードパーティーアイデンティティプロバイダー向けロールの作成) を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「アクセス許可セット」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、(ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。クロスアカウントアクセスのロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントリソースアクセス」を参照してください。
- クロスサービスアクセス — 一部の は、他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して でアクションを実行する場合 AWS、ユーザーはプリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストリクエストリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS の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。
- サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスリンクロールの許可を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。 AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、「IAM ユーザーガイド」の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、 AWS ID またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) AWS がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

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。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。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「IAM ポリシーの作成」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「マネージドポリシーとインラインポリシーの比較」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーで IAM の AWS マネージドポリシーを使用することはできません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、 AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、『Amazon Simple Storage Service デベロッパーガイド』の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) に対する最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、 AWS アカウント ビジネスが所有する複数の をグループ化して一元管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、AWS Organizations ユーザーガイドの「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、IAM ユーザーガイドの「ポリシー評価ロジック」を参照してください。

が IAM と AWS Database Migration Service 連携する方法

IAM を使用して へのアクセスを管理する前に AWS DMS、 で使用できる IAM 機能を理解しておく必要があります AWS DMS。 AWS DMS 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「IAM AWS と連携する のサービス」を参照してください。

トピック

- AWS DMS アイデンティティベースのポリシー
- AWS DMS リソースベースのポリシー
- AWS DMS タグに基づく認可
- の IAM ロール AWS DMS
- DMS Fleet Advisor のアイデンティティとアクセス管理

AWS DMS アイデンティティベースのポリシー

IAM アイデンティティベースのポリシーでは、許可または拒否されたアクションとリソースを指定でき、さらにアクションが許可または拒否された条件を指定できます。 AWS DMS 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 ポリシーエレメントのリファレンス」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権限を付与するポリシーで使用されます。

のポリシーアクションは、アクションの前にプレフィックス AWS DMS を使用します`dms:`。例えば、 API オペレーションを使用して AWS DMS `CreateReplicationTask`レプリケーションタ

スクを作成するアクセス許可を付与するには、ポリシーに dms:CreateReplicationTask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 element. AWS DMS defines のいずれかを含める必要があります。このサービスで実行できるタスクを記述する独自のアクションのセットを定義します。

単一のステートメントに複数の 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": [
      "dms:action1",
      "dms:action2"
```

ワイルドカード *を使用して複数のアクションを指定することができます。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": "dms:Describe*"
```

AWS DMS アクションのリストを確認するには、「IAM ユーザーガイド」の「 で定義されるアクション AWS Database Migration Service」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、リソースレベルの権限をサポートしないアクションの場合は、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すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

AWS DMS は、次のリソースで動作します。

- 証明書
- エンドポイント

- イベントサブスクリプション
- レプリケーション インスタンス
- レプリケーションサブネット (セキュリティ) グループ
- レプリケーションタスク

が AWS DMS 必要とするリソースは、呼び出すアクションによって異なります。関連付けられているリソースや、リソース ARN で指定されたリソースに対して、これらのアクションを許可するポリシーが必要です。

例えば、 AWS DMS エンドポイントリソースには次の ARN があります。

```
arn:${Partition}:dms:${Region}:${Account}:endpoint/${InstanceId}
```

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。

たとえば、ステートメントで us-east-2 リージョンの 1A2B3C4D5E6F7G8H9I0J1K2L3M エンドポイント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:dms:us-east-2:987654321098:endpoint/1A2B3C4D5E6F7G8H9I0J1K2L3M"
```

特定のアカウントに属するすべてのエンドポイントを指定するには、ワイルドカード (*) を使用します。

```
"Resource": "arn:aws:dms:us-east-2:987654321098:endpoint/*"
```

リソースを作成するためのアクションなど、一部の AWS DMS 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": "*"
```

一部の AWS DMS API アクションには、複数のリソースが含まれます。たとえば、StartReplicationTask はレプリケーション タスクを開始し、ソースとターゲットの 2 つのデータベース エンドポイント リソースに接続するため、 ユーザーには、ソース エンドポイントを読み取るアクセス許可とターゲット エンドポイントに書き込むアクセス許可が必要です。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" ]
```

ポリシー AWS DMS を使用してリソースへのアクセスを制御する方法の詳細については、「」を参照してくださいリソース名を使用したアクセスの制御。 AWS DMS 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AWS Database Migration Serviceで定義されるリソースを参照してください。どのアクションで各リソースの ARN を指定できるかについては、AWS Database Migration Serviceで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を評価します。1 つの条件キーに複数の値を指定すると、 は論理ORオペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートします。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。

AWS DMS は独自の条件キーのセットを定義し、一部のグローバル条件キーの使用もサポートします。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。

AWS DMS は、条件キーで使用できる一連の標準タグを定義し、独自のカスタムタグを定義することもできます。詳細については、「タグを使用したアクセスへのコントロール」を参照してください。

AWS DMS 条件キーのリストを確認するには、「IAM ユーザーガイド」の「 の条件キー AWS Database Migration Service」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、「AWS Database Migration Serviceで定義されるアクション」と「AWS Database Migration Serviceで定義されるリソース」を参照してください。

例

AWS DMS アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAWS Database Migration Service アイデンティティベースのポリシーの例。

AWS DMS リソースベースのポリシー

リソースベースのポリシーは、指定されたプリンシパルが特定の AWS DMS リソースに対して実行できるアクションと、どの条件下で実行できるかを指定する JSON ポリシードキュメントです。は、サポートされているターゲットエンドポイントに移行されたデータを暗号化するために作成する AWS KMS 暗号化キーのリソースベースのアクセス許可ポリシー AWS DMS をサポートします。サポートされているターゲットエンドポイントには Amazon Redshift や Amazon S3 があります。リソースベースのポリシーを使用することで、これらの暗号化キーを使用するためのアクセス許可を、各ターゲットエンドポイントの他のアカウントに付与できます。

クロスアカウントアクセスを有効にするには、アカウント全体、または別のアカウントの IAM エンティティをリソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる AWS アカウントにある場合は、プリンシパルエンティティにリソースへのアクセス許可も付与する必要があります。アクセス許可は、アイデンティティベースのポリシーをエンティティにアタッチすることで付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、ID ベースのポリシーをさらに付与する必要はありません。詳細については、「IAM ユーザーガイド」の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

この AWS DMS サービスは、 AWS KMS 暗号化キーにアタッチされているキーポリシー と呼ばれるリソースベースのポリシーのタイプを 1 つだけサポートします。このポリシーでは、サポートされているターゲットエンドポイントで移行されたデータを暗号化できるプリンシパルエンティティ (アカウント、ユーザー、ロール、フェデレーティッドユーザー) を定義します。

サポートされているターゲットエンドポイント用に作成する暗号化キーにリソースベースのポリシーをアタッチする方法については、「Amazon Redshift ターゲットデータを暗号化する AWS KMS キーの作成と使用」および「Amazon S3 ターゲットオブジェクトを暗号化する AWS KMS キーの作成」をご参照ください。

例

AWS DMS リソースベースのポリシーの例については、「」を参照してくださいのリソースベースのポリシーの例 AWS KMS。

AWS DMS タグに基づく認可

AWS DMS リソースにタグをアタッチしたり、 へのリクエストでタグを渡すことができます AWS DMS。タグに基づいてアクセスを制御するには、、、または 条件aws:TagKeysキーを使用してポリシーの条件要素にタグ情報を指定します。 はdms:ResourceTag/*key-name*aws:RequestTag/*key-name*、条件キーで使用できる一連の標準タグ AWS DMS を定義し、独自のカスタムタグを定義することもできます。詳細については、「タグを使用したアクセスへのコントロール」を参照してください。

タグに基づいてリソースへのアクセスを制限するアイデンティティベースのポリシーの例については、「タグに基づく AWS DMS リソースへのアクセス」をご参照ください。

の IAM ロール AWS DMS

IAM ロールは、特定のアクセス許可を持つ AWS アカウント内のエンティティです。

での一時的な認証情報の使用 AWS DMS

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。

AWS DMS では、一時的な認証情報の使用がサポートされています。

サービスリンクロール

サービスにリンクされたロールを使用すると、 AWS サービスは他の 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実行できます。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。IAM 管理者は、サービスリンクロールの許可を表示できますが、編集することはできません。

AWS DMS 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、「」を参照してくださいサービスにリンクされたロールの使用。

サービスロール

この機能により、ユーザーに代わってサービスがサービスロールを引き受けることが許可されます。このロールにより、サービスがお客様に代わって他のサービスのリソースにアクセスし、アクションを完了することが許可されます。サービスロールは、IAM アカウントに表示され、アカウントによって所有されます。つまり、IAM 管理者は、このロールの権限を変更できます。ただし、それにより、サービスの機能が損なわれる場合があります。

AWS DMS は、特定のソースエンドポイントまたはターゲットエンドポイントを使用するために作成する必要がある 2 種類のサービスロールをサポートします。

- 次のソースエンドポイントとターゲットエンドポイント (またはそのリソース) への AWS DMS アクセスを許可するアクセス許可を持つロール。
  - ターゲットとしての Amazon DynamoDB — 詳細については、「AWS Database Migration Service のターゲットとして DynamoDB を使用する場合の前提条件」をご参照ください。
  - OpenSearch ターゲットとしての - 詳細については、「」を参照してくださいAmazon OpenSearch Service を AWS Database Migration Service のターゲットとして使用するための前提条件。
  - ターゲットとしての Amazon Kinesis — 詳細については、「のターゲットとして Kinesis データストリームを使用するための前提条件 AWS Database Migration Service」をご参照ください。
  - Amazon Redshift がターゲット - 指定のロールを作成する必要があるのは、データを暗号化するカスタム KMS 暗号化キーを作成する場合か、中間タスクストレージを保持するカスタム S3 バケットを指定する場合のみです。詳細については、「Amazon Redshift ターゲットデータを暗号化する AWS KMS キーの作成と使用」または「Amazon S3 バケットのセットアップ」をご参照ください。
  - ソースまたはターゲットとしての Amazon S3 - 詳細については、のソースとして Amazon S3 を使用する場合の前提条件 AWS DMSまたはターゲットとして Amazon S3 を使用するための前提条件 をご参照ください。

  たとえば、S3 ソースエンドポイントからデータを読み取る場合や、S3 ターゲットエンドポイントにデータをプッシュする場合は、これらのエンドポイントオペレーションごとに S3 にアクセスするための前提条件としてサービスロールを作成する必要があります。
- AWS CLI と AWS DMS API を使用するために必要なアクセス許可を持つロール – 作成する必要がある 2 つの IAM ロールは `dms-vpc-role`と ですdms-cloudwatch-logs-role。Amazon Redshift をターゲットデータベースとして使用する場合は、IAM ロールを作成してdms-access-for-endpoint AWS アカウントに追加する必要があります。詳細については、「AWS CLI および AWS DMS API で使用する IAM ロールの作成」を参照してください。

での IAM ロールの選択 AWS DMS

データベース移行に AWS CLI または AWS DMS API を使用する場合は、 AWS DMS の機能を使用する前に、特定の IAM ロールを AWS アカウントに追加する必要があります。これらのロールのうち 2 つは dms-vpc-role と dms-cloudwatch-logs-role です。Amazon Redshift をターゲットデータベースとして使用する場合は、dms-access-for-endpoint AWS アカウントに IAM ロールも追加する必要があります。詳細については、「AWS CLI および AWS DMS API で使用する IAM ロールの作成」を参照してください。

DMS Fleet Advisor のアイデンティティとアクセス管理

IAM のアイデンティティベースのポリシーを使用すると、許可または拒否されるアクションとリソース、さらにアクションが許可または拒否される条件を指定できます。DMS Fleet Advisor は、特定のアクション、リソース、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 policy elements reference (IAM JSON ポリシーエレメントのリファレンス)」を参照してください。

DMS Fleet Advisor は、IAM ロールを使用して Amazon Simple Storage Service にアクセスします。IAM ロールは、特定のアクセス許可を持つ AWS アカウント内のエンティティです。詳細については、「IAM リソースを作成する」を参照してください。

AWS Database Migration Service アイデンティティベースのポリシーの例

デフォルトでは、IAM ユーザーおよびロールには、 AWS DMS リソースを作成または変更するアクセス許可はありません。また、、 AWS Management Console AWS CLI、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、ユーザーとロールに必要な、指定されたリソースで特定の API オペレーションを実行する権限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。

これらの JSON ポリシードキュメント例を使用して IAM の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
- AWS DMS コンソールを使用する
- 自分の権限の表示をユーザーに許可する
- 1 つの Amazon S3 バケットへのアクセス

- タグに基づく AWS DMS リソースへのアクセス

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で誰かが AWS DMS リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – 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 カ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 を通じ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 – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、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 でのセキュリティのベストプラクティス」を参照してください。

AWS DMS コンソールを使用する

次のポリシーでは、 AWS DMS コンソールを含む AWS DMS へのアクセスを許可し、Amazon EC2 などの他の Amazon サービスから必要な特定のアクションに対するアクセス許可も指定します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dms:*",
            "Resource": "arn:aws:dms:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:PassRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVpcs",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
```

```
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:Get*",
                "cloudwatch:List*"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:FilterLogEvents",
                "logs:GetLogEvents"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        }
    ]
}
```

これらのアクセス許可の内訳は、コンソールを使用するためにそれぞれのアクセス許可が必要な理由を理解するうえで役立ちます。

次のセクションは、利用可能な AWS KMS キーとエイリアスをユーザーがリストし、コンソールに表示することを許可するために必要です。KMS キーの Amazon リソースネーム (ARN) がわかり、AWS Command Line Interface (AWS CLI) のみを使用している場合、このエントリは必要ではありません。

```
{
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases",
                "kms:DescribeKey"
            ],
```

```
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        }
```

次のセクションは、エンドポイントとともにロール ARN を渡す必要がある特定のエンドポイントタイプに必要です。さらに、必要な AWS DMS ロールが事前に作成されていない場合、 AWS DMS コンソールはロールを作成できます。すべてのロールが事前に設定されている場合、必要なものは `iam:GetRole` および `iam:PassRole` のみです。ロールの詳細については、「AWS CLI および AWS DMS API で使用する IAM ロールの作成」をご参照ください。

```
{
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:PassRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        }
```

は Amazon EC2 インスタンスを作成し、作成されたレプリケーションインスタンスのネットワークを設定する AWS DMS 必要があるため、次のセクションは必須です。これらのリソースはお客様のアカウント内に存在するため、お客様に代わってこれらのアクションを実行できる必要があります。

```
{
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVpcs",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        }
```

次のセクションは、ユーザーがレプリケーション インスタンスのメトリクスを表示することを許可するために必要です。

```
{
            "Effect": "Allow",
            "Action": [
                "cloudwatch:Get*",
                "cloudwatch:List*"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        }
```

このセクションは、ユーザーがレプリケーションログを表示することを許可するために必要です。

```
{
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:FilterLogEvents",
                "logs:GetLogEvents"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        }
```

AWS DMS コンソールは、 AWS DMS コンソールを使用するときに自動的に AWS アカウントにアタッチされる複数のロールを作成します。移行に AWS Command Line Interface （ AWS CLI) または AWS DMS API を使用する場合は、これらのロールをアカウントに追加する必要があります。これらのロールの追加についての詳細は、「AWS CLI および AWS DMS API で使用する IAM ロールの作成」をご参照ください。

このポリシーを使用して AWS DMS にアクセスするための要件の詳細については、「」を参照してくださいAWS DMSの使用に必要な IAM アクセス許可。

自分の権限の表示をユーザーに許可する

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 S3 バケットへのアクセス

AWS DMS は、データベース移行の中間ストレージとして Amazon S3 バケットを使用します。通常、 AWS DMS はこの目的のためにデフォルトの S3 バケットを管理します。ただし、特定のケースでは、特に AWS CLI または AWS DMS API を使用する場合、 AWS DMS では代わりに独自の S3 バケットを指定できます。たとえば、Amazon Redshift ターゲット エンドポイントにデータを移行するための独自の S3 バケットを指定できます。この場合、 AWS管理`AmazonDMSRedshiftS3Role`ポリシーに基づいてアクセス許可を持つロールを作成する必要があります。

次の例は `AmazonDMSRedshiftS3Role` ポリシーの 1 つのバージョンを示しています。これにより、 AWS DMS は AWS アカウント内の IAM ユーザーに Amazon S3 バケットの 1 つへのアクセス

権を付与できます。また、このユーザーは、オブジェクトの追加、更新、削除を行うこともできます。

このポリシーでは、ユーザーに s3:PutObject、s3:GetObject、s3:DeleteObject のアクセス許可を付与するだけでなく、s3:ListAllMyBuckets、s3:GetBucketLocation、および s3:ListBucket のアクセス許可も付与します。これらが、コンソールで必要とされる追加のアクセス許可です。その他のアクセス許可により、 AWS DMS はバケットのライフサイクルを管理できます。また、オブジェクトをコピーするには、s3:GetObjectAcl アクションが必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:ListBucket",
                "s3:DeleteBucket",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:GetObjectVersion",
                "s3:GetBucketPolicy",
                "s3:PutBucketPolicy",
                "s3:GetBucketAcl",
                "s3:PutBucketVersioning",
                "s3:GetBucketVersioning",
                "s3:PutLifecycleConfiguration",
                "s3:GetLifecycleConfiguration",
                "s3:DeleteBucketPolicy"
            ],
            "Resource": "arn:aws:s3:::dms-*"
        }
    ]
}
```

このポリシーに基づくロールの作成の詳細については、「Amazon S3 バケットのセットアップ」をご参照ください。

タグに基づく AWS DMS リソースへのアクセス

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 AWS DMS リソースへのアクセスをコントロールできます。この例では、すべての AWS DMS エンドポイントへのアクセスを許可するポリシーを作成する方法を示します。ただし、アクセス許可が付与されるのは、エンドポイントデータベースタグ Owner にそのユーザーのユーザー名の値がある場合のみです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dms:*",
            "Resource": "arn:aws:dms:*:*:endpoint/*",
            "Condition": {
                "StringEquals": {"dms:endpoint-tag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

このポリシーはアカウントの IAM ユーザーにアタッチできます。という名前のユーザーが AWS DMS エンドポイントにアクセスrichard-roeしようとする場合、エンドポイントデータベースに Owner=richard-roeまたは のタグを付ける必要がありますowner=richard-roe。それ以外の場合、このユーザーはアクセスを拒否されます。条件キー名では大文字と小文字が区別されないため、条件タグキー Owner は Owner と owner の両方に一致します。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。

のリソースベースのポリシーの例 AWS KMS

AWS DMS では、カスタム AWS KMS 暗号化キーを作成して、サポートされているターゲットエンドポイントデータを暗号化できます。キーポリシーを作成して、サポートされているターゲットデータ暗号化用に作成する暗号化キーに、このポリシーをアタッチする方法については、「Amazon Redshift ターゲットデータを暗号化する AWS KMS キーの作成と使用」および「Amazon S3 ターゲットオブジェクトを暗号化する AWS KMS キーの作成」をご参照ください。

トピック

- Amazon Redshift ターゲットデータを AWS KMS 暗号化するためのカスタム暗号化キーのポリシー
- Amazon S3 ターゲットデータを AWS KMS 暗号化するためのカスタム暗号化キーのポリシー

Amazon Redshift ターゲットデータを AWS KMS 暗号化するためのカスタム暗号化キーのポリシー

次の例は、Amazon Redshift ターゲットデータを暗号化するために作成する AWS KMS 暗号化キー用に作成されたキーポリシーの JSON を示しています。

```
{
  "Id": "key-consolepolicy-3",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::987654321098:root"
        ]
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow access for Key Administrators",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::987654321098:role/Admin"
        ]
      },
      "Action": [
        "kms:Create*",
        "kms:Describe*",
        "kms:Enable*",
        "kms:List*",
        "kms:Put*",
        "kms:Update*",
        "kms:Revoke*",
        "kms:Disable*",
        "kms:Get*",
        "kms:Delete*",
        "kms:TagResource",
        "kms:UntagResource",
        "kms:ScheduleKeyDeletion",
```

```
        "kms:CancelKeyDeletion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::987654321098:role/DMS-Redshift-endpoint-access-role"
        ]
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::987654321098:role/DMS-Redshift-endpoint-access-role"
        ]
      },
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": "*",
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": true
        }
      }
    }
  ]
}
```

ここでは、キーを作成する前に作成した Amazon Redshift ターゲットエンドポイントデータにアクセスするためのロールが、キーポリシーで参照されている箇所を確認できます。この例では、DMS-Redshift-endpoint-access-role です。また、異なるプリンシパル (ユーザーとロール) に対して許可されているさまざまなキーアクションも確認できます。たとえば、DMS-Redshift-endpoint-access-role のすべてのユーザーは、ターゲットデータを暗号化、復号化、および再暗号化できます。このようなユーザーは、エクスポート用のデータキーを生成して、 の外部でデータを暗号化することもできます AWS KMS。また、先ほど作成した AWS KMS キーなど、キーに関する詳細情報を返すこともできます。さらに、このようなユーザーは、ターゲット エンドポイントなどの AWS リソースへのアタッチメントを管理できます。

Amazon S3 ターゲットデータを AWS KMS 暗号化するためのカスタム暗号化キーのポリシー

次の例は、Amazon S3 ターゲットデータを暗号化するために作成する AWS KMS 暗号化キー用に作成されたキーポリシーの JSON を示しています。

```
{
  "Id": "key-consolepolicy-3",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::987654321098:root"
        ]
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow access for Key Administrators",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::987654321098:role/Admin"
        ]
      },
      "Action": [
        "kms:Create*",
```

```
        "kms:Describe*",
        "kms:Enable*",
        "kms:List*",
        "kms:Put*",
        "kms:Update*",
        "kms:Revoke*",
        "kms:Disable*",
        "kms:Get*",
        "kms:Delete*",
        "kms:TagResource",
        "kms:UntagResource",
        "kms:ScheduleKeyDeletion",
        "kms:CancelKeyDeletion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::987654321098:role/DMS-S3-endpoint-access-role"
        ]
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::987654321098:role/DMS-S3-endpoint-access-role"
        ]
      },
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
```

```
        "kms:RevokeGrant"
      ],
      "Resource": "*",
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": true
        }
      }
    }
  ]
```

ここでは、キーの作成前に作成した Amazon S3 ターゲットエンドポイントデータにアクセスするためのロールが、キーポリシーで参照されている箇所を確認できます。この例では、DMS-S3-endpoint-access-role です。また、異なるプリンシパル (ユーザーとロール) に対して許可されているさまざまなキーアクションも確認できます。たとえば、DMS-S3-endpoint-access-role のすべてのユーザーは、ターゲットデータを暗号化、復号化、および再暗号化できます。このようなユーザーは、エクスポート用のデータキーを生成して、 の外部でデータを暗号化することもできます AWS KMS。また、先ほど作成した AWS KMS キーなど、キーに関する詳細情報を返すこともできます。さらに、このようなユーザーは、ターゲット エンドポイントなどの AWS リソースへのアタッチメントを管理できます。

シークレットを使用して AWS Database Migration Service エンドポイントにアクセスするには

の場合 AWS DMS、シークレットは暗号化されたキーであり、シークレット認証 を通じて、サポートされている AWS DMS ソースまたはターゲットエンドポイントのデータベース接続を認証するために、一連のユーザー認証情報を表すために使用できます。Oracle Automatic Storage Management (ASM) も使用する Oracle エンドポイント AWS DMS の場合、Oracle ASM にアクセスするためのユーザー認証情報を表す追加のシークレットが必要です。

を使用してシークレット認証 AWS DMS に必要なシークレットを作成できます。 AWS Secrets Managerこれは、クラウド内およびオンプレミスのアプリケーション、サービス、IT リソースにアクセスするための認証情報を安全に作成、保存、取得するためのサービスです。これは、ユーザーを介さずに認証情報のセキュリティを強化する、暗号化されたシークレット値の定期的な自動ローテーションに対応します。 AWS Secrets Manager また、 でシークレット値のローテーションを有効にすると、シークレットに依存するデータベースの移行に影響を与えずに、このシークレット値のローテーションが確実に実行されます。エンドポイント データベース接続を内密に認証するには、エンドポイント設定に含める SecretsManagerSecretId に割り当てる ID または ARN シークレットを作成します。Oracle ASM を Oracle エンドポイントの一部として内密に認証するには、エンドポイ

ント設定にも含める SecretsManagerOracleAsmSecretId に ID または ARN を割り当てるシークレットを作成します。

> **Note**
>
> Amazon RDS Aurora が管理するマスター認証情報は使用できません。これらの認証情報には、接続を確立 AWS DMS する必要があるホストまたはポート情報は含まれません。その代わりに、新しいユーザーとシークレットを作成します。ユーザーとシークレットの作成の詳細については、次の「AWS Management Console を使用してシークレットとシークレットアクセスロールを作成する」を参照してください。

の詳細については AWS Secrets Manager、「 AWS Secrets Manager ユーザーガイドAWS 」の「Secrets Manager とは」を参照してください。

AWS DMS は、サポートされているソースエンドポイントとターゲットエンドポイントで、以下のオンプレミスデータベースまたは AWSマネージドデータベースのシークレット認証をサポートします。

- Amazon DocumentDB
- IBM Db2 LUW
- Microsoft SQL Server
- MongoDB
- MySQL
- Oracle
- PostgreSQL
- Amazon Redshift
- SAP ASE

これらのデータベースに接続するにはエンドポイント設定の一部として、次のいずれかの値セットを入力できますが、両方を入力することはできません。

- UserName、Password、ServerName、Port の設定を使用したデータベース接続を認証するクリアテキスト値。Oracle ASM も使用する Oracle エンドポイントの場合は、AsmUserName、AsmPassword、AsmServerName の設定を使用して ASM を認証するために追加のクリアテキスト値を含めます。

- SecretsManagerSecretId と SecretsManagerAccessRoleArn 設定用の値を使用したシークレット認証。Oracle ASM を使用する Oracle エンドポイントの場合は、SecretsManagerOracleAsmSecretIdとSecretsManagerOracleAsmAccessRoleArn 設定用に追加の値を含めます。これらの設定のシークレット値には次のものが含まれます。
  - SecretsManagerSecretId — AWS Secrets Managerでエンドポイントデータベースアクセス用に作成したシークレットの完全な Amazon リソースネーム (ARN)、ARN の一部、またはフレンドリ名。
  - SecretsManagerAccessRoleArn – ユーザーに代わってこのシークレットへのアクセスを提供するために IAM で作成したSecretsManagerSecretIdシークレット AWS DMS アクセスロールの ARN。
  - SecretsManagerOracleAsmSecretId — AWS Secrets Managerで Oracle ASM アクセス用に作成したシークレットの完全な Amazon リソースネーム (ARN)、ARN の一部、またはフレンドリ名。
  - SecretsManagerOracleAsmAccessRoleArn — ユーザーに代わってこの SecretsManagerOracleAsmSecretId シークレットへの AWS DMS アクセスを提供するために IAM に作成したシークレットアクセスロールの ARN。

> **Note**
>
> 単一のシークレットアクセスロールを使用して、シークレットとSecretsManagerSecretIdシークSecretsManagerOracleAsmSecretIdレットの両方 AWS DMS へのアクセスを提供することもできます。両方のシークレットに対してこの単一のシークレットアクセスロールを作成する場合は、このアクセスロール用の同じ ARN を SecretsManagerAccessRoleArn と SecretsManagerOracleAsmAccessRoleArn の両方に割り当てます。例えば、両方のシークレットに対するシークレット アクセス ロールの ARN が変数 ARN2xsecrets に割り当てられている場合、これらの ARN 設定を次のように設定できます。
>
> ```
> SecretsManagerAccessRoleArn = ARN2xsecrets;
> SecretsManagerOracleAsmAccessRoleArn = ARN2xsecrets;
> ```

これらの値を作成する詳細については、「AWS Management Console を使用してシークレットとシークレットアクセスロールを作成する」をご参照ください。

エンドポイントに必要なシークレットおよびシークレットアクセスロールエンドポイント設定を作成し指定した後、これらのシークレットの詳細を含む CreateEndpoint または ModifyEndpoint API リクエストを実行するユーザーアカウントの許可を更新します。これらのアカウントのアクセスIAM:GetRole許可に、シークレットアクセスロールに対する アクセスSecretsManager:DescribeSecret許可と、シークレットに対する アクセス許可が含まれていることを確認します。 アクセスロールとそのシークレットの両方を検証するために、 はこれらのアクセス許可 AWS DMS を必要とします。

必要なユーザー許可を提供して検証するには

1. にサインイン AWS Management Console し、 で AWS Identity and Access Management コンソールを開きますhttps://console.aws.amazon.com/iam/。
2. [Users] (ユーザー) を選択し、CreateEndpoint と ModifyEndpoint の API コールを行うために使用する [User ID] (ユーザー ID) を選択します。
3. [Permissions] (許可) タブで、[{} JSON] を選択します。
4. ユーザーに次のアクセス許可があることを確認します。

```
{
 "Statement": [{
   "Effect": "Allow",
   "Action": [
    "iam:GetRole",
    "iam:PassRole"
   ],
   "Resource": "SECRET_ACCESS_ROLE_ARN"
  },
  {
   "Effect": "Allow",
   "Action": "secretsmanager:DescribeSecret",
   "Resource": "SECRET_ARN"
  }
 ]
}
```

5. ユーザーにこれらの許可がない場合は、許可を追加します。
6. IAM ロールを使用して DMS API コールを行う場合は、それぞれのロールで上記のステップを繰り返します。

7. ターミナルを開き、 を使用して、上記で使用したロールまたはユーザーを引き受けることで、アクセス許可が正しく付与されていること AWS CLI を検証します。

   a. IAM `get-role` コマンド SecretAccessRole を使用して、 に対するユーザーのアクセス許可を検証します。

   ```
   aws iam get-role --role-name ROLE_NAME
   ```

   ROLE_NAME は口 `SecretsManagerAccessRole` 名前に置き換えます。

   コマンドがエラーメッセージを返す場合は、正しく許可されていることを確認してください。

   b. Secrets Manager `describe-secret` コマンドを使用して、シークレットに対するユーザー許可を検証します。

   ```
   aws secretsmanager describe-secret --secret-id SECRET_NAME OR SECRET_ARN --
   region=REGION_NAME
   ```

   ユーザーには、フレンドリ名、ARN の一部、または完全な ARN を指定できます。詳細については、「describe-secret」をご参照ください。

   コマンドがエラー メッセージを返す場合は、正しく許可されていることを確認してください。

AWS Management Console を使用してシークレットとシークレットアクセスロールを作成する

を使用して AWS Management Console 、エンドポイント認証用のシークレットを作成し、ユーザーに代わって がシークレットにアクセス AWS DMS できるようにするポリシーとロールを作成できます。

を使用してシークレットを作成するには AWS Management Console 、 を使用してソースエンドポイントとターゲットエンドポイントの接続のデータベースを認証 AWS DMS します。

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

2. [新しいシークレットを保存] を選択します。
3. [Store a new secret] (新しいシークレットの保存) ページの [Select secret type] (シークレットタイプの選択) で、[Other type of secrets] (他の種類のシークレット) を選択し、次に[Plaintext] (プレーンテキスト) を選択します。

> **Note**
>
> これ以降で、エンドポイントデータベースに接続するために、クリアテキストの認証情報を入力する必要があるのはここだけです。

4. [Plaintext] (プレーンテキスト) フィールド

   - `SecretsManagerSecretId` にアイデンティティを割り当てるシークレットでは、次の JSON 構造を入力します。

```
{
  "username": db_username,
  "password": db_user_password,
  "port": db_port_number,
  "host": db_server_name
}
```

> **Note**
>
> これは、エンドポイントデータベースの認証に必要な JSON メンバーの最小リストです。任意の JSON エンドポイント設定をすべて小文字で JSON メンバーとして追加できます。ただし、 AWS DMS では、エンドポイント認証用の追加 JSON メンバーは無視されます。

ここで、*db_username* は、データベースにアクセスしているユーザーの名前で、*db_user_password* は、データベースのユーザーパスワード、*db_port_number* は、データベースにアクセスするためのポート番号、*db_server_name* は、次の例のように、ウェブ上のデータベースサーバー名 (アドレス) です。

```
{
  "username": "admin",
  "password": "some_password",
  "port": "8190",
```

```
  "host": "oracle101.abcdefghij.us-east-1.rds.amazonaws.com"
}
```

- SecretsManagerOracleAsmSecretId にアイデンティティを割り当てるシークレットについては、次の JSON 構造を入力します。

```
{
  "asm_user": asm_username,
  "asm_password": asm_user_password,
  "asm_server": asm_server_name
}
```

> **Note**
>
> これは、Oracle エンドポイント用に Oracle ASM を認証するために最低限必要な JSON メンバーのリストです。また、利用可能な Oracle ASM エンドポイント設定に基づいて指定できる完全なリストでもあります。

ここで、*asm_username* は、Oracle ASM にアクセスしているユーザー名、*asm_user_password* は Oracle ASM ユーザーのパスワード、*asm_server_name* は、次の例のように、ポートを含むウェブの Oracle ASM サーバー名 (アドレス) です。

```
{
  "asm_user": "oracle_asm_user",
  "asm_password": "oracle_asm_password",
  "asm_server": "oracle101.abcdefghij.us-east-1.rds.amazonaws.com:8190/+ASM"
}
```

5. シークレットを AWS KMS 暗号化する暗号化キーを選択します。によって AWS Secrets Manager サービス用に作成されたデフォルトの暗号化キーを受け入れるか、作成した AWS KMS キーを選択できます。
6. このシークレットを参照する名前とオプションの説明を指定します。これは、SecretsManagerSecretId または SecretsManagerOracleAsmSecretId の値として使用するフレンドリ名です。
7. シークレットで自動ローテーションを有効にする場合は、説明に従ってシークレットの認証情報をローテーションするアクセス許可を持つ AWS Lambda 関数を選択または作成する必要があります。ただし、Lambda 関数を使用するように自動ローテーションを設定する前に、関数の構成設定で次の 4 文字を EXCLUDE_CHARACTERS 環境変数の値に追加します。

```
;.:+{}
```

AWS DMS では、エンドポイント認証情報に使用されるパスワードにこれらの文字は使用できません。これらを除外するように Lambda 関数を設定すると、 AWS Secrets Manager はローテーションされたパスワードの値の一部としてこれらの文字を生成できなくなります。Lambda 関数を使用するように自動ローテーションを設定すると、 はシークレットを AWS Secrets Manager すぐにローテーションしてシークレット設定を検証します。

> **Note**
>
> データベースエンジンの構成によっては、データベースでローテーションされた認証情報がフェッチされない場合があります。この場合、認証情報を更新するには、タスクをマニュアルで再起動する必要があります。

8. シークレットを確認して に保存します AWS Secrets Manager。その後、 で各シークレットをわかりやすい名前で検索し AWS Secrets Manager、シークレット ARN を の値として、`SecretsManagerSecretId`または`SecretsManagerOracleAsmSecretId`必要に応じて取得して、エンドポイントデータベース接続と Oracle ASM (使用されている場合) へのアクセスを認証できます。

シークレットアクセスポリシーとロールを作成して **`SecretsManagerAccessRoleArn`**または を設定するには**`SecretsManagerOracleAsmAccessRoleArn`**、 が AWS DMS 適切なシークレットにアクセス AWS Secrets Manager できるようにします。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/ で AWS Identity and Access Management (IAM) コンソールを開きます。
2. [Policies] (ポリシー) を選択し、次に[Create Policy] (ポリシー作成) を選択します
3. [JSON] を選択し、次のポリシーを入力して、シークレットへのアクセスと復号化を有効にします。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": secret_arn,
```

```
        },
        {
             "Effect": "Allow",
             "Action": [
                        "kms:Decrypt",
                        "kms:DescribeKey"
                    ],
             "Resource": kms_key_arn,
        }
     ]
}
```

ここで、*secret_arn* はシークレット ARN で、必要に応じて SecretsManagerSecretId または SecretsManagerOracleAsmSecretId のいずれかより入手できます。*kms_key_arn* は、次の例のように、シークレットを暗号化するために使用する AWS KMS キーの ARN です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-2:123456789012:secret:MySQLTestSecret-qeHamH"
        },
        {
             "Effect": "Allow",
             "Action": [
                        "kms:Decrypt",
                        "kms:DescribeKey"
                    ],
             "Resource": "arn:aws:kms:us-
east-2:123456789012:key/761138dc-0542-4e58-947f-4a3a8458d0fd"
        }
     ]
}
```

> **Note**
>
> によって作成されたデフォルトの暗号化キーを使用する場合 AWS Secrets Manager、の AWS KMS アクセス許可を指定する必要はありません*kms_key_arn*。

ポリシーで両方のシークレットへのアクセスを許可する場合は、もう 1 つの *[secret_arn]* に追加の JSON リソースオブジェクトを指定するだけです。
シークレットが別のアカウントにある場合は、SecretsManagerAccessRoleArn ロールにクロスアカウントのシークレットを検証するための追加のポリシーが必要です。このようなユースケースでは、ポリシーに secretsmanager:DescribeSecret アクションを追加します。クロスアカウントシークレットの設定の詳細については、「別のアカウント のユーザーの AWS Secrets Manager シークレットへのアクセス許可」を参照してください。

4. フレンドリ名とオプションの説明を使用して、ポリシーを確認して作成します。
5. [Roles] (ロール)、[Create role] (ロールの作成) の順に選択します。
6. 信頼されたエンティティの種類に、[AWS service] (サービス) を選択します。
7. 信頼されたサービスのリストから [DMS] を選択し、[Next: Permissions] (次へ: アクセス許可) を選択します。
8. ステップ 4 で作成したポリシーを検索しアタッチし、任意のタグを追加しながらロールを確認します。この時点で、 ロールの信頼関係を編集して、 AWS DMS リージョンサービスプリンシパルを信頼されたエンティティとして使用します。このプリンシパルの形式は以下のとおりです。

```
dms.region-name.amazonaws.com
```

ここで、*region-name* は 例えば us-east-1 などといったリージョンの名前です。したがって、この AWS DMS リージョンのリージョンサービスプリンシパルが続きます。

```
dms.us-east-1.amazonaws.com
```

9. ロールの信頼されたエンティティを編集した後、フレンドリ名とオプションの説明を使用してロールを作成します。IAM でフレンドリ名を持つ新しいロールを検索し、ロール ARN を SecretsManagerAccessRoleArn または SecretsManagerOracleAsmAccessRoleArn の値として取得して、エンドポイントデータベース接続を認証することができるようになりました。

プライベートサブネットのレプリケーションインスタンスで Secrets Manager を使用するには

1. Secrets Manager VPC エンドポイントを作成し、エンドポイントの DNS を書き留めます。Secrets Manager VPC エンドポイントの作成方法の詳細については、AWS Secrets

Manager ユーザーガイドの「VPC endpoint を介した Secrets Manager への接続」 を参照ください。

2. レプリケーションインスタンスのセキュリティグループを Secrets Manager VPC エンドポイントにアタッチします。
3. レプリケーションインスタンスのセキュリティグループのエグレスルールでは、送信先 `0.0.0.0/0` のすべてのトラフィックを許可します。
4. エンドポイントの追加接続属性 `secretsManagerEndpointOverride=`*`secretsManager endpoint DNS`* を設定し、次の例に示すように、シークレットマネージャーの VPC エンドポイント DNS を入力します。

```
secretsManagerEndpointOverride=vpce-1234a5678b9012c-12345678.secretsmanager.eu-west-1.vpce.amazonaws.com
```

AWS DMS のサービスにリンクされたロールの使用

AWS Database Migration Service は AWS Identity and Access Management (IAM) の 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、AWS DMS に直接関連付けられた固有のタイプの IAM ロールです。サービスにリンクされたロールは、AWS DMS により事前定義済みのロールであり、ユーザーに代わってサービスからその他の 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を備えています。

サービスにリンクされたロールを使用すると、必要な許可を手動で追加する必要がなくなるため、AWS DMS の設定が簡単になります。AWS DMS は、このサービスにリンクされたロールのアクセス許可を定義します。特に定義されていない限り、AWS DMS 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ており、このアクセス許可ポリシーをその他の IAM エンティティにアタッチすることはできません。

サービスにリンクされたロールは、まずその関連リソースを削除しない限り削除できません。これにより、リソースへのアクセス権限を誤って削除することを避けることができ、AWS DMS リソースは保護されます。

サービスにリンクされたロールをサポートするその他のサービスについては、「IAM と連携する AWS のサービス」で サービスにリンクされたロール 列に はい と記載されているサービスを検索してください。リンク付きの はい を選択すると、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が表示されます。

AWS DMS 機能のサービスにリンクされたロール

トピック

- AWS DMS Fleet Advisor のサービスにリンクされたロール
- AWS DMS Serverless のサービスにリンクされたロール

AWS DMS Fleet Advisor のサービスにリンクされたロール

AWS DMS Fleet Advisor は、AWSServiceRoleForDMSFleetAdvisor 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。DMS Fleet Advisor は、Amazon CloudWatch メトリクスの管理にこのサービスにリンクされたロールを使用します。このサービスにリンクされたロールは、AWSDMSFleetAdvisorServiceRolePolicy 管理ポリシーにアタッチされます。このポリシーの最新情報については、「AWS の マネージドポリシー AWS Database Migration Service」を参照してください

サービスにリンクされたロール AWSServiceRoleForDMSFleetAdvisor は、次のサービスを信頼してロールを引き受けます。

- dms-fleet-advisor.amazonaws.com

AWSDMSFleetAdvisorServiceRolePolicy という名前のロールアクセス許可ポリシーにより、AWS DMS Fleet Advisor は指定されたリソースに対して次のアクションを完了できます。

- アクション: all AWS resources での cloudwatch:PutMetricData

  このアクセス許可を使用して、プリンシパルは Amazon CloudWatch にメトリクスのデータポイントを公開できます。AWS DMSDMS Fleet Advisor では、CloudWatch からのデータベースメトリクスを含むグラフを表示するためにこのアクセス許可が必要です。

次のコード例は、AWSDMSFleetAdvisorServiceRolePolicy ロールを作成するために使用する AWSDMSFleetAdvisorServiceRolePolicy ポリシーを示しています。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Resource": "*",
        "Action": "cloudwatch:PutMetricData",
        "Condition": {
            "StringEquals": {
```

```
                "cloudwatch:namespace": "AWS/DMS/FleetAdvisor"
            }
        }
    }
}
```

ユーザー、グループ、ロールなどの IAM エンティティがサービスにリンクされたロールを作成、編集、削除できるようにするには、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。

AWS DMS Fleet Advisor のサービスにリンクされたロールの作成

IAM コンソールを使用して、[DMS – Fleet Advisor] ユースケースでサービスにリンクされたロールを作成できます。AWS CLI または AWS API で、`dms-fleet-advisor.amazonaws.com` サービス名を使用してサービスにリンクされたロールを作成します。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作成」を参照してください。このサービスにリンクされたロールを削除すれば、同じプロセスを使用して、もう一度ロールを作成できます。

データコレクターを作成する前に、このロールを必ず作成します。DMS Fleet Advisor はこのロールを使用して AWS Management Console でデータベースメトリクスを含むグラフを表示します。詳細については、「データコレクターの作成」を参照してください。

AWS DMS Fleet Advisor のサービスにリンクされたロールの編集

AWS DMS では、AWSServiceRoleForDMSFleetAdvisor の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がそのロールを参照するる可能性があるため、ロール名は変更することはできません。ただし、IAM を使用してロールの説明の編集はできます。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。

AWS DMS Fleet Advisor のサービスにリンクされたロールの削除

サービスにリンクされたロールを必要とする機能またはサービスを使用する必要がなくなった場合は、そのロールを削除することをお勧めします。これにより、アクティブにモニタリングまたはメンテナンスされない未使用のエンティティを低減できます。ただし、手動で削除する前に、サービスにリンクされたロールのリソースをクリーンアップする必要があります。

> **Note**
>
> リソースを削除する際に AWS DMS のサービスで該当ロールが使用されている場合、削除が失敗することがあります。失敗した場合は、数分待ってからもう一度オペレーションを実行します。

AWSServiceRoleForDMSFleetAdvisor が使用する AWS DMS リソースを削除するには

1. AWS Management Console にサインインして、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。
2. ナビゲーションペインで、[検出] の下にある [データコレクター] を選択します。[データコレクター] ページが開きます。
3. データコレクターを選択して、[削除] をクリックします。
4. 削除を確認するには、テキスト入力フィールドにデータコレクター名を入力します。次に、[削除] を選択します。

> **Important**
>
> DMS データコレクターを削除すると、DMS Fleet Advisor は、このコレクターを使用して検出したすべてのデータベースをインベントリから削除します。

すべてのデータコレクターを削除したら、サービスにリンクされたロールを削除できます。

IAM を使用してサービスリンクロールを手動で削除するには

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 AWSServiceRoleForDMSFleetAdvisor を削除します。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。

AWS DMS Fleet Advisor のサービスにリンクされたロールをサポートするリージョン

AWS DMS Fleet Advisor は、このサービスが利用可できる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、「サポートされている AWS リージョン」を参照してください。

AWS DMS Serverless のサービスにリンクされたロール

AWS DMS サーバーレスは、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。AWSServiceRoleForDMSServerless AWS DMS は、このサービスにリンクされたロールを使用して、Amazon AWS DMS CloudWatch メトリックスなどのリソースをユーザーに代わって作成および管理します。 AWS DMS はこのロールを使用するので、必要なのはレプリケーションだけです。このサービスリンクロールは、マネージドポリシー AWSDMSServerlessServiceRolePolicy にアタッチされます。このポリシーの更新については、「AWS の マネージドポリシー AWS Database Migration Service」を参照してください

AWSServiceRoleForDMSServerless サービスにリンクされたロールは、以下のサービスを信頼してロールを引き受けます。

- dms.amazonaws.com

次のコード例は、 AWSDMSServerlessServiceRolePolicy ロールの作成に使用するポリシーを示しています。 AWSServiceRoleForDMSServerless

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "id0",
            "Effect": "Allow",
            "Action": [
                "dms:CreateReplicationInstance",
                "dms:CreateReplicationTask"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "dms:req-tag/ResourceCreatedBy": "DMSServerless"
                }
            }
        },
        {
            "Sid": "id1",
            "Effect": "Allow",
            "Action": [
                "dms:DescribeReplicationInstances",
                "dms:DescribeReplicationTasks"
```

```
            ],
            "Resource": "*"
        },
        {
            "Sid": "id2",
            "Effect": "Allow",
            "Action": [
                "dms:StartReplicationTask",
                "dms:StopReplicationTask",
                "dms:DeleteReplicationTask",
                "dms:DeleteReplicationInstance"
            ],
            "Resource": [
                "arn:aws:dms:*:*:rep:*",
                "arn:aws:dms:*:*:task:*"
            ],
            "Condition": {
                "StringEqualsIgnoreCase": {
                    "aws:ResourceTag/ResourceCreatedBy": "DMSServerless"
                }
            }
        },
        {
            "Sid": "id3",
            "Effect": "Allow",
            "Action": [
                "dms:TestConnection",
                "dms:DeleteConnection"
            ],
            "Resource": [
                "arn:aws:dms:*:*:rep:*",
                "arn:aws:dms:*:*:endpoint:*"
            ]
        }
    ]
}
```

ユーザー、グループ、ロールなどの IAM エンティティがサービスにリンクされたロールを作成、編集、削除できるようにするには、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 User Guide」(IAM ユーザーガイド) の「Service-linked role permissions」(サービスリンクロールのアクセス権限) を参照してください。

AWS DMS Serverless のサービスにリンクされたロールの作成

レプリケーションを作成すると、 AWS DMS AWS DMS サーバーレスはサーバーレスサービスにリンクされたロールをプログラム的に作成します。このロールは IAM コンソールで確認できます。このロールは手動で作成することもできます。ロールを手動で作成するには、IAM コンソールを使用して DMS ユースケースのサービスにリンクされたロールを作成します。 AWS CLI または AWS API で、サービス名に for を使用してサービスにリンクされたロールを作成します。dms.amazonaws.com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作成」を参照してください。このサービスにリンクされたロールを削除しても、この同じプロセスを使用して、もう一度ロールを作成できます。

> **Note**
>
> アカウントにレプリケーションがある場合にロールを削除すると、レプリケーションは失敗します。

AWS DMS Serverless のサービスにリンクされたロールの編集

AWS DMS AWSServiceRoleForDMSServerless サービスにリンクされたロールは編集できません。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「サービスリンクロールの編集」を参照してください。

AWS DMS Serverless のサービスにリンクされたロールの削除

サービスリンクロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。これにより、アクティブにモニタリングまたはメンテナンスされない未使用のエンティティを低減できます。ただし、手動で削除する前に、サービスリンクロールのリソースをクリーンアップする必要があります。

> **Note**
>
> AWS DMS サービスがそのロールを使用しているときにリソースを削除しようとすると、削除が失敗する可能性があります。失敗した場合は、数分待ってから操作を再試行してください。

AWS DMS が使用しているリソースを削除するには AWSServiceRoleForDMSServerless

1. AWS Management Console にサインインし、https://console.aws.amazon.com/dms/v2/ AWS DMS のコンソールを開きます。
2. ナビゲーションペインで、[検出] の下にある [サーバーレス] を選択します。[サーバーレス] ページが開きます。
3. サーバーレスレプリケーションを選択して、[削除] をクリックします。
4. 削除を確認するには、テキスト入力フィールドにサーバーレスレプリケーション名を入力します。次に、[削除] を選択します。

すべてのサーバーレスレプリケーションを削除したら、サービスにリンクされたロールを削除できます。

サービスにリンクされたロールを IAM で手動削除するには

IAM コンソール、 AWS CLI、または AWS API を使用して、 AWSServiceRoleForDMSServerless サービスにリンクされたロールを削除します。詳細については、IAM ユーザーガイドの「サービスリンクロールの削除」を参照してください。

AWS DMS Serverless のサービスにリンクされたロールをサポートするリージョン

AWS DMS サーバーレスでは、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を使用できます。

AWS Database Migration Service ID とアクセスのトラブルシューティング

次の情報は、 と IAM の使用時に発生する可能性がある一般的な問題の診断 AWS DMS と修正に役立ちます。

トピック

- でアクションを実行する権限がない AWS DMS
- iam を実行する権限がありません。PassRole
- 管理者として他のユーザーにアクセスを許可したい AWS DMS
- 自分の AWS アカウント以外のユーザーに自分の AWS DMS リソースへのアクセスを許可したい

でアクションを実行する権限がない AWS DMS

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。担当の管理者はお客様のユーザー名とパスワードを発行した人です。

次の例のエラーは、IAM mateojackson ユーザーがコンソールを使用して AWS DMS エンドポイントの詳細を表示しようとしているが、 アクセスdms: DescribeEndpoint許可がない場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 dms:DescribeEndpoint on resource: my-postgresql-target
```

この場合、Mateo は、dms:DescribeEndpoint アクションを使用して my-postgresql-target エンドポイントにアクセスできるように、ポリシーの更新を管理者に依頼します。

iam を実行する権限がありません。PassRole

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して AWS DMSにロールを渡すことができるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、marymajor という IAM ユーザーがコンソールを使用して AWS DMSでアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

管理者として他のユーザーにアクセスを許可したい AWS DMS

他のユーザーが にアクセスできるようにするには AWS DMS、アクセスが必要なユーザーまたはアプリケーションの IAM エンティティ (ユーザーまたはロール) を作成する必要があります。ユーザーは、このエンティティの認証情報を使用して AWSにアクセスします。次に、 AWS DMSの適切なアクセス許可を付与するポリシーを、そのエンティティにアタッチする必要があります。

すぐに開始するには、IAM ユーザーガイドの「IAM が委任した最初のユーザーおよびグループの作成」を参照してください。

自分の AWS アカウント以外のユーザーに自分の AWS DMS リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- がこれらの機能 AWS DMS をサポートしているかどうかを確認するには、「」を参照してくださいが IAM と AWS Database Migration Service 連携する方法。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティー に提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイドの外部認証されたユーザーへのアクセスの提供 (ID フェデレーション) を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いについては、IAM ユーザーガイドの「IAM でのクロスアカウントリソースアクセス」を参照してください。

AWS DMSの使用に必要な IAM アクセス許可

AWS DMSを使用するには、特定の IAM アクセス許可と IAM ロールを使用します。IAM ユーザーとしてサインインしていて、 を使用する場合は AWS DMS、アカウント管理者が、このセクションで説明するポリシーを、 の実行に使用する IAM ユーザー、グループ、またはロールにアタッチする必要があります AWS DMS。IAM アクセス許可の詳細については、『IAM ユーザーガイド』をご参照ください。

次のポリシーでは AWS DMS、 へのアクセスと、、IAM、Amazon EC2 AWS KMS、Amazon などの他の Amazon サービスから必要な特定のアクションに対するアクセス許可を付与します CloudWatch。 CloudWatch は、 AWS DMS 移行をリアルタイムでモニタリングし、移行の進行状況を示すメトリクスを収集および追跡します。 CloudWatch ログを使用して、タスクに関する問題をデバッグできます。

> **Note**
>
> タグ付けを使用して AWS DMS リソースへのアクセスをさらに制限できます。タグ付けを使用して AWS DMS リソースへのアクセスを制限する方法の詳細については、「」を参照してくださいリソース名とタグを使用したファイングレインアクセスコントロール。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dms:*",
            "Resource": "arn:aws:dms:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "iam:GetRole",
                "iam:PassRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVpcs",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:Get*",
                "cloudwatch:List*"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:FilterLogEvents",
                "logs:GetLogEvents"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        }
    ]
}
```

これらの以下のアクセス許可明細は、それぞれのアクセス許可が必要な理由を理解するうえで役立ちます。

ユーザーが AWS DMS API オペレーションを呼び出せるようにするには、次のセクションが必要です。

```
{
            "Effect": "Allow",
            "Action": "dms:*",
            "Resource": "arn:aws:dms:region:account:resourcetype/id"
}
```

次のセクションは、コンソールで表示するために使用可能な AWS KMS キーとエイリアスをユーザーが一覧表示できるようにするために必要です。KMS キーの Amazon リソースネーム (ARN) がわかっていて、 AWS Command Line Interface () のみを使用している場合、このエントリは必要ありませんAWS CLI。

```
{
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        }
```

次のセクションは、エンドポイントとともに IAM ロールの ARN を渡す必要がある特定のエンドポイントタイプに必要になります。さらに、必要な AWS DMS ロールが事前に作成されていない場合、AWS DMS コンソールでロールを作成できます。すべてのロールが事前に設定されている場合、必要なものは `iam:GetRole` および `iam:PassRole` のみです。ロールの詳細については、「AWS CLI および AWS DMS API で使用する IAM ロールの作成」をご参照ください。

```
{
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:PassRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
```

```
        }
```

は Amazon EC2 インスタンスを作成し、作成されたレプリケーションインスタンスのネットワークを設定する AWS DMS 必要があるため、次のセクションは必須です。これらのリソースはお客様のアカウント内に存在するため、お客様に代わってこれらのアクションを実行できる必要があります。

```
{
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVpcs",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        }
```

次のセクションは、ユーザーがレプリケーション インスタンスのメトリクスを表示することを許可するために必要です。

```
{
            "Effect": "Allow",
            "Action": [
                "cloudwatch:Get*",
                "cloudwatch:List*"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        }
```

このセクションは、ユーザーがレプリケーションログを表示することを許可するために必要です。

```
{
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:FilterLogEvents",
```

```
                "logs:GetLogEvents"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        }
```

AWS DMS コンソールを使用すると、 AWS アカウントに自動的にアタッチされるロールが AWS DMS いくつか作成されます。移行に AWS Command Line Interface （ AWS CLI) または AWS DMS API を使用する場合は、これらのロールをアカウントに追加する必要があります。これらのロールの追加についての詳細は、「AWS CLI および AWS DMS API で使用する IAM ロールの作成」をご参照ください。

AWS CLI および AWS DMS API で使用する IAM ロールの作成

データベース移行に AWS CLI または AWS DMS API を使用する場合は、 の機能を使用する前に、AWS アカウントに 3 つの IAM ロールを追加する必要があります AWS DMS。これらのロールのうち 2 つは dms-vpc-role と dms-cloudwatch-logs-role です。Amazon Redshift をターゲットデータベースとして使用する場合は、dms-access-for-endpoint AWS アカウントに IAM ロールも追加する必要があります。

管理ポリシーの更新は自動です。IAM ロールでカスタムポリシーを使用する場合、このドキュメントで管理ポリシーの更新事項がないか定期的に確認してください。管理ポリシーの詳細は、get-policy コマンドと get-policy-version コマンドを組み合わせて使用して表示できます。

たとえば、次の get-policy コマンドは、指定された IAM ロールに関する情報を取得します。

```
aws iam get-policy --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonDMSVPCManagementRole
```

コマンドから返される情報は、次のとおりです。

```
{
    "Policy": {
        "PolicyName": "AmazonDMSVPCManagementRole",
        "Description": "Provides access to manage VPC settings for AWS managed customer
 configurations",
        "CreateDate": "2015-11-18T16:33:19Z",
        "AttachmentCount": 1,
        "IsAttachable": true,
        "PolicyId": "ANPAJHKIGMBQI4AEFFSYO",
        "DefaultVersionId": "v3",
        "Path": "/service-role/",
        "Arn": "arn:aws:iam::aws:policy/service-role/AmazonDMSVPCManagementRole",
        "UpdateDate": "2016-05-23T16:29:57Z"
    }
}
```

次の get-policy-version コマンドは、IAM ポリシー情報を取得します。

```
aws iam get-policy-version --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonDMSVPCManagementRole --version-id v3
```

コマンドから返される情報は、次のとおりです。

```
{
    "PolicyVersion": {
        "CreateDate": "2016-05-23T16:29:57Z",
        "VersionId": "v3",
        "Document": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": [
                    "ec2:CreateNetworkInterface",
                    "ec2:DescribeAvailabilityZones",
                    "ec2:DescribeInternetGateways",
                    "ec2:DescribeSecurityGroups",
                    "ec2:DescribeSubnets",
                    "ec2:DescribeVpcs",
                    "ec2:DeleteNetworkInterface",
                    "ec2:ModifyNetworkInterfaceAttribute"
                ],
                "Resource": "arn:aws:service:region:account:resourcetype/id",
                "Effect": "Allow"
            }
        ]
    },
    "IsDefaultVersion": true
    }
}
```

同じコマンドを使用して、AmazonDMSRedshiftS3Role および AmazonDMSCloudWatchLogsRole 管理ポリシーに関する情報を取得できます。

> **Note**
>
> データベース移行に AWS DMS コンソールを使用する場合、これらのロールは自動的に AWS アカウントに追加されます。

次の手順では、dms-vpc-role、dms-cloudwatch-logs-role、および dms-access-for-endpoint の各 IAM ロールを作成します。

AWS CLI または AWS DMS API で使用する dms-vpc-role IAM ロールを作成するには

1. 次の IAM ポリシーを含む JSON ファイルを作成します。JSON ファイルに dmsAssumeRolePolicyDocument.json という名前を付けます。

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

   次のコマンドを使用して、 AWS CLI を使用してロールを作成します。

```
aws iam create-role --role-name dms-vpc-role --assume-role-policy-document file://
dmsAssumeRolePolicyDocument.json
```

2. 次のコマンドを使用して AmazonDMSVPCManagementRole ポリシーを dms-vpc-role にアタッチします。

```
aws iam attach-role-policy --role-name dms-vpc-role --policy-arn
 arn:aws:iam::aws:policy/service-role/AmazonDMSVPCManagementRole
```

AWS CLI または AWS DMS API で使用する dms-cloudwatch-logs-role IAM ロールを作成するには

1. 次の IAM ポリシーを含む JSON ファイルを作成します。JSON ファイルに dmsAssumeRolePolicyDocument2.json という名前を付けます。

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

   次のコマンドを使用して、 AWS CLI を使用してロールを作成します。

```
aws iam create-role --role-name dms-cloudwatch-logs-role --assume-role-policy-
document file://dmsAssumeRolePolicyDocument2.json
```

2. 次のコマンドを使用して AmazonDMSCloudWatchLogsRole ポリシーを dms-cloudwatch-logs-role にアタッチします。

```
aws iam attach-role-policy --role-name dms-cloudwatch-logs-role --policy-arn
 arn:aws:iam::aws:policy/service-role/AmazonDMSCloudWatchLogsRole
```

Amazon Redshift をターゲットデータベースとして使用する場合は、IAM ロール dms-access-for-endpoint を作成して Amazon S3 へのアクセスを可能にする必要があります。

Amazon Redshift をターゲットデータベースとして使用する dms-access-for-endpoint IAM ロールを作成するには

1. 次の IAM ポリシーを含む JSON ファイルを作成します。JSON ファイルに dmsAssumeRolePolicyDocument3.json という名前を付けます。

```
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Sid": "1",
     "Effect": "Allow",
     "Principal": {
       "Service": "dms.amazonaws.com"
     },
     "Action": "sts:AssumeRole"
   },
   {
     "Sid": "2",
     "Effect": "Allow",
     "Principal": {
       "Service": "redshift.amazonaws.com"
     },
     "Action": "sts:AssumeRole"
   }
 ]
}
```

2. 次のコマンドを使用して、 AWS CLI を使用してロールを作成します。

```
  aws iam create-role --role-name dms-access-for-endpoint --assume-role-policy-
document file://dmsAssumeRolePolicyDocument3.json
```

3. 次のコマンドを使用して AmazonDMSRedshiftS3Role ポリシーを dms-access-for-endpoint ロールにアタッチします。

```
aws iam attach-role-policy --role-name dms-access-for-endpoint \
    --policy-arn arn:aws:iam::aws:policy/service-role/AmazonDMSRedshiftS3Role
```

これで、 AWS CLI または AWS DMS API を使用するための IAM ポリシーが設定されているはずです。

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、混乱した代理人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。

aws:SourceArnaws:SourceAccountリソースポリシーではグローバル条件コンテキストキーとグローバル条件コンテキストキーを使用して、 AWS Database Migration Service リソースに別のサービスを付与する権限を制限することをおすすめします。aws:SourceArn 値に AWS DMS レプリケーションインスタンス名 (ARN) などのアカウント ID が含まれていない場合は、両方のグローバル条件コンテキストキーを使用してアクセス権限を制限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。

AWS DMS 3.4.7 以降のバージョンでは、混乱した副オプションをサポートするようになりました。詳細については、「AWS Database Migration Service 3.4.7 リリースノート」を参照してください。レプリケーションインスタンスが AWS DMS 3.4.6 以前のバージョンを使用している場合は、混乱した代理オプションを設定する前に必ず最新バージョンにアップグレードしてください。

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には、グローバルコンテキスト条件キー aws:SourceArn で、ARN の未知部分を示すためにワイルドカード文字 (*) を使用します。例えば、arn:aws:dms:*:*123456789012*:rep:* です。

トピック

- AWS DMS API と併用する IAM ロール (サービス間の混乱した代理人防止)
- サービス間の混乱を防ぐために Amazon S3 にプリフライト評価を保存する IAM ポリシー

- Amazon DynamoDB をターゲットエンドポイントとして使用し、 AWS DMS クロスサービスの混乱副防止を実現

AWS DMS API と併用する IAM ロール (サービス間の混乱した代理人防止)

AWS CLI または AWS DMS API をデータベース移行に使用するには、の機能を使用する前に、dms-vpc-roleおよび dms-cloudwatch-logs-role IAM AWS ロールをアカウントに追加する必要があります。 AWS DMS詳細については、「AWS CLI および AWS DMS API で使用する IAM ロールの作成」を参照してください。

次の例は、my-replication-instance レプリケーションインスタンスで dms-vpc-role ロールを使用する際のポリシーを示しています。これらのポリシーを使用して、混乱した代理問題を防止します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "dms.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "your_account_id"
                },
                "ArnEqual": {
                    "AWS:SourceArn": "arn:aws:dms:your_region:your_account_id:rep:my-
replication-instance"
                }
            }
        }
    ]
}
```

サービス間の混乱を防ぐために Amazon S3 にプリフライト評価を保存する IAM ポリシー

事前評価の結果を S3 バケットに保存するには、Amazon S3 のオブジェクトの管理を許可する AWS DMS IAM ポリシーを作成します。詳細については、「IAM リソースを作成する」を参照してください。

以下の例は、 AWS DMS 指定されたユーザーアカウントでのすべてのタスクと評価実行へのアクセスを許可する IAM ロールに設定された、混乱した副条件を含む信頼ポリシーを示しています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "dms.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "your_account_id"
                },
                "ArnLike": {
                    "AWS:SourceArn": [
                        "arn:aws:dms:your_region:your_account_id:assessment-run:*",
                        "arn:aws:dms:region:your_account_id:task:*"
                    ]
                }
            }
        }
    ]
}
```

Amazon DynamoDB をターゲットエンドポイントとして使用し、 AWS DMS クロスサービスの混乱副防止を実現

Amazon DynamoDB をデータベース移行のターゲットエンドポイントとして使用するには、DynamoDB テーブルへのアクセスを引き受けて許可する IAM ロールを作成する必要があります。 AWS DMS その後、 AWS DMSでターゲットの DynamoDB エンドポイントを作成する際に、

このロールを使用します。詳細については、「ターゲットとしての Amazon DynamoDB の使用」を参照してください。

次の例は、 AWS DMS すべてのエンドポイントに DynamoDB テーブルへのアクセスを許可する IAM ロールに設定された、混乱した副条件を含む信頼ポリシーを示しています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "dms.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "your_account_id"
                },
                "ArnLike": {
                    "AWS:SourceArn":
 "arn:aws:dms:your_region:your_account_id:endpoint:*"
                }
            }
        }
    ]
}
```

AWS の マネージドポリシー AWS Database Migration Service

トピック

- AWS マネージドポリシー: AmazonDMSVPCManagementRole
- AWS 管理ポリシー： AWSDMSServerlessServiceRolePolicy
- AWS マネージドポリシー: AmazonDMSCloudWatchLogsRole
- AWS マネージドポリシー： AWSDMSFleetAdvisorServiceRolePolicy
- AWS DMSAWS 管理ポリシーの更新

AWS マネージドポリシー: AmazonDMSVPCManagementRole

このポリシーは dms-vpc-roleロールにアタッチされ、 AWS DMS がユーザーに代わってアクションを実行できるようにします。

このポリシーは、 がネットワークリソースを管理できるようにする権限 AWS DMS を寄稿者に付与します。

許可の詳細

このポリシーには、次のオペレーションが含まれます。

- ec2:CreateNetworkInterface - ネットワークインターフェイスを作成するには、このアクセス許可 AWS DMS が必要です。これらのインターフェイスは、 AWS DMS レプリケーション インスタンスがソースデータベースとターゲットデータベースに接続するために不可欠です。
- ec2:DescribeAvailabilityZones – このアクセス許可により AWS DMS 、 はリージョンのアベイラビリティーゾーンに関する情報を取得できます。 はこの情報 AWS DMS を使用して、冗長性と可用性のために正しいゾーンにリソースをプロビジョニングします。
- ec2:DescribeInternetGateways – VPC で設定されたインターネットゲートウェイを理解するために、このアクセス許可が必要になる AWS DMS 場合があります。この情報は、レプリケーション インスタンスまたはデータベースがインターネットアクセスを必要とする場合に重要です。
- ec2:DescribeSecurityGroups – セキュリティグループは、インスタンスとリソースへのインバウンドトラフィックとアウトバウンドトラフィックを制御します。 AWS DMS は、ネットワークインターフェイスを正しく設定し、レプリケーションインスタンスとデータベース間の適切な通信を確保するために、セキュリティグループを記述する必要があります。
- ec2:DescribeSubnets – このアクセス許可により AWS DMS 、 は VPC 内のサブネットを一覧表示できます。 はこの情報 AWS DMS を使用して適切なサブネットでレプリケーションインスタンスを起動し、必要なネットワーク接続を確保します。
- ec2:DescribeVpcs – レプリケーションインスタンスとデータベースが存在するネットワーク環境を理解する AWS DMS には、VPCs の説明が不可欠です。これには、CIDR ブロックやその他の VPC 固有の設定の把握が含まれます。
- ec2:DeleteNetworkInterface - 不要になったネットワークインターフェイスをクリーンアップするには、このアクセス許可 AWS DMS が必要です。これにより、リソース管理と不要なコストの回避に役立ちます。

- ec2:ModifyNetworkInterfaceAttribute – このアクセス許可は、 が管理するネットワークインターフェイスの属性を変更 AWS DMS するために必要です。これには、接続とセキュリティを確保するための設定の調整が含まれる場合があります。
- ec2:DescribeDhcpOptions – 指定された VPC の DHCP オプションセットの詳細 AWS DMS を取得します。この情報は、レプリケーションインスタンスのネットワークを正しく設定するために必要です。
- ec2:DescribeNetworkInterfaces – VPC 内の既存のネットワークインターフェイスに関する情報 AWS DMS を取得します。この情報は、 がネットワークインターフェイスを正しく AWS DMS 設定し、移行プロセスに適したネットワーク接続を確保するために必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterface",
    "ec2:DescribeAvailabilityZones",
    "ec2:DescribeInternetGateways",
    "ec2:DescribeSecurityGroups",
    "ec2:DescribeSubnets",
    "ec2:DescribeVpcs",
    "ec2:DeleteNetworkInterface",
    "ec2:ModifyNetworkInterfaceAttribute",
    "ec2:DescribeDhcpOptions",
    "ec2:DescribeNetworkInterfaces"
   ],
   "Resource": "*"
  }
    ]
}
```

AWS 管理ポリシー： AWSDMSServerlessServiceRolePolicy

このポリシーは AWSServiceRoleForDMSServerlessロールにアタッチされ、 AWS DMS がユーザーに代わってアクションを実行できるようにします。詳細については、「AWS DMS Serverless のサービスにリンクされたロール」を参照してください。

このポリシーは、 がレプリケーションリソースを管理できるようにする権限 AWS DMS を寄稿者に付与します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- dms – プリンシパルが AWS DMS リソースとやり取りできるようにします。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "id0",
            "Effect": "Allow",
            "Action": [
                "dms:CreateReplicationInstance",
                "dms:CreateReplicationTask"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "dms:req-tag/ResourceCreatedBy": "DMSServerless"
                }
            }
        },
        {
            "Sid": "id1",
            "Effect": "Allow",
            "Action": [
                "dms:DescribeReplicationInstances",
                "dms:DescribeReplicationTasks"
            ],
            "Resource": "*"
        },
        {
            "Sid": "id2",
            "Effect": "Allow",
            "Action": [
                "dms:StartReplicationTask",
                "dms:StopReplicationTask",
                "dms:DeleteReplicationTask",
                "dms:DeleteReplicationInstance"
            ],
            "Resource": [
                "arn:aws:dms:*:*:rep:*",
```

```
                "arn:aws:dms:*:*:task:*"
            ],
            "Condition": {
                "StringEqualsIgnoreCase": {
                    "aws:ResourceTag/ResourceCreatedBy": "DMSServerless"
                }
            }
        },
        {
            "Sid": "id3",
            "Effect": "Allow",
            "Action": [
                "dms:TestConnection",
                "dms:DeleteConnection"
            ],
            "Resource": [
                "arn:aws:dms:*:*:rep:*",
                "arn:aws:dms:*:*:endpoint:*"
            ]
        }
    ]
}
```

AWS マネージドポリシー: AmazonDMSCloudWatchLogsRole

このポリシーは dms-cloudwatch-logs-roleロールにアタッチされ、 AWS DMS がユーザーに代わってアクションを実行できるようにします。詳細については、「AWS DMS のサービスにリンクされたロールの使用」を参照してください。

このポリシーは、 がレプリケーションログ AWS DMS を CloudWatch ログに発行できるようにする権限を寄稿者に付与します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- logs – プリンシパルがログを CloudWatch ログに発行できるようにします。このアクセス許可は、 が AWS DMS を使用してレプリケーションログを表示 CloudWatch できるようにするために必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDescribeOnAllLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "AllowDescribeOfAllLogStreamsOnDmsTasksLogGroup",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:dms-tasks-*",
                "arn:aws:logs:*:*:log-group:dms-serverless-replication-*"
            ]
        },
        {
            "Sid": "AllowCreationOfDmsLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:dms-tasks-*",
                "arn:aws:logs:*:*:log-group:dms-serverless-replication-*:log-stream:"
            ]
        },
        {
            "Sid": "AllowCreationOfDmsLogStream",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:dms-tasks-*:log-stream:dms-task-*",
```

```
                "arn:aws:logs:*:*:log-group:dms-serverless-replication-*:log-
stream:dms-serverless-*"
            ]
        },
        {
            "Sid": "AllowUploadOfLogEventsToDmsLogStream",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:dms-tasks-*:log-stream:dms-task-*",
                "arn:aws:logs:*:*:log-group:dms-serverless-replication-*:log-
stream:dms-serverless-*"
            ]
        }
    ]
}
```

AWS マネージドポリシー： AWSDMSFleetAdvisorServiceRolePolicy

IAM エンティティ AWSDMSFleetAdvisorServiceRolePolicy に をアタッチすることはできません。このポリシーは、 AWS DMS Fleet Advisor 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アタッチされます。詳細については、「AWS DMS のサービスにリンクされたロールの使用」を参照してください。

このポリシーは、 AWS DMS Fleet Advisor が Amazon CloudWatch メトリクスを発行できるようにする寄稿者アクセス許可を付与します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `cloudwatch` — プリンシパルがメトリクスデータポイントを Amazon に発行できるようにします CloudWatch。このアクセス許可は、 AWS DMS Fleet Advisor が を使用してデータベースメトリクスを含むグラフを表示 CloudWatch できるようにするために必要です。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Resource": "*",
        "Action": "cloudwatch:PutMetricData",
        "Condition": {
            "StringEquals": {
                "cloudwatch:namespace": "AWS/DMS/FleetAdvisor"
            }
        }
    }
}
```

AWS DMSAWS 管理ポリシーの更新

このサービスがこれらの変更の追跡を開始した AWS DMS 以降の の AWS マネージドポリシーの更新に関する詳細を表示します。このページの変更に関する自動アラートを受け取るには、 AWS DMS ドキュメント履歴ページの RSS フィードにサブスクライブしてください。

変更	説明	日付
AmazonDMSVPCManagementRole – 変更	AWS DMS がユーザーに代わってネットワーク設定を管理 AWS DMS できるようにする `ec2:DescribeDhcpOptions` および `ec2:DescribeNetworkInterfaces` オペレーションが追加されました。	2024 年 6 月 17 日
AWSDMSServerlessServiceRolePolicy - 新しいポリシー	AWS DMS は、Amazon CloudWatch メトリクスの公開など、ユーザーに代わってがサービスを作成および管理 AWS DMS できるようにす	2023 年 5 月 22 日

変更	説明	日付
	る AWSDMSServerlessServiceRolePolicy ロールを追加しました。	
AmazonDMSCloudWatchLogsRole – 変更	AWS DMS は、サーバーレスリソースの ARN を、付与された各アクセス許可に追加し、サーバーレス AWS DMS レプリケーション設定から Logs へのレプリケーション CloudWatch ログのアップロードを許可しました。	2023 年 5 月 22 日
AWSDMSFleetAdvisorServiceRolePolicy - 新しいポリシー	AWS DMS Fleet Advisor は、メトリクスデータポイントを Amazon に公開できるようにする新しいポリシーを追加しました CloudWatch。	2023 年 3 月 6 日
AWS DMS が変更の追跡を開始しました	AWS DMS が AWS マネージドポリシーの変更の追跡を開始しました。	2023 年 3 月 6 日

AWS Database Migration Service のコンプライアンス検証

第三者監査人が、複数の AWS コンプライアンスプログラムの一環として AWS Database Migration Service のセキュリティとコンプライアンスを評価します。これには次が対象となります。

- SOC
- PCI
- ISO 規格
- FedRAMP
- DoD CC SRG
- HIPAA BAA
- MTCS
- CS
- K-ISMS
- ENS High
- OSPAR
- HITRUST CSF

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 AWS のサービス」を参照してください。一般的な情報にについては、「AWS コンプライアンスプログラム」を参照してください。

AWS Artifact を使用して第三者監査人の監査レポートをダウンロードできます。詳細については、「Downloading reports in AWS artifact」を参照してください。

AWS DMS 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、企業のコンプライアンス目標、適用法と規制によって定まります。AWS は、コンプライアンスに役立つ次のリソースを提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – このようなデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、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 AWS で導入するための手順を提供しています。
- Architecting for HIPAA security and compliance on Amazon Web Services 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ています。

- AWS コンプライアンスのリソース – 業界と地域別のワークブックとガイドのコレクションが提供されています。
- AWS Config – この AWS サービスは、リソースの設定が社内慣行、業界ガイドライン、規制にどの程度準拠しているかを評価します。
- AWS Security Hub - この AWS サービスは、AWS 内のセキュリティ状態の包括的なビューを提供し、セキュリティ上の業界標準とベストプラクティスへの準拠の確認に役立ちます。

AWS Database Migration Service での耐障害性

AWS のグローバルインフラストラクチャは AWS リージョンとアベイラビリティーゾーンを中心に構築されます。AWSリージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立し隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、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、耐障害性、および拡張性が優れています。

AWS の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」をご参照ください。

AWSグローバルインフラストラクチャに加え、AWS DMSは、Multi-AZ Multi-AZオプションを選択した場合、マルチ AZ 配置を使用するレプリケーション インスタンスに高可用性およびフェイルオーバーにも対応します。

マルチ AZ 配置では、AWS DMS は異なるアベイラビリティーゾーンにレプリケーション インスタンスのスタンバイ レプリカを自動的にプロビジョニングして維持します。プライマリレプリケーション インスタンスは、同期的にスタンバイレプリカにレプリケートされます。プライマリレプリケーション インスタンスに障害が発生するか、応答しない場合、スタンバイ状態で中断時間をできる限り抑えて、実行中のタスクを再開します。プライマリはその状態を常にスタンバイにレプリケーションしているため、マルチ AZ 配置ではパフォーマンス上のオーバーヘッドが発生します。

マルチ AZ 配置の使用については、「AWS DMS レプリケーションインスタンスの使用」をご参照ください。

AWS Database Migration Service でのインフラストラクチャのセキュリティ

マネージドサービスである AWS Database Migration Service は AWS グローバルネットワークセキュリティで保護されています。AWS セキュリティサービスと AWS がインフラストラクチャを保護する方法については、「AWS クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「セキュリティの柱 - AWS Well-Architected Framework」の「インフラストラクチャ保護」を参照してください。

AWS の公開 API コールを使用して、ネットワーク経由で AWS DMS にアクセスします。クライアントは次をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 は必要であり、TLS 1.3 をお勧めします。
- DHE (Ephemeral Diffie-Hellman) や ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) などの Perfect Forward Secrecy (PFS) を備えた暗号スイート。Java 7 以降などのほとんどの最新のシステムは、これらのモードをサポートしています。

さらに、リクエストは、アクセス キー ID と、IAM プリンシパルに関連付けられたシークレットアクセスキーを使用して署名されている必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時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。

このような API オペレーションはネットワークの場所を問わず呼び出すことができます。ただし、AWS DMS はリソースベースのアクセスポリシーもサポートしており、例えば、送信元 IP アドレスに応じてアクションを制限できます。また、AWS DMS ポリシーを使用して、特定の Amazon VPC エンドポイントまたは特定の仮想プライベートクラウド (VPC) からのアクセスを制御することもできます。これにより、実質的に AWS ネットワーク内の特定の VPC からの特定の AWS DMS リソースへのネットワークアクセスは分離されます。AWS DMS でリソースベースのアクセスポリシーを使用する方法の詳細については、「リソース名とタグを使用したファイングレインアクセスコントロール」を参照してください。

AWS DMS との通信を単一の VPC 内に限定するには、AWS PrivateLink を介して AWS DMS に接続できる VPC インターフェイスエンドポイントを作成します。AWS PrivateLink は AWS DMS への呼び出しと関連する結果がインターフェイス エンドポイントが作成されている特定の VPC に限定されるうえで役立ちます。その後、このインターフェイスエンドポイントの URL を AWS CLI または SDK を使用して実行する、すべての AWS DMS コマンドでオプションとして指定できます。これにより、AWS DMS との通信全体が VPC に限定されたままとなり、それ以外はパブリックインターネットから見えなくなります。

単一の VPC 内の DMS にアクセスするためのインターフェイスエンドポイントを作成するには

1. AWS Management Console にサインインして、https://console.aws.amazon.com/vpc/ で Amazon VPC コンソールを開きます。
2. ナビゲーションペインから [エンドポイント] を選択します。これにより、[エンドポイントの作成] ページが開き、VPC から AWS DMS にインターフェイスエンドポイントを作成できます。
3. [AWS サービス] を選択してから、[サービス名] の値を検索して選択します。この場合、次のフォームで AWS DMS となります。

   ```
   com.amazonaws.region.dms
   ```

   ここで、*region* には、例えば com.amazonaws.us-west-2.dms などの AWS DMS が実行されている AWS リージョンを指定します。
4. [VPC] では、例えば vpc-12abcd34 など、VPC から作成するインターフェイスエンドポイントを選択します。
5. [アベイラビリティーゾーン] と [サブネット ID] の値を選択します。上記の値は、例えば us-west-2a（usw2-az1）や subnet-ab123cd4 など、選択した AWS DMS エンドポイントが実行できる場所を示す必要があります。
6. [DNS 名を有効化] を選択して、DNS 名でエンドポイントを作成します。この DNS 名は、ランダムな文字列 (ab12dc34) とハイフンでつながれたエンドポイント ID (vpce-12abcd34efg567hij) で構成されます。上記は、ドットで区切られ、逆の順序のドットで追加された vpce (dms.us-west-2.vpce.amazonaws.com) でサービス名から区切られます。

   例えば、vpce-12abcd34efg567hij-ab12dc34.dms.us-west-2.vpce.amazonaws.com となります。
7. [セキュリティグループ] では、エンドポイントに使用するグループを選択します。

   セキュリティグループを設定する際、そのグループ内からのアウトバウンド HTTPS コールを必ず許可します。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の作成」を参照してください。
8. [ポリシー] では、[フルアクセス] またはカスタム値を選択します。例えば次のとおり、エンドポイントの特定のアクションとリソースへのアクセスを制限するカスタムポリシーを選択できます。

   ```
   {
   ```

```
  "Statement": [
    {
      "Action": "dms:*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": [
        "dms:ModifyReplicationInstance",
        "dms:DeleteReplicationInstance"
      ],
      "Effect": "Deny",
      "Resource": "arn:aws:dms:us-west-2:<account-id>:rep:<replication-instance-
id>",
      "Principal": "*"
    }
  ]
}
```

このサンプルポリシーでは、特定のレプリケーションインスタンスの削除または変更を除き すべての AWS DMS API コールを許可します。

これで、ステップ 6 でオプションとして作成した DNS 名を使用して形成された URL を指定できるようになりました。すべての AWS DMS CLI コマンドまたは API オペレーションにこれを指定し、作成したインターフェイスエンドポイントを使用してサービスインスタンスにアクセスします。例えば次のとおり、この VPC で DMS CLI コマンド `DescribeEndpoints` を実行できます。

```
$ aws dms describe-endpoints --endpoint-url https://vpce-12abcd34efg567hij-
ab12dc34.dms.us-west-2.vpce.amazonaws.com
```

プライベート DNS オプションを有効にすると、エンドポイント URL をリクエストに指定する必要はありません。

VPC インターフェイスエンドポイントの作成と使用 (プライベート DNS オプションの有効化など) の詳細については、「Amazon VPC ユーザーガイド」の「インターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。

リソース名とタグを使用したファイングレインアクセスコントロール

Amazon リソースネーム (ARNsに基づくリソース名とリソースタグを使用して、 AWS DMS リソースへのアクセスを管理できます。これを行うには、許可されたアクションを定義するか、条件ステートメントを IAM ポリシーに含めます。

リソース名を使用したアクセスの制御

IAM ユーザーアカウントを作成し、 AWS DMS リソースの ARN に基づいてポリシーを割り当てることができます。

次のポリシーは、ARN arn:aws:dms:us-east-1:152683116:rep:DOH67ZTOXGLIXMIHKITV を持つ AWS DMS レプリケーションインスタンスへのアクセスを拒否します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dms:*"
            ],
            "Effect": "Deny",
            "Resource": "arn:aws:dms:us-east-1:152683116:rep:DOH67ZTOXGLIXMIHKITV"
        }
    ]
}
```

たとえば、このポリシーが有効になっていると、次のコマンドは失敗します。

```
$ aws dms delete-replication-instance
   --replication-instance-arn "arn:aws:dms:us-
east-1:152683116:rep:DOH67ZTOXGLIXMIHKITV"

A client error (AccessDeniedException) occurred when calling the
 DeleteReplicationInstance
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
```

```
dms:DeleteReplicationInstance on resource: arn:aws:dms:us-
east-1:152683116:rep:DOH67ZTOXGLIXMIHKITV

$ aws dms modify-replication-instance
   --replication-instance-arn "arn:aws:dms:us-
east-1:152683116:rep:DOH67ZTOXGLIXMIHKITV"

A client error (AccessDeniedException) occurred when calling the
 ModifyReplicationInstance
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:ModifyReplicationInstance on resource: arn:aws:dms:us-
east-1:152683116:rep:DOH67ZTOXGLIXMIHKITV
```

AWS DMS エンドポイントとレプリケーションタスクへのアクセスを制限する IAM ポリシーを指定することもできます。

次のポリシーは、 AWS DMS エンドポイントの ARN を使用してエンドポイントへのアクセスを制限します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dms:*"
            ],
            "Effect": "Deny",
            "Resource": "arn:aws:dms:us-
east-1:152683116:endpoint:D6E37YBXTNHOA6XRQSZCUGX"
        }
    ]
}
```

たとえば、次のコマンドは、エンドポイントの ARN を使用するポリシーが有効になっていると失敗します。

```
$ aws dms delete-endpoint
   --endpoint-arn "arn:aws:dms:us-east-1:152683116:endpoint:D6E37YBXTNHOA6XRQSZCUGX"
```

```
A client error (AccessDeniedException) occurred when calling the DeleteEndpoint
 operation:
User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
 dms:DeleteEndpoint
on resource: arn:aws:dms:us-east-1:152683116:endpoint:D6E37YBXTNHOA6XRQSZCUGX

$ aws dms modify-endpoint
   --endpoint-arn "arn:aws:dms:us-east-1:152683116:endpoint:D6E37YBXTNHOA6XRQSZCUGX"


A client error (AccessDeniedException) occurred when calling the ModifyEndpoint
 operation:
User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
 dms:ModifyEndpoint
on resource: arn:aws:dms:us-east-1:152683116:endpoint:D6E37YBXTNHOA6XRQSZCUGX
```

次のポリシーは、 AWS DMS タスクの ARN を使用してタスクへのアクセスを制限します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dms:*"
            ],
            "Effect": "Deny",
            "Resource": "arn:aws:dms:us-east-1:152683116:task:UO3YR4N47DXH3ATT4YMWOIT"
        }
    ]
}
```

たとえば、次のコマンドは、タスクの ARN を使用するポリシーが有効になっていると失敗します。

```
$ aws dms delete-replication-task
   --replication-task-arn "arn:aws:dms:us-
east-1:152683116:task:UO3YR4N47DXH3ATT4YMWOIT"
```

```
A client error (AccessDeniedException) occurred when calling the DeleteReplicationTask
 operation:
User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
 dms:DeleteReplicationTask
on resource: arn:aws:dms:us-east-1:152683116:task:UO3YR4N47DXH3ATT4YMWOIT
```

タグを使用したアクセスへのコントロール

AWS DMS は、追加のタグ付け要件なしに、ユーザー定義ポリシーで使用できる共通のキーと値のペアのセットを定義します。 AWS DMS リソースのタグ付けの詳細については、「」を参照してくださいAWS Database Migration Service でのリソースへのタグ付け。

で使用できる標準タグを次に示します AWS DMS。

- aws:CurrentTime – リクエストの日時を表し、時間基準に基づいてアクセスを制限できるようにします。
- aws:EpochTime – このタグは、前の aws:CurrentTime tag に似ていますが、現在の時刻は Unix エポックからの経過秒数として表されます。
- aws: MultiFactorAuthPresent – これは、リクエストが多要素認証によって署名されたかどうかを示すブールタグです。
- aws: MultiFactorAuthAge – 多要素認証トークンの経過時間 (秒単位) へのアクセスを提供します。
- aws:principaltype - 現在のリクエストに対するプリンシパルのタイプ (ユーザー、アカウント、フェデレーティッドユーザーなど) へのアクセスを提供します。
- aws:SourceIp – リクエストを発行するユーザーのソース IP アドレスを表します。
- aws: UserAgent – リソースをリクエストするクライアントアプリケーションに関する情報を提供します。
- aws:userid – リクエストを発行しているユーザーの ID へのアクセスを提供します。
- aws:username – リクエストを発行しているユーザーの名前へのアクセスを提供します。
- dms: InstanceClass – レプリケーションインスタンスホストのコンピューティングサイズへのアクセスを提供します (複数可) 。
- dms: StorageSize – ストレージボリュームサイズ (GB 単位) へのアクセスを提供します。

独自のタグを定義することもできます。カスタマー定義タグは、 AWS タグ付けサービスに保持される単純なキーと値のペアです。このタグを AWS DMS リソース (レプリケーション インスタンス、エンドポイント、タスクを含む) に追加できます。これらのタグはポリシーの IAM「条件」ステート

メントを使用してマッチングされ、特定の条件付きタグを使用して参照されます。タグキーにはプレフィックスとして「dms」、リソースタイプ、および「tag」が付きます。以下にタグ形式を示します。

```
dms:{resource type}-tag/{tag key}={tag value}
```

たとえば、タグ「stage=production」を含むレプリケーション インスタンスに対してのみ API コールの成功を許可するポリシーを定義するとします。次の条件ステートメントは、指定されたタグを持つリソースに一致します。

```
"Condition":
{
    "streq":
        {
            "dms:rep-tag/stage":"production"
        }
}
```

次のタグを、このポリシー条件に一致するレプリケーション インスタンスに追加します。

```
stage production
```

AWS DMS リソースに既に割り当てられているタグに加えて、ポリシーを記述して、特定のリソースに適用できるタグキーと値を制限することもできます。この場合、タグのプレフィックスは「req」です。

たとえば、次のポリシーステートメントは、ユーザーが特定のリソースに割り当てることができるタグを、許可される値の特定のリストに制限します。

```
 "Condition":
{
    "streq":
        {
            "dms:rep-tag/stage": [ "production", "development", "testing" ]
        }
}
```

次のポリシー例では、 AWS DMS リソースタグに基づいて リソースへのアクセスを制限します。

次のポリシーでは、タグの値が「Desktop」、タグキーが「Env」のレプリケーション インスタンスへのアクセスを制限します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dms:*"
            ],
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "dms:rep-tag/Env": [
                        "Desktop"
                    ]
                }
            }
        }
    ]
}
```

次のコマンドは、タグの値が「Desktop」で、タグキーが「Env」の場合にアクセスを制限する IAM ポリシーに基づいて、成功または失敗します。

```
$ aws dms list-tags-for-resource
   --resource-name arn:aws:dms:us-east-1:152683116:rep:46DHOU7JOJYOJXWDOZNFEN
   --endpoint-url http://localhost:8000
{
    "TagList": [
        {
            "Value": "Desktop",
            "Key": "Env"
        }
    ]
}

$ aws dms delete-replication-instance
```

```
   --replication-instance-arn "arn:aws:dms:us-
east-1:152683116:rep:46DHOU7JOJYOJXWDOZNFEN"
A client error (AccessDeniedException) occurred when calling the
 DeleteReplicationInstance
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:DeleteReplicationInstance on resource: arn:aws:dms:us-
east-1:152683116:rep:46DHOU7JOJYOJXWDOZNFEN

$ aws dms modify-replication-instance
   --replication-instance-arn "arn:aws:dms:us-
east-1:152683116:rep:46DHOU7JOJYOJXWDOZNFEN"

A client error (AccessDeniedException) occurred when calling the
 ModifyReplicationInstance
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:ModifyReplicationInstance on resource: arn:aws:dms:us-
east-1:152683116:rep:46DHOU7JOJYOJXWDOZNFEN

$ aws dms add-tags-to-resource
   --resource-name arn:aws:dms:us-east-1:152683116:rep:46DHOU7JOJYOJXWDOZNFEN
   --tags Key=CostCenter,Value=1234

A client error (AccessDeniedException) occurred when calling the AddTagsToResource
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:AddTagsToResource on resource: arn:aws:dms:us-
east-1:152683116:rep:46DHOU7JOJYOJXWDOZNFEN

$ aws dms remove-tags-from-resource
   --resource-name arn:aws:dms:us-east-1:152683116:rep:46DHOU7JOJYOJXWDOZNFEN
   --tag-keys Env

A client error (AccessDeniedException) occurred when calling the
 RemoveTagsFromResource
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:RemoveTagsFromResource on resource: arn:aws:dms:us-
east-1:152683116:rep:46DHOU7JOJYOJXWDOZNFEN
```

次のポリシーは、タグ値が「Desktop」で、タグキーが「Env」である AWS DMS エンドポイントへのアクセスを制限します。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dms:*"
            ],
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "dms:endpoint-tag/Env": [
                        "Desktop"
                    ]
                }
            }
        }
    ]
}
```

次のコマンドは、タグの値が「Desktop」で、タグキーが「Env」の場合にアクセスを制限する IAM ポリシーに基づいて、成功または失敗します。

```
$ aws dms list-tags-for-resource
   --resource-name arn:aws:dms:us-east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I
{
    "TagList": [
        {
            "Value": "Desktop",
            "Key": "Env"
        }
    ]
}

$ aws dms delete-endpoint
   --endpoint-arn "arn:aws:dms:us-east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I"

A client error (AccessDeniedException) occurred when calling the DeleteEndpoint
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:DeleteEndpoint on resource: arn:aws:dms:us-
east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I
```

```
$ aws dms modify-endpoint
   --endpoint-arn "arn:aws:dms:us-east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I"


A client error (AccessDeniedException) occurred when calling the ModifyEndpoint
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:ModifyEndpoint on resource: arn:aws:dms:us-
east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I

$ aws dms add-tags-to-resource
   --resource-name arn:aws:dms:us-east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I
   --tags Key=CostCenter,Value=1234

A client error (AccessDeniedException) occurred when calling the AddTagsToResource
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:AddTagsToResource on resource: arn:aws:dms:us-
east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I

$ aws dms remove-tags-from-resource
   --resource-name arn:aws:dms:us-east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I
   --tag-keys Env

A client error (AccessDeniedException) occurred when calling the
 RemoveTagsFromResource
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:RemoveTagsFromResource on resource: arn:aws:dms:us-
east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I
```

次のポリシーでは、タグの値が「Desktop」、タグキーが「Env」のレプリケーションタスクへのアクセスを制限します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dms:*"
            ],
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {
                    "dms:task-tag/Env": [
                        "Desktop"
                    ]
                }
            }
        }
    ]
}
```

次のコマンドは、タグの値が「Desktop」で、タグキーが「Env」の場合にアクセスを制限する IAM ポリシーに基づいて、成功または失敗します。

```
$ aws dms list-tags-for-resource
   --resource-name arn:aws:dms:us-east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3
{
    "TagList": [
        {
            "Value": "Desktop",
            "Key": "Env"
        }
    ]
}

$ aws dms delete-replication-task
   --replication-task-arn "arn:aws:dms:us-east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3"

A client error (AccessDeniedException) occurred when calling the DeleteReplicationTask
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:DeleteReplicationTask on resource: arn:aws:dms:us-
east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3

$ aws dms add-tags-to-resource
   --resource-name arn:aws:dms:us-east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3
   --tags Key=CostCenter,Value=1234

A client error (AccessDeniedException) occurred when calling the AddTagsToResource
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:AddTagsToResource on resource: arn:aws:dms:us-
east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3
```

```
$ aws dms remove-tags-from-resource
   --resource-name arn:aws:dms:us-east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3
   --tag-keys Env

A client error (AccessDeniedException) occurred when calling the
 RemoveTagsFromResource
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:RemoveTagsFromResource on resource: arn:aws:dms:us-
east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3
```

暗号化キーの設定と AWS KMS アクセス許可の指定

AWS DMS は、レプリケーション インスタンスで使用されるストレージとエンドポイント接続情報を暗号化します。レプリケーション インスタンスが使用するストレージを暗号化するために、 は AWS アカウントに固有の AWS Key Management Service （ AWS KMS) キー AWS DMS を使用します。このキーは で表示および管理できます AWS KMS。アカウント (aws/dms) でデフォルトの KMS キーを使用できます。あるいは、カスタム KMS キーを作成できます。既存の KMS キーがある場合、暗号化にそのキーを使用することもできます。

> **Note**
>
> 暗号化 AWS KMS キーとして使用するカスタムキーまたは既存のキーは、対称キーである必要があります。 は非対称暗号化キーの使用をサポート AWS DMS していません。対称キーと非対称キーの使用詳細については、AWS Key Management Service デベロッパーガイドの「https://docs.aws.amazon.com/kms/latest/developerguide/symmetric-asymmetric.html」をご参照ください。

レプリケーション インスタンスの初回起動時に、 レプリケーション インスタンス作成ページのアドバンストセクションで、カスタム KMS キーを選択していない場合は、デフォルトの KMS キー (aws/dms) が作成されます。デフォルトの KMS キーを使用する場合、移行用の IAM ユーザーアカウントにはアクセス許可として kms:ListAliases と kms:DescribeKey のみを付与する必要があります。デフォルトの KMS キーの使用に関する詳細については、「AWS DMSの使用に必要な IAM アクセス許可」をご参照ください。

カスタム KMS キーを使用するには、次のオプションの 1 つを使用して、カスタム KMS キーにアクセス許可を割り当てます。

- 移行に使用する IAM ユーザーアカウントを AWS KMS 、カスタムキーのキー管理者またはキーユーザーとして追加します。これにより、IAM ユーザーアカウントに必要な AWS KMS 権限が確実に付与されます。このアクションは、 AWS DMSを使用するために IAM ユーザーアカウントに付与する IAM のアクセス許可に追加されます。キーユーザーを許可する詳しい方法については、AWS Key Management Service デベロッパーガイド の「KMS キーの使用をユーザーに許可する」をご参照ください。
- カスタム KMS キーに対するキー管理者あるいはキーユーザーとして IAM ユーザーアカウントを追加したくない場合、 AWS DMSを使用するために IAM ユーザーアカウントに付与する必要がある IAM のアクセス許可に、次のアクセス許可を追加で付与してください。

```
{
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases",
                "kms:DescribeKey",
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:ReEncrypt*"
            ],
            "Resource": "*"
        },
```

AWS DMS は KMS キーエイリアスでも動作します。独自の AWS KMS キーを作成して KMS キーへのユーザーアクセスを許可する方法の詳細については、「AWS KMS デベロッパーガイド」をご参照ください。

KMS キー識別子を指定しない場合、 はデフォルトの暗号化キー AWS DMS を使用します。 は AWS 、アカウントの AWS DMS のデフォルトの暗号化キー AWS KMS を作成します。アカウント AWS には、 AWS リージョンごとに異なるデフォルトの暗号化キーがあります。

AWS DMS リソースの暗号化に使用される AWS KMS キーを管理するには、 を使用します AWS Key Management Service。 は、安全で可用性の高いハードウェアとソフトウェア AWS KMS を組み合わせて、クラウド向けにスケーリングされたキー管理システムを提供します。を使用して AWS KMS、暗号化キーを作成し、これらのキーの使用方法を制御するポリシーを定義できます。

AWS KMS 「」を参照してください。 AWS Management Console

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/kms で AWS Key Management Service（ AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. AWS KMS キーを操作するには、次のいずれかのオプションを選択します。

   - AWS が作成および管理するアカウントのキーを表示するには、ナビゲーションペインでAWS マネージドキー を選択します。
   - ユーザーが作成および管理するアカウント内のキーを表示するには、ナビゲーションペインで [Customer managed keys] (カスタマーマネージドキー) を選択します。

AWS KMS は をサポートしているため AWS CloudTrail、キーの使用状況を監査して、キーが適切に使用されていることを確認できます。 AWS KMS キーは、Amazon RDS、Amazon S3、Amazon Redshift、Amazon EBS などの および AWS DMS サポートされている AWS サービスと組み合わせて使用できます。

カスタム AWS KMS キーを作成して、次の AWS DMS エンドポイントのターゲットデータを暗号化することもできます。

- Amazon Redshift – 詳細については、「Amazon Redshift ターゲットデータを暗号化する AWS KMS キーの作成と使用」をご参照ください。
- Amazon S3 – 詳細については、「Amazon S3 ターゲットオブジェクトを暗号化する AWS KMS キーの作成」をご参照ください。

KMS キーを使用して AWS DMS リソースを作成した後は、それらのリソースの暗号化キーを変更することはできません。 AWS DMS リソースを作成する前に、必ず暗号化キーの要件を確認してください。

のネットワークセキュリティ AWS Database Migration Service

を使用する際に作成するネットワークのセキュリティ要件は、ネットワークの設定方法 AWS Database Migration Service によって異なります。のネットワークセキュリティの一般的なルール AWS DMS は次のとおりです。

- レプリケーション インスタンスは、ソースとターゲットのエンドポイントにアクセスできる必要があります。レプリケーション インスタンスのセキュリティグループには、データベースポートでデータベースエンドポイントへの送信をインスタンスに許可するネットワーク ACL またはルールが必要です。
- データベースエンドポイントには、レプリケーション インスタンスからの受信アクセスを許可するネットワーク ACL およびセキュリティグループルールを含める必要があります。これは、構成に応じて、レプリケーション インスタンスのセキュリティグループ、プライベート IP アドレス、パブリック IP アドレス、または NAT ゲートウェイのパブリックアドレスを使用して実現できます。
- ネットワークで VPN トンネルが使用されている場合、NAT ゲートウェイとして機能する Amazon EC2 インスタンスは、レプリケーション インスタンスにそのゲートウェイを通じたトラフィックの送信を許可するセキュリティグループを使用する必要があります。

デフォルトでは、 AWS DMS レプリケーションインスタンスで使用される VPC セキュリティグループには、すべてのポートで 0.0.0.0/0 への出力を許可するルールがあります。このセキュリティグループを変更するか、独自のセキュリティグループを使用する場合、少なくとも、対応するデータベースポートでソースおよびターゲットエンドポイントへの送信が許可される必要があります。

データベース移行に使用できるネットワーク構成には、それぞれ固有のセキュリティ上の考慮事項があります。

- すべてのデータベース移行コンポーネントが 1 つの VPC にある設定 - エンドポイントで使用されるセキュリティグループは、データベースポートでレプリケーション インスタンスからの進入を許可する必要があります。レプリケーション インスタンスによって使用されるセキュリティグループでエンドポイントに侵入可能なことを確認するかまたは、エンドポイントにより使用されるセキュリティグループに、レプリケーション インスタンスのプライベート IP アドレスにアクセスを許可するセキュリティルールを作成できます。
- 複数の VPC を使用する構成 - レプリケーション インスタンスで使用されるセキュリティグループには、VPC 範囲とデータベースの DB ポートに関するルールが必要です。

- AWS Direct Connect または VPN を使用した VPC へのネットワークの設定 - VPC からオンプレミス VPN へのトンネルに向かうトラフィックを許可する VPN トンネル。この設定では、特定の IP アドレスまたは範囲に向かうトラフィックを、VPC からオンプレミス VPN へのトラフィックをブリッジできるホストに送信するルーティングルールが VPC に含まれています。この場合、レプリケーション インスタンスのプライベート IP アドレスまたはセキュリティグループから NAT インスタンスへのトラフィックを許可する必要がある独自のセキュリティグループ設定が NAT ホストに含まれています。
- インターネットを使用した VPC へのネットワークの設定 - VPC セキュリティグループには、VPC に向かわないトラフィックをインターネットゲートウェイに送信するルーティングルールが含まれている必要があります。この設定では、エンドポイントへの接続がレプリケーション インスタンス上のパブリック IP アドレスから行われているように見えます。
- を使用した VPC 外の RDS DB インスタンスから VPC 内の DB インスタンスへの設定 ClassicLink – ソースまたはターゲットの Amazon RDS DB インスタンスが VPC 内に存在せず、レプリケーションインスタンスが配置されている VPC とセキュリティグループを共有していない場合は、プロキシサーバーを設定し、 ClassicLink を使用してソースデータベースとターゲットデータベースを接続できます。
- ソース エンドポイントがレプリケーション インスタンスで使用されている VPC の外にあり、NAT のゲートウェイを使用している - 単一の Elastic network interface にバインドされた単一の Elastic IP アドレスを使用してネットワークアドレス変換 (NAT) ゲートウェイを設定できます。次に、この Elastic network interface は NAT 識別子 (nat-#####) を受け取ります。インターネットゲートウェイではなくその NAT ゲートウェイへのデフォルトルートが VPC に含まれている場合、レプリケーション インスタンスはインターネットゲートウェイのパブリック IP アドレスを使用してデータベースエンドポイントに接続しているように見えます。この場合、VPC 外のデータベースエンドポイントへの進入は、レプリケーション インスタンスのパブリック IP アドレスではなく NAT アドレスからの進入を許可する必要があります。
- 非 RDBMS エンジンの VPC エンドポイント - AWS DMS は、非 RDBMS エンジンの VPC エンドポイントをに対応しません。

での SSL の使用 AWS Database Migration Service

Secure Sockets Layer (SSL) を使用することで、ソースおよびターゲットエンドポイントへの接続を暗号化できます。そのためには、 AWS DMS マネジメントコンソールまたは AWS DMS API を使用して、エンドポイントに証明書を割り当てることができます。 AWS DMS コンソールを使用して証明書を管理することもできます。

どのデータベースも同じ方法で SSL を使用しているとは限りません。Amazon Aurora MySQL 互換エディションは、SSL のエンドポイントとして、サーバー名、クラスター内のプライマリ インスタンスのエンドポイントを使用します。Amazon Redshift エンドポイントではすでに SSL 接続が使用されているため、 AWS DMSによりセットアップされた SSL 接続は必要ありません。Oracle エンドポイントには追加の手順が必要です。詳細については、「Oracle エンドポイントでの SSL のサポート」をご参照ください。

トピック

- AWS DMSで SSL を使用する場合の制限
- 証明書の管理
- MySQL 互換、PostgreSQL、または SQL Server のエンドポイントでの SSL の有効化

エンドポイントに証明書を割り当てるには、ルート証明書を指定するか、エンドポイントにデプロイされるサーバー SSL 証明書の署名に使用された、ルートに導く (証明書バンドルとして) 中間 CA 証明書チェーンを指定します。証明書は、PEM 形式の X509 ファイルとしてのみ受け入れられます。証明書をインポートすると、エンドポイントにその証明書を指定するために使用できる Amazon リソースネーム (ARN) を受け取ります。Amazon RDS を使用する場合は、Amazon RDS によってホストされている `rds-combined-ca-bundle.pem` ファイルで提供されているルート CA と証明書バンドルをダウンロードできます。このファイルのダウンロード方法については、Amazon RDS ユーザーガイド の「SSL/TLS を使用した DB インスタンス接続の暗号化」をご参照ください。

SSL 証明書認証に使用する SSL モードは、複数の中から選択できます。

- none – 接続は暗号化されていません。このオプションは安全ではありませんが、必要なオーバーヘッドが小さくなります。
- require – 接続は SSL (TLS) を使用して暗号化されますが、CA 検証は行われません。このオプションは安全性が高まりますが、必要なオーバーヘッドが増えます。
- verify-ca – 接続は暗号化されています。このオプションは安全性が高まりますが、必要なオーバーヘッドが増えます。このオプションでは、サーバー証明書が認証されます。

- verify-full – 接続は暗号化されています。このオプションは安全性が高まりますが、必要なオーバーヘッドが増えます。このオプションでは、サーバー証明書が認証され、サーバーのホスト名が証明書のホスト名属性と一致することが確認されます。

すべての SSL モードがすべてのデータベースエンドポイントで機能するわけではありません。次の表は、各データベースエンジンでサポートされている SSL モードを示しています。

DB エンジン	なし	require	verify-ca	verify-full
MySQL/MariaDB/ Amazon Aurora MySQL	デフォルト値	サポート外	サポート対象	サポート
Microsoft SQL Server	デフォルト値	サポート	サポート外	サポート
PostgreSQL	デフォルト値	サポート	サポート対象	サポート
Amazon Redshift	デフォルト値	SSL が有効でない	SSL が有効でない	SSL が有効でない
Oracle	デフォルト値	サポート外	サポート	サポート外
SAP ASE	デフォルト値	SSL が有効でない	SSL が有効でない	サポート
MongoDB	デフォルト値	サポート	サポート外	サポート
Db2 LUW	デフォルト値	サポート外	サポート	サポート外
Db2 for z/OS	デフォルト値	サポート外	サポート	サポート外

Note

DMS コンソールまたは API の SSL モードオプションは、Kinesis や DynamoDB などの一部のデータ ストリーミングおよび NoSQL サービスには適用されません。これらはデフォルトで安全なため、DMS は SSL モードの設定が none と等しくなっていることを示します (SSL モード=なし)。SSL を使用するために、エンドポイントに追加の設定は必要ありません。例えば、Kinesis をターゲット エンドポイントとして使用する場合、デフォルトでセキュリ

ティで保護されます。Kinesis への API コールはすべて SSL を使用するため、DMS エンドポイントに追加の SSL オプションは不要です。DMS が Kinesis Data Stream への接続時にデフォルトで使用する HTTPS プロトコルを使用して、SSL エンドポイントを介して安全にデータを保存し取得できます。

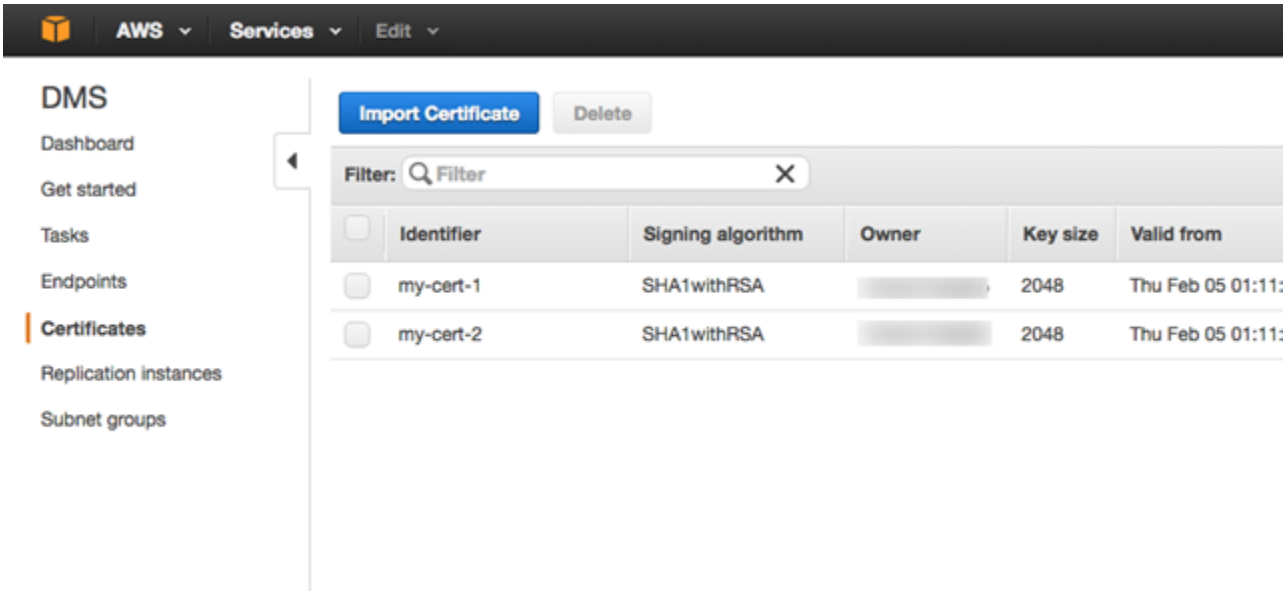
AWS DMSで SSL を使用する場合の制限

での SSL の使用に関する制限は次のとおりです AWS DMS。

- Amazon Redshift ターゲットエンドポイントへの SSL 接続はサポートされていません。 AWS DMS は Amazon S3 バケットを使用して Amazon Redshift データベースにデータを転送します。この転送は、Amazon Redshift によってデフォルトで暗号化されます。
- SSL が有効な Oracle エンドポイントで変更データキャプチャ (CDC) タスクを実行すると、SQL タイムアウトが発生することがあります。CDC カウンターに想定の数値が反映されないという問題がある場合は、タスク設定の `ChangeProcessingTuning` セクションの `MinimumTransactionSize` パラメータに小さい値を設定します。最低値 100 から始めることができます。`MinimumTransactionSize` パラメータの詳細については、「変更処理のチューニング設定」をご参照ください。
- インポートできる証明書の形式は、.pem 形式および .sso (Oracle ウォレット) 形式のみです。
- 場合によっては、サーバーの SSL 証明書が中間認証局 (CA) によって署名されていることがあります。その場合は、中間 CA からルート CA までの証明書チェーン全体が 1 つの.pem ファイルとしてインポートされていることを確認します。
- サーバーで自己署名証明書を使用している場合、SSL モードとして [require] を選択します。require SSL モードでは、サーバーの SSL 証明書が暗黙的に信頼され、証明書が CA により署名されたかどうかの検証は試行されません。

証明書の管理

DMS コンソールを使用すると、SSL 証明書を表示および管理できます。DMS コンソールを使用して証明書をインポートすることもできます。



MySQL 互換、PostgreSQL、または SQL Server のエンドポイントでの SSL の有効化

新しく作成したエンドポイントまたは既存のエンドポイントに SSL 接続を追加できます。

SSL を使用して AWS DMS エンドポイントを作成するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。

   AWS Identity and Access Management (IAM) ユーザーとしてサインインしている場合は、 にアクセスするための適切なアクセス許可があることを確認してください AWS DMS。データベース移行に必要なアクセス許可の詳細については、「AWS DMSの使用に必要な IAM アクセス許可」をご参照ください。
2. ナビゲーションペインで [Certificates] を選択します。
3. [Import Certificate] を選択します。
4. エンドポイントへの接続の暗号化に使用する証明書をアップロードします。

> **Note**
>
> また、データベースエンドポイントの作成ページで新しい CA 証明書を追加を選択して、エンドポイントを作成または変更 AWS DMS するときにコンソールを使用して証明書をアップロードすることもできます。

ターゲットとして Aurora サーバーレスの場合は、Aurora サーバーレスでの TLS/SSL の使用に記載されている証明書を取得します。

5. ステップ 2: ソースエンドポイントとターゲットエンドポイントを指定するでの説明に従って、エンドポイントを作成します。

SSL を使用するように既存の AWS DMS エンドポイントを変更するには

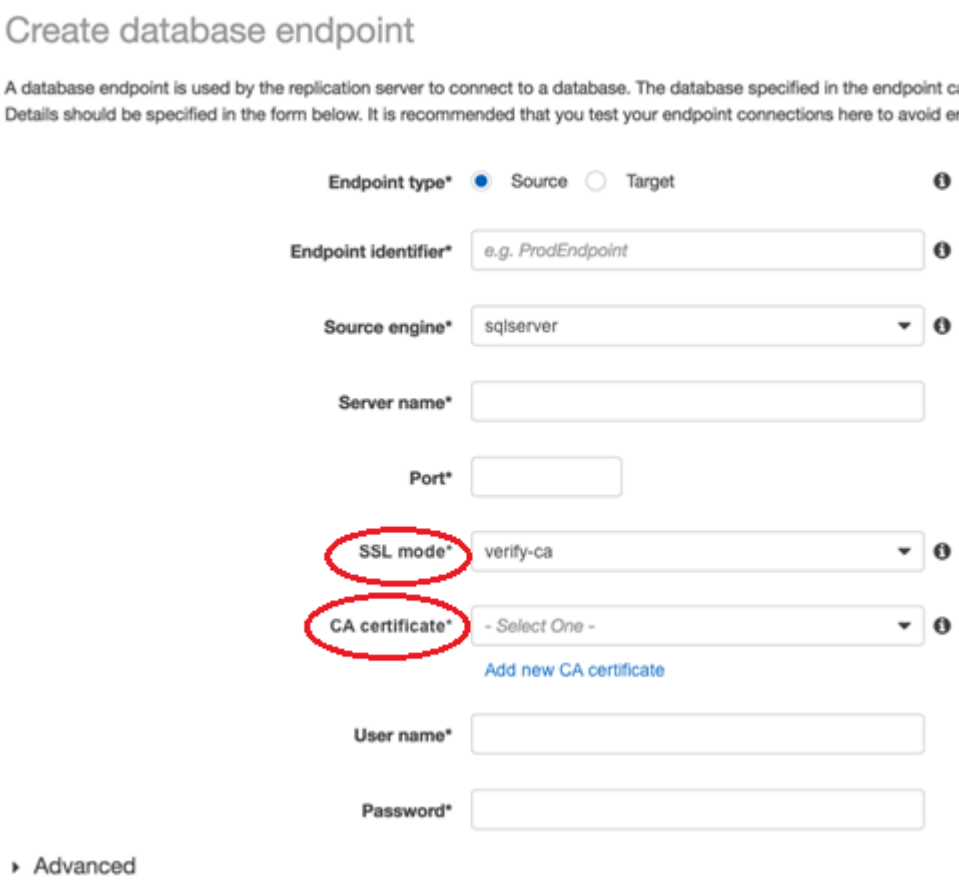
1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。

   IAM ユーザーとしてサインインしている場合は、 AWS DMSにアクセスするための適切なアクセス許可があることを確認します。データベース移行に必要なアクセス許可の詳細については、「AWS DMSの使用に必要な IAM アクセス許可」をご参照ください。
2. ナビゲーションペインで [Certificates] を選択します。
3. [Import Certificate] を選択します。
4. エンドポイントへの接続の暗号化に使用する証明書をアップロードします。

   > **Note**
   >
   > また、データベースエンドポイントの作成ページで新しい CA 証明書を追加を選択して、エンドポイントを作成または変更 AWS DMS するときにコンソールを使用して証明書をアップロードすることもできます。

5. ナビゲーションペインで、[Endpoints] を選択し、変更するエンドポイントを選択して [Modify] を選択します。
6. SSL モードの値を選択します。

   [verify-ca] モードまたは [verify-full] モードを選択した場合は、次に示すように、使用する証明書を [CA 証明書] に指定します。



7. [変更] を選択します。
8. エンドポイントが変更されている場合は、エンドポイントを選択して [接続のテスト] を選択し、SSL 接続が機能しているかどうかを調べます。

ソースおよびターゲットエンドポイントを作成したら、これらのエンドポイントを使用するタスクを作成します。タスクの作成に関する詳細については、「ステップ 3: タスクを作成してデータを移行する」をご参照ください。

データベースのパスワードの変更

ほとんどの状況では、ソースまたはターゲットエンドポイント用のデータベースのパスワードを変更するのは簡単です。移行またはレプリケーションタスクで現在使用しているエンドポイントのデータベースパスワードを変更する必要がある場合、プロセスにはいくつかの追加ステップが必要です。以下の手順は、その方法を示しています。

移行またはレプリケーションタスクでエンドポイント用のデータベースのパスワードを変更するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/dms/v2/ で AWS DMS コンソールを開きます。

   IAM ユーザーとしてサインインしている場合は、 AWS DMSにアクセスするための適切なアクセス許可があることを確認します。必要なアクセス権限の詳細については、「AWS DMSの使用に必要な IAM アクセス許可」をご参照ください。
2. ナビゲーションペインで、[データベース移行タスク] を選択します。
3. データベースのパスワードを変更するエンドポイントを使用するタスクを選択してから、[Stop] を選択します。
4. タスクが停止されている間、データベースの操作に使用するネイティブツールを使用して、エンドポイント用のデータベースのパスワードを変更できます。
5. DMS マネジメントコンソールに戻り、ナビゲーションペインから [Endpoints] を選択します。
6. パスワードを変更したデータベースのエンドポイントを選択してから、[Modify] を選択します。
7. [パスワード] ボックスに新しいパスワードを入力し、[保存] を選択します。
8. ナビゲーションペインで、[データベース移行タスク] を選択します。
9. 先ほど停止したタスクを選択し、[再起動/再開] を選択します。
10. タスクを続行する方法に応じて、[再起動] または [再開] のいずれかを選択し、[タスクの開始] を選択します。

AWS Database Migration Service のクォータ

以下では、AWS Database Migration Service (AWS DMS) でのリソース クォータと命名に関する制約について説明します。

次のデータベースの最大サイズ AWS DMS が移行可能かは、いくつかの要因によって決まります。これらには出典環境、出典データベースのデータ ディストリビューション、出典システムがどの程度ビジーかによって異なります。

特定のシステムが AWS DMS の候補となるかどうかを調べる最善の方法は、テストすることです。ゆっくりとスタートして設定が機能するか確認します。次にいくらか複雑なオブジェクトを追加します。最後に、テストとして全ロードを試みます。

AWS Database Migration Service のリソース クォータ

各 AWS アカウントには AWS リージョン別に、作成可能な AWS DMS リソースの数に適用されるクォータがあります。リソースのクォータに達すると、そのリソースを作成するための追加の呼び出しは、失敗して例外が発生します。

次の表に、AWS リージョンごとの AWS DMS リソースとそのクォータを示します。

[リソース]	デフォルトのクォータ
API リクエストのスロットリング	1 秒あたり最大 100 リクエスト
API リクエストレートのリフレッシュレート	1 秒あたり 8 リクエスト
ユーザーアカウントごとのレプリケーションインスタンス	60
レプリケーション インスタンスの合計ストレージ容量	30 TB
ユーザーアカウントごとのイベントサブスクリプション数	60
ユーザーアカウントごとのレプリケーションサブネットグループ数	60
レプリケーション サブネット グループあたりのサブネット	60

[リソース]	デフォルトのクォータ
ユーザーアカウントあたりのエンドポイント数	1,000
レプリケーション インスタンスあたりのエンドポイント	100
ユーザーアカウントごとのタスク数	600
レプリケーション インスタンスごとのタスク	200
ユーザーアカウントあたりの証明書数	100
ユーザーアカウントあたりのデータプロバイダー数	1,000
ユーザーアカウントごとのインスタンスプロファイル数	60
ユーザーアカウントごとの移行プロジェクト数	10
ユーザーアカウントごとの DMS データコレクター数	10
一度に生成されるターゲットレコメンデーション数	100
DMS データコレクターが 1 時間あたりにアップロードできるファイル数	500
ユーザーアカウントごとの同種データ移行数	600
同時実行される同種データ移行数	100
移行プロジェクトあたりの同種データ移行数	10
サーバーレスレプリケーション	100

API リクエストのスロットリング クォータとリフレッシュレートの詳細については、「API リクエストのスロットリングについて把握する」をご参照ください。

ストレージのクォータ 30 TB は、特定の AWS リージョン内のAWS DMS レプリケーション インスタンスすべてに適用されます。ターゲットとソースの同期が間に合わない場合に変更をキャッシュするためや、ログ情報を保存するために、このストレージは使用されます。

API リクエストのスロットリングについて把握する

AWS DMS は、さまざまな API リクエストクォータをサポートします。ただし、最大 API リクエストは、200 API コールです。つまり、API リクエストがこのレートを超えると、API リクエストがスロットリング調整されます。また、別の API リクエストを行う前にクォータの更新に AWS DMS でかかる時間に応じて、1 秒あたりの API コール数を減らすこともできます。このクォータは、API コールを直接行う場合と、AWS DMS マネジメントコンソールを使用する一環としてAPI コールがユーザーの代わりに行われる場合の両方に適用されます。

API リクエストのスロットリングがどのように機能するかは、AWS DMS が API リクエストを追跡するトークン バケットを保持することを想像すると理解しやすいです。このシナリオでは、バケット内の各トークンで単一の API コールを行うことができます。バケットに一度に含めることができるトークンは 200 までです。API コールを行うと AWS DMS はバケットから 1 つのトークンを削除します。200 の API コールを 1 秒未満で実行すると、バケットは空になり、別の API コールの試行は失敗します。API コールを行わない 1 秒ごとに、AWS DMS はバケットに最大 200 トークンを上限に 4 トークンを追加します。これは、AWS DMS API リクエストのリフレッシュレートです。スロットリング後はいつでも、バケットにトークンを追加すると、呼び出しが再びスロットリングされるまで、トークンと同じ数の追加の API コールを使用できます。

AWS CLI を使用してスロットリングされた API コールを実行いる場合は AWS DMS が次のようなエラーを返します：

```
An error occurred (ThrottlingException) when calling the AwsDmsApiCall operation
 (reached max retries: 2): Rate exceeded
```

ここで *AwsDmsApiCall* はスロットルされた AWS DMS API オペレーションの名前たとえば、DescribeTableStatistics です。その後、スロットリングを回避するために、十分な遅延後に再試行するか、別の呼び出しを行うことができます。

> **Note**
>
> Amazon EC2 などの他のサービスによって管理される API リクエストスロットリングとは異なり、AWS DMS によって管理される API リクエスト　スロットリング クォータの増加を注文することはできません。

AWS Database Migration Service での移行タスクのトラブルシューティング

以下では、AWS Database Migration Service (AWS DMS) での問題のトラブルシューティングに関するトピックを見つけることができます。これらのトピックは、AWS DMS とを選択したエンドポイント データベースの両方を使用して一般的な問題を解決するのに役立ちます。

AWS Support ケースがあれば、サポートエンジニアが、エンドポイント データベース構成の 1 つに関する潜在的な問題を特定できることもあります。エンジニアが、データベースに関する診断情報を返すためのサポート スクリプトを実行するように頼む場合もあります。タスク設定ファイルを使用してタスク設定を設定する方法については、「AWS DMS での診断サポート スクリプトの操作」をご参照ください。

トラブルシューティングの目的で、 はレプリケーションインスタンスでトレースファイルとダンプファイルをAWS DMS収集します。トラブルシューティングが必要な問題が発生した場合は、これらのファイルを AWS サポートに提供できます。デフォルトでは、DMS は 30 日より古いトレースファイルとダンプファイルを消去します。トレースとダンプファイルの収集をオプトアウトするには、AWS サポートでケースを開きます。

トピック

- LOB 列でプライマリ キーが作成されたときにタスクが失敗する
- プライマリ キーのないターゲットテーブル上の重複レコード
- 予約済み IP 範囲の送信元エンドポイント
- Amazon Athena クエリでタイムスタンプが文字化けする
- Oracle に関する問題のトラブルシューティング
- MySQL に関する問題のトラブルシューティング
- PostgreSQL に関する問題のトラブルシューティング
- Microsoft SQL Server 問題のトラブルシューティング
- Amazon Redshift に関する問題のトラブルシューティング
- Amazon Aurora MySQL に関する問題のトラブルシューティング
- SAP ASE に関する問題のトラブルシューティング
- IBM Db2 に関する問題のトラブルシューティング
- AWS Database Migration Service のレイテンシーに関する問題のトラブルシューティング
- AWS DMS での診断サポート スクリプトの操作
- AWS DMS 診断サポート AMI の使用

移行タスクの実行が遅い

移行タスクの実行が遅くなる、または後続のタスクの実行が初期タスクより遅くなる原因として、いくつかの問題が考えられます。

移行タスクの実行が遅くなる最も一般的な原因は、AWS DMS のレプリケーション インスタンスに割り当てられているリソースが不十分であることです。レプリケーション インスタンスの CPU およびメモリ、スワップファイル、 IOPS の使用率をチェックし、インスタンスのリソースが実行中のタスクのために十分であることを確認します。例えば、エンドポイントとしての Amazon Redshift の複数のタスクは I/O 集約型です。レプリケーションのインスタンスに対して IOPS を増やすか、より効率的に移行するためにレプリケーションの複数のインスタンス間で作業を分割できます。

レプリケーションのインスタンスサイズの決定に関する詳細については、「レプリケーションインスタンス向けの最適なサイズの選択」をご参照ください。

以下を実行すると、初期の移行のロードがスピードアップします。

- ターゲットが Amazon RDS DB インスタンスの場合、Multi-AZ がターゲット DB インスタンスに対して有効でないことを確認します。

- ロード中の自動バックアップまたはターゲットデータベースでのログ作成をオフにし、移行が完了したらこれらの機能をオンに戻します。
- プロビジョンド IOPS がターゲットで利用可能な場合は、この機能を使用します。
- 移行データに LOB が含まれる場合は、タスクが LOB 移行のために最適化されていることを確認します。LOB 用の最適化の詳細については、「ターゲットメタデータのタスク設定」をご参照ください。

タスクのステータスバーが動かない

タスクのステータスバーで、タスクの進捗状況を予測できます。この予測の正確さはソースデータベースのテーブル統計の正確さによって異なります。テーブル統計が正確であればあるほど、正確に予測できます。

予測された行の統計がないテーブルが 1 つだけのタスクでは、AWS DMS はどのような種類であっても完了率の予測を提供できません。この場合、タスクのステータスと、ロードされた行の表示を使って、タスクが実際に実行されて進行していることを確認できます。

タスクは完了しましたが、何も移行されませんでした

タスクの完了後に何も移行されなかった場合は、次の操作を実行します。

- エンドポイントを作成したユーザーが、移行するテーブルへの読み取りアクセス権を持っているかどうかを確認します。
- 移行するオブジェクトがテーブルであるかどうかを確認します。ビューの場合は、テーブルマッピングを更新し、オブジェクトロケータを[view](ビュー) または[all](すべて) として指定します。詳細については、「 コンソールからテーブル選択および変換を指定する」を参照してください。

外部キーとセカンダリ インデックスが見つからない

AWS DMS によって、テーブル、プライマリ キー、場合によっては一意のインデックスが作成されますが、ソースからデータを効率的に移行するために必要ではない他のオブジェクトは作成されません。たとえば、セカンダリインデックス、非プライマリキーの制約、データデフォルトは作成されません。

データベースからセカンダリオブジェクトを移行するには、ソースデータベースと同じデータベースエンジンに移行中の場合、データベースのネイティブツールを使用します。ソースデータベースで

使用したものと異なるデータベースエンジンへ移行中の場合、AWS Schema Conversion Tool (AWS SCT) を使用してセカンダリオブジェクトを移行します。

AWS DMS は CloudWatch ログを作成しない

レプリケーションタスクで CloudWatch ログが作成されない場合は、アカウントに `dms-cloudwatch-logs-role`ロールがあることを確認してください。このロールが存在しない場合は、次を実行して作成します。

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. [ロール] タブをクリックします。[ロールの作成] を選択します。
3. [信頼されたエンティティを選択] セクションで、[AWS のサービス] を選択します。
4. [ユースケースの選択] セクションで、[DMS] を選択します。
5. [次へ: アクセス許可] を選択します。
6. **`AmazonDMSCloudWatchLogsRole`** 検索フィールドに「」と入力し、AmazonDMSCloudWatchLogsRole の横にあるチェックボックスをオンにします。これにより、 へのアクセスAWS DMS許可が付与されます CloudWatch。
7. [次へ: タグ] を選択します。
8. [次へ: レビュー] を選択します。
9. [ロール名] に **`dms-cloudwatch-logs-role`** と入力します。この名前では大文字と小文字が区別されます。
10. [ロールの作成] を選択します。

Amazon RDS への接続で問題が発生する

ソースまたはターゲットとして設定した Amazon RDS DB インスタンスに接続できない理由は複数考えられます。チェックすべき項目 :

- ユーザー名とパスワードの組み合わせが正しいことを確認します。
- インスタンス用に Amazon RDS コンソールに表示されるエンドポイント値が AWS DMS エンドポイントを作成するのに使用したエンドポイント識別子と同じであることを確認します。
- インスタンス用に Amazon RDS コンソールで表示されるポート値が AWS DMSエンドポイントに割り当てられたポートと同じであることを確認します。

- Amazon RDS DB インスタンスに割り当てられたセキュリティグループが、AWS DMS のレプリケーションのインスタンスからの接続を許可することを確認します。
- AWS DMS のレプリケーション インスタンスと Amazon RDS DB インスタンスが同じ仮想プライベートクラウド (VPC) にない場合は、DB インスタンスがパブリックにアクセス可能であることを確認します。

エラーメッセージ: スレッドの接続文字列が正しくありません。正しくないスレッド値「0」

エンドポイントへの接続をテストしているとき、このエラーが頻繁に発生します。このエラーは、接続文字列にエラーがあることを示します。例えば、ホスト IP アドレスの後ろにスペースがあります。もう一つは、接続文字列にコピーされた不正な文字です。

ネットワーキングに関する問題の発生

最も一般的なネットワーキング問題には、AWS DMS のレプリケーションのインスタンスによって使用される VPC のセキュリティグループが関係しています。デフォルトでは、このセキュリティグループではすべてのポートの 0.0.0.0/0 からの送信を許可するルールがあります。多くの場合、このセキュリティグループを変更するか、独自のセキュリティグループを使用します。その場合は、少なくとも、それぞれのデータベースポート上のソース エンドポイントとターゲット エンドポイントに出力を与えるようにしてください。

その他の設定関連の問題には、次のようなものがあります：

- [Replication instance and both source and target endpoints in the same VPC] (レプリケーション インスタンスと、ソースとターゲット両方のエンドポイントが同じ VPC 内にある) - エンドポイントが使用するセキュリティグループはレプリケーション インスタンスからのデータベースポートへの受信を許可する必要があります。レプリケーション インスタンスによって使用されるセキュリティグループにエンドポイントへの入力があることを確認します。または、レプリケーション インスタンスのプライベート IP アドレスへのアクセスを許可するエンドポイントによって使用されるセキュリティグループにルールを作成できます。
- [Source endpoint is outside the VPC used by the replication instance (using an internet gateway)] (ソースエンドポイントがレプリケーション インスタンス (インターネットゲートウェイを使用) が使用する VPC の外部にある) - VPC セキュリティグループは VPC 宛てでないトラフィックをインターネットゲートウェイに送信するルーティングルールを含む必要があります。この設定では、エンドポイントへの接続がレプリケーション インスタンス上のパブリック IP アドレスから行われているように見えます。

- [Source endpoint is outside the VPC used by the replication instance (using a NAT gateway)] (ソースエンドポイントがレプリケーションインスタンスで使用されている VPC の外にあり、NAT のゲートウェイを使用している) - 単一の Elastic network interface にバインドされた単一の Elastic IP アドレスを使用してネットワークアドレス変換 (NAT) ゲートウェイを設定します。この NAT ゲートウェイは NAT 識別子 (nat-#####) を受け取ります。

  場合によっては、インターネットゲートウェイではなく、その NAT ゲートウェイへのデフォルトルートが VPC に含まれていることがあります。このような場合、レプリケーション インスタンスは NAT ゲートウェイのパブリック IP アドレスを使用してデータベース エンドポイントに接続しているように見えます。この場合、VPC 外のデータベース エンドポイントへの進入は、レプリケーション インスタンスのパブリック IP アドレスではなく NAT アドレスからの進入を許可する必要があります。

独自のオンプレミス ネームサーバーの使用については、「 独自のオンプレミスネームサーバーの使用」をご参照ください。

全ロード後、CDC が停止する

複数の AWS DMS 設定が互いに競合すると、全ロード移行後に、レプリケーション変更が遅くなるか停止することがあります。

例えば、[Target table preparation mode](ターゲットテーブル作成モード) パラメータが [Do nothing](何もしない)または[Truncate](切り詰め) に設定されているとします。この場合、AWS DMS が、プライマリ キーと一意なインデックスの作成を含め、ターゲットテーブルでのセットアップを行わないようにしました。ターゲットテーブルでプライマリ キーまたは固有キーを作成していない場合、AWS DMS は更新プログラムごとに、テーブル全体のスキャンを行います。このアプローチは、パフォーマンスに大きな影響を与える可能性があります。

タスク再開時のプライマリ キー制約違反エラー

このエラーは、以前の移行タスクからのターゲットデータベースにデータが残っている場合に発生することがあります。[Target table preparation mode] (ターゲットテーブル作成モード) パラメータが [Do nothing](何もしない) に設定されている場合、AWS DMS は以前のタスクから挿入されたデータの削除を含め、ターゲットテーブルの準備を行いません。

タスクを再開し、これらのエラーを回避するために、タスクの以前の実行からターゲットテーブルに挿入される行を削除する必要があります。

スキーマの初回ロードが失敗する

場合によっては、スキーマの初期ロードがエラー `Operation:getSchemaListDetails:errType=, status=0, errMessage=, errDetails=` で失敗することがあります。

このような場合、AWS DMS によって使用されるユーザー アカウントがソース エンドポイントに接続するには、必要なアクセス許可がありません。

不明なエラーが発生してタスクが失敗する

不明なタイプのエラーの原因はさまざまです。ただし、多くの場合、問題には AWS DMS に割り当てられたレプリケーション インスタンスリソースが不十分です。

レプリケーション インスタンスの CPU およびメモリ、スワップファイル、 IOPS の使用率をチェックし、インスタンスのリソースが移行を実行するために十分であることを確認します。モニタリングの詳細については、「AWS Database Migration Service のメトリクス」をご参照ください。

タスクを再開するとテーブルが最初からロードされる

AWS DMS は、テーブルの初回ロードを完了しなかった場合、テーブルのロードを最初から再開します。タスクが再起動されると、AWS DMSは、最初のロードが完了しなかったときに、テーブルを最初からリロードします。

タスクあたりのテーブル数が問題の原因の問題

レプリケーション タスクあたりのテーブル数に制限はありません。ただし、経験則として、タスク内のテーブルの数を 60,000 個未満に制限することをお勧めします。1 つのタスクが 60,000 個超のテーブルを使用すると、多くの場合リソース使用がボトルネックになることがあります。

LOB 列でプライマリ キーが作成されたときにタスクが失敗する

FULL LOB モードまたは LIMITED LOB モードでは、AWS DMS は LOB データ型のプライマリ キーのレプリケーションをサポートしません。

DMS は、最初に LOB 列が NULL である行を移行し、後で LOB 列を更新します。したがって、LOB 列にプライマリキーが作成されると、プライマリキーを NULL にできないため、最初の挿入は失敗

します。回避方法として、別の列をプライマリ キーとして追加し、LOB 列からプライマリ キーを削除します。

プライマリ キーのないターゲットテーブル上の重複レコード

全ロードと CDC タスクを実行すると、プライマリ キーまたは一意のインデックスがないターゲットテーブルに重複レコードを作成できます。全ロードと CDC タスク中にターゲットテーブルでレコードが重複しないようにするには、ターゲットテーブルにプライマリ キーまたは一意のインデックスがあることを確認してください。

予約済み IP 範囲の送信元エンドポイント

AWS DMS ソースデータベースが 192.168.0.0/24 の予約済み IP 範囲内の IP アドレスを使用している場合、ソースエ ンドポイント接続テストは失敗します。次のステップは、考えられる回避方法を提供します：

1. 192.168.0.0/24 のソースデータベースと通信できる予約済み範囲外の Amazon EC2 インスタンスを見つけます。
2. socat プロキシをインストールして実行します。例を以下に示します。

```
yum install socat

socat -d -d -lmlocal2 tcp4-listen:database port,bind=0.0.0.0,reuseaddr,fork
 tcp4:source_database_ip_address:database_port
&
```

AWS DMS エンドポイントには、Amazon EC2インスタンスの IP アドレスと上記のデータベースポートを使用します。AWS DMS のデータベースポートへのアクセスを許可するセキュリティグループがエンドポイントにあることを確認します。DMSタスクの実行中はプロキシが実行されている必要があることに注意します。ユースケースによっては、プロキシのセットアップを自動化する必要がある場合があります。

Amazon Athena クエリでタイムスタンプが文字化けする

Athena クエリでタイムスタンプがガブルされている場合は、 AWS Management Consoleまたは ModifyEndpointアクションを使用して Amazon S3 エンドポイント

のparquetTimestampInMillisecond値を に設定しますtrue。詳細については、[S3Settings] (S3設定)をご参照ください。

Oracle に関する問題のトラブルシューティング

以下では、AWS DMS を Oracle データベースと使用する際に固有の問題のトラブルシューティングについて学習できます。

トピック

- ビューからデータを取得する
- Oracle 12c から LOB を移行する
- Oracle LogMiner と Binary Reader の切り替え
- エラー: Oracle CDC は停止しました。122301 Oracle CDC の最大再試行カウンターを超えました。
- Oracle ソース エンドポイントにサプリメンタル ログを自動的に追加する
- LOB の変更がキャプチャされない
- エラー: ORA-12899: 列 column-name の値が大きすぎる
- NUMBER のデータ型が誤って解釈される
- 全ロード中にレコードが欠落している
- テーブルエラー
- エラー:Oracle アーカイブされた REDO ログの宛先 ID を取得できません
- Oracle REDO ログまたはアーカイブログの読み取りパフォーマンスの評価

ビューからデータを取得する

ビューからデータを 1 回プルできます。これを継続的レプリケーションに使用することはできません。ビューからデータを抽出できるようにするには、Oracle ソースエンドポイントページの [エンドポイント設定] セクションに次のコードを追加する必要があります。ビューからデータを抽出すると、そのビューはターゲットスキーマのテーブルとして表示されます。

```
"ExposeViews": true
```

Oracle 12c から LOB を移行する

AWS DMS では、Oracle データベースの変更をキャプチャするために、Binary Reader と Oracle の 2 つの方法を使用できます LogMiner。デフォルトでは、 AWS DMSは Oracle を使用して変更 LogMiner をキャプチャします。ただし、Oracle 12c では、Oracle LogMiner は LOB 列をサポートしていません。Oracle 12c での LOB 列の変更をキャプチャするには、Binary Reader を使用します。

Oracle LogMiner と Binary Reader の切り替え

AWS DMS では、ソース Oracle データベースの変更をキャプチャするために、Binary Reader と Oracle の 2 つの方法を使用できます LogMiner。Oracle がデフォルト LogMiner です。Binary Reader を使用して変更をキャプチャするように切り替えるには、次を実行します。

Binary Reader を使用して変更をキャプチャするには

1. にサインインAWS Management Consoleし、https://console.aws.amazon.com/dms/v2/ で AWS DMSコンソールを開きます。
2. [Endpoints] (エンドポイント) を選択します。
3. Binary Reader を使用したい Oracle ソース エンドポイントを選択します。
4. [変更] を選択します。
5. [Advanced] (詳細) を選択し、[Extra connection attributes](追加の接続属性) テキストボックスに以下のコードを追加します。

```
useLogminerReader=N
```

6. SQL-Plus のような Oracle デベロッパーツールを使って、Oracle エンドポイントに接続するのに使用される AWS DMS ユーザーアカウントに追加の権限を与えます。

```
SELECT ON V_$TRANSPORTABLE_PLATFORM
```

エラー: Oracle CDC は停止しました。122301 Oracle CDC の最大再試行カウンターを超えました。

このエラーは、AWS DMS が変更をキャプチャするために使用できるようになる前に、必要な Oracle アーカイブログがサーバーから削除されると発生します。データベースサーバーのログリテンションポリシーを増やします。Amazon RDS データベースでは、ログの保持を延長するために、以下の手順を実行します。たとえば、以下のコードを使うと Amazon RDS DB インスタンスでのログの保持が 24 時間に延長されます。

```
exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24);
```

Oracle ソース エンドポイントにサプリメンタル ログを自動的に追加する

デフォルトでは、AWS DMS のサプリメンタルロギングはオフになっています。Oracle ソースエンドポイントのサプリメンタルロギングを自動的にオンにするには、以下の手順を実行します。

Oracle ソースエンドポイントにサプリメンタルロギングを追加するには

1. にサインインAWS Management Consoleし、https://console.aws.amazon.com/dms/v2/ で AWS DMSコンソールを開きます。
2. [Endpoints] (エンドポイント) を選択します。
3. サプリメンタル ログを追加する Oracle ソース エンドポイントを選択します。
4. [変更] を選択します。
5. [Advanced] (詳細) を選択し、[Extra connection attributes] (追加の接続属性) テキストボックスに以下のコードを追加します。

   ```
   addSupplementalLogging=Y
   ```

6. [変更] を選択します。

LOB の変更がキャプチャされない

現在、LOB の変更をキャプチャするには、テーブルに AWS DMS のプライマリキーがあることが必要です。LOB を含むテーブルにプライマリキーがない場合、LOB の変更をキャプチャするにはいくつかのアクションを行うことができます。

- テーブルにプライマリキーを追加します。この手順は、ID 列を追加し、トリガーを使用してシーケンスを入力するだけです。
- プライマリ キーとしてシステム生成 ID を含むテーブルのマテリアライズド ビューを作成し、テーブルではなくマテリアライズド ビューを移行します。
- ロジカルスタンバイを作成し、テーブルにプライマリキーを追加して、ロジカルスタンバイから移行します。

エラー: ORA-12899: 列 *column-name* の値が大きすぎる

エラー[ORA-12899: value too large for column *column-name*] (ORA-12899: column-name 列の値が大きすぎる) は、多くの場合、いくつかの問題によって引き起こされます。

これらの問題の 1 つで、ソースデータベースとターゲットデータベースで使用される文字セットに不一致があります。

別の問題では、2 つのデータベース間で各国語サポート (NLS) の設定が異なります。このエラーの一般的な原因は、ソースデータベースの NLS_LENGTH_SEMANTICS パラメータが CHAR に設定されており、ターゲットデータベースの NLS_LENGTH_SEMANTICS パラメータが BYTE に設定されていることです。

NUMBER のデータ型が誤って解釈される

Oracle NUMBER データ型は、NUMBER の精度とスケールに応じてさまざまな AWS DMS データ型に変換されます。このような変換はOracle のソースデータ型に記載されています。ソースの Oracle エンドポイントのエンドポイント設定の使用も、NUMBER 型の変換方法に影響します。エンドポイントの設定は「のソースとして Oracle を使用する場合のエンドポイント設定 AWS DMS」に記載されています。

全ロード中にレコードが欠落している

全ロードを実行する場合、AWS DMS は、データベースレベルで開いているトランザクションを検索し、トランザクションがコミットされるのを待ちます。例えば、タスクの設定

TransactionConsistencyTimeout=600 に基づいて、AWS DMS は、オープントランザクションがテーブルマッピングに含まれていないテーブル上にある場合でも、10 分間待機します。ただし、オープントランザクションがテーブルマッピングに含まれるテーブルにあり、トランザクションが時間内にコミットされていない場合、ターゲットテーブルの結果からレコードが欠落します。

TransactionConsistencyTimeout タスクの設定変更し、オープントランザクションがコミットに時間がかかることがわかっている場合は、待機時間を増やします。

また、FailOnTransactionConsistencyBreached タスク設定が false のデフォルト値であることに注意してください。これは AWS DMS が他のトランザクションを引き続き適用しつつも、オープントランザクションは失われることを意味します。オープントランザクションが時間内に終了しないときにタスクが失敗するようにするには、FailOnTransactionConsistencyBreached を true に設定できます。

テーブルエラー

WHERE 句がプライマリ キー列を参照するわけではなく、サプリメンタル ログがすべての列に使用されるわけではない場合、レプリケーション中に Table Error がテーブル統計情報に表示されます。

この問題を解決するには、参照テーブルのすべての列のサプリメンタル ログを有効にします。詳細については、「サプリメンタル ロギングの設定」をご参照ください。

エラー:Oracle アーカイブされた REDO ログの宛先 ID を取得できません

このエラーは、Oracle ソースにアーカイブログが生成されていないか、V$ARCHIVED_LOG が空である場合に発生します。ログを手動で切り替えることで、エラーを解決できます。

Amazon RDS データベースでは、ログ ファイル切り替えのために以下の手順を実行します。switch_logfile プロシージャにはパラメータがありません。

```
exec rdsadmin.rdsadmin_util.switch_logfile;
```

自己管理 Oracle ソース データベースの場合、次のコマンドを使用してログ スイッチを強制します。

```
ALTER SYSTEM SWITCH LOGFILE ;
```

Oracle REDO ログまたはアーカイブログの読み取りパフォーマンスの評価

Oracle ソースでパフォーマンスの問題が発生した場合は、Oracle REDO ログまたはアーカイブログの読み取りパフォーマンスを評価して、パフォーマンスを改善する方法を探ることができます。REDO ログまたはアーカイブログの読み取りパフォーマンスをテストするには、AWS DMS diagnostic Amazon マシンイメージ (AMI) を使用します。

AWS DMS diagnostic AMI を使用して、次のことを行うことができます。

- bFile メソッドを使用して REDO ログファイルのパフォーマンスを評価します。
- LogMiner メソッドを使用して、REDO ログファイルのパフォーマンスを評価します。
- PL/SQL (dbms_lob.read) メソッドを使用して REDO ログファイルのパフォーマンスを評価します。
- シングルスレッドを使用して ASMFile の読み取りパフォーマンスを評価します。
- マルチスレッドを使用して、ASMFile の読み取りパフォーマンスを評価します。
- Direct OS Readfile() Windows または Pread64 Linux 関数を使用して、REDO ログファイルを評価します。

その後、結果に基づいて是正措置を講じることができます。

Oracle REDO ログファイルまたはアーカイブログファイルの読み取りパフォーマンスをテストするには

1. AWS DMS diagnostic AMI Amazon EC2 インスタンスを作成して、このインスタンスに接続します。

   詳細については、「Working with the AWS DMS diagnostic AMI」を参照してください。
2. awsreplperf コマンドを実行します。

   ```
   $ awsreplperf
   ```

   このコマンドは AWS DMS Oracle 読み取りパフォーマンスユーティリティオプションを表示します。

   ```
   0. Quit
   1. Read using Bfile
   2. Read using LogMiner
   3. Read file PL/SQL (dms_lob.read)
   ```

```
4. Read ASMFile Single Thread
5. Read ASMFile Multi Thread
6. Readfile() function
```

3. リストからオプションを選択します。
4. 次のデータベース接続とアーカイブログ情報を入力します。

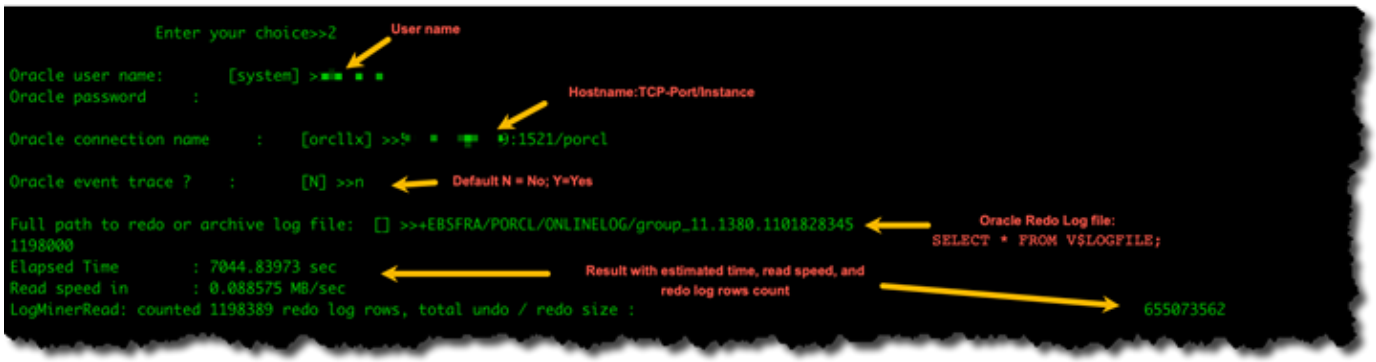
```
Oracle user name [system]:
Oracle password:

Oracle connection name [orcllx]:
Connection format hostname:port/instance

Oracle event trace? [N]:
Default N = No or Y = Yes

Path to redo or archive log file []:
```

5. 表示される出力を調べて、関連する読み取りパフォーマンス情報を探します。例えば、オプション番号 2、 を使用した読み取り LogMinerを選択した結果出力を次に示します。



6. ユーティリティを終了するには、[0] (ゼロ) を入力します。

次のステップ

- 読み取り速度が許容可能なしきい値を下回っていることが結果で示されたら、エンドポイントで「Oracle diagnostic support script」を実行し、「待機時間」、「ロードプロファイル」、「IO プロファイル」セクションを確認します。次に、読み取りパフォーマンスの改善に向けて、正常でない設定を調整します。例えば、REDO ログファイルが最大 2 GB の場合は、パフォーマンスを向上させるために LOG_BUFFER を 200 MB に増やしてみます。
- 「AWS DMS のベストプラクティス」を確認して、DMS レプリケーションインスタンス、タスク、エンドポイントが最適に設定されていることを確認します。

MySQL に関する問題のトラブルシューティング

以下では、AWS DMS を MySQL データベースと使用する際に固有の問題のトラブルシューティングについて学習できます。

トピック

バイナリログ作成が無効化されるため、Amazon RDS DB インスタンスのエンドポイントの CDC タスクが失敗する

自動バックアップが無効化されているためにこの問題が Amazon RDS DB インスタンスに発生します。バックアップ保持期間を 0 以外の値に設定することで自動バックアップを有効にすることができます。

ターゲット MySQL のインスタンスへの接続は、タスクの実行中に接続が切断されます

MySQL ターゲットからの接続が切断されている LOB のタスクがあるとき、タスクログに以下のようなエラーが表示されます。

```
[TARGET_LOAD ]E: RetCode: SQL_ERROR SqlState: 08S01 NativeError:
2013 Message: [MySQL][ODBC 5.3(w) Driver][mysqld-5.7.16-log]Lost connection
to MySQL server during query [122502] ODBC general error.
```

```
 [TARGET_LOAD ]E: RetCode: SQL_ERROR SqlState: HY000 NativeError:
2006 Message: [MySQL][ODBC 5.3(w) Driver]MySQL server has gone away
[122502] ODBC general error.
```

この場合、一部のタスク設定の調整が必要になる場合があります。

タスクが MySQL ターゲットから切断される問題を解決するには、以下を実行します。

- 最大の LOB を保持するために、十分に大きい値に設定された データベース変数 `max_allowed_packet`があることを確認します。
- 以下の変数が大きいタイムアウト値に設定されていることを確認します。これらの変数ごとに、少なくとも 5 分の値を使用することをお勧めします。
  - `net_read_timeout`
  - `net_write_timeout`
  - `wait_timeout`

MySQL システム変数の設定については、MySQL ドキュメントの「サーバーのシステム変数」をご参照ください。

MySQL 互換エンドポイントへの自動コミットを追加する

MySQL 互換ターゲットエンドポイントに自動コミットを追加するには

1. にサインインAWS Management Consoleし、https://console.aws.amazon.com/dms/v2/ で AWS DMSコンソールを開きます。
2. [Endpoints] (エンドポイント) を選択します。
3. 自動コミットを追加したい MySQL 互換ターゲット エンドポイントを選択します。
4. [変更] を選択します。
5. [Advanced] (詳細) を選択し、[Extra connection attributes] (追加の接続属性) テキストボックスに以下のコードを追加します。

```
Initstmt= SET AUTOCOMMIT=1
```

6. [変更] を選択します。

MySQL 互換ターゲットエンドポイントで外部キーを無効化する

MySQL または Amazon Aurora MySQL 互換エディション、 MariaDB エンドポイントの [Advanced] (詳細) セクションの [Extra Connection Attributes] (追加接続属性) に以下を追加して、MySQL の外部キー チェックを無効化できます。

MySQL 互換ターゲットエンドポイントで外部キーを無効化するには

1. にサインインAWS Management Consoleし、https://console.aws.amazon.com/dms/v2/ で AWS DMSコンソールを開きます。
2. [Endpoints] (エンドポイント) を選択します。
3. 外部キーを無効にしたい MySQL または Aurora MySQL、 MariaDB のターゲット エンドポイントを選択します。
4. [変更] を選択します。
5. [Advanced] (詳細) を選択し、[Extra connection attributes] (追加の接続属性) テキストボックスに以下のコードを追加します。

```
Initstmt=SET FOREIGN_KEY_CHECKS=0
```

6. [変更] を選択します。

文字が疑問符に置き換えられる

この問題を引き起こす可能性がある最も一般的な状況は、AWS DMS がサポートしない文字セットでソースエンドポイントの文字がエンコードされている場合です。

[Bad event](不良イベント) ログのエントリ

移行ログの [Bad event](不良イベント) のエントリは通常、ソース データベース エンドポイントで、サポートされていないデータ定義言語(DDL) オペレーションが試行されたことを指します。サポート

されていない DDL オペレーションは、レプリケーション インスタンスが省略できないイベントを発生させ、不良イベントが記録されます。

この問題を解決するには、タスクを最初から再起動します。これにより、テーブルがリロードされ、サポートされていない DDL オペレーションが発行された時点で変更のキャプチャがスタートされます。

MySQL 5.5 の変更データキャプチャ

Amazon RDS MySQL 互換データベースの AWS DMS 変更データ　キャプチャ (CDC) では、MySQL バージョン 5.5 以前でサポートされていない、全イメージ行ベースのバイナリログ作成が必要です。AWS DMS CDC を使用するには、Amazon RDS DB インスタンスを MySQL バージョン 5.6 にアップグレードする必要があります。

Amazon RDS DB インスタンスのバイナリログ保持を延長する

AWS DMS では、変更データキャプチャのバイナリログファイルを保持する必要があります。Amazon RDS DB インスタンスでログの保持を延長するには、以下の手順を使用します。以下の例では、バイナリログの保持を 24 時間に延長します。

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

ログメッセージ: ソースデータベースからの一部の変更は、ターゲットデータベースに適用されても効果がありません。

AWS DMS が MySQL データベースの列の値を既存の値に更新すると、MySQL は `zero rows affected` のメッセージを返します。この動作は、Oracle や SQL Server などの他のデータベースエンジンとは異なります。これらのエンジンは、置換値が現在の値と同じであっても、1 つの行を更新します。

エラー: 識別子が長すぎます

以下のエラーは識別子が長すぎるときに発生します。

```
TARGET_LOAD E: RetCode: SQL_ERROR SqlState: HY000 NativeError:
1059 Message: MySQLhttp://ODBC 5.3(w) Driverhttp://mysqld-5.6.10Identifier
```

```
name 'name' is too long 122502 ODBC general error. (ar_odbc_stmt.c:4054)
```

場合によっては、AWS DMS を、ターゲットデータベースにテーブルとプライマリ キーを作成するように設定します。この場合、現在、DMS はソースデータベースで使用されたプライマリ キーと同じ名前を使用していません。代わりに、DMS はテーブル名に基づいてプライマリ キー名を作成します。テーブル名が長いと、自動作成された識別子が、MySQL に許可されている制限よりも長くなる場合があります。

この問題の当面解決策として、まずターゲットデータベースでテーブルとプライマリキーを事前に作成します。次に、タスク設定[Target table preparation mode] (ターゲットテーブル作成モード)を [Do nothing] (何もしない)または[Truncate] (切り詰め) に設定するタスクを使用し、ターゲットテーブルに代入します。

エラー: サポートされていない文字セットによりフィールドデータ変換が失敗しました

以下のエラーは、サポートされていない文字セットによりフィールドデータ変換が失敗した場合に発生します。

```
"[SOURCE_CAPTURE ]E: Column 'column-name' uses an unsupported character set [120112]
A field data conversion failed. (mysql_endpoint_capture.c:2154)
```

接続に関係したデータベースのパラメータを確認します。以下のコマンドを使用してこれらのパラメータを設定できます。

```
SHOW VARIABLES LIKE '%char%';
```

Error: Codepage 1252 to UTF8 [120112] A field data conversion failed

以下のエラーは、ソース MySQL データベースにコードページ 1252 以外の文字コードが含まれている場合に発生することがあります。

```
[SOURCE_CAPTURE ]E: Error converting column 'column_xyz' in table
'table_xyz with codepage 1252 to UTF8 [120112] A field data conversion failed.
(mysql_endpoint_capture.c:2248)
```

回避策として、ソースの MySQL エンドポイントで追加の接続属性 (CharsetMapping) を使用して、文字セットのマッピングを指定します。このエンドポイント設定を追加する場合は、AWS DMS 移行タスクを最初からやり直す必要がある場合があります。

例えば、次のエンドポイント設定は、ソース文字セットが Utf8 または latin1 である MySQL ソースエンドポイントに使用できます。ここで 65001 は UTF8 コードページ識別子です。

```
CharsetMapping=utf8,65001
CharsetMapping=latin1,65001
```

インデックス、外部キー、カスケード更新、または削除が移行されない

AWS DMS は、インデックスや外部キーなどのセカンダリオブジェクトの移行をサポートしていません。カスケード更新または削除オペレーションから子テーブルに加えられた変更をレプリケートするには、ターゲットテーブルでトリガーを実行する外部キー制約をアクティブにする必要があります。この制限を回避するには、ターゲットテーブルに外部キーを手動で作成します。次に、以下の説明のとおり、フルロードと CDC 向けに単一のタスクを作成するか、フルロードと CDC 向けに 2 つの別々のタスクを作成します。

フルロードと CDC をサポートする単一タスクを作成する

この手順では、フルロードと CDC 向けの単一タスクを使用して外部キーとインデックスを移行する方法について説明します。

フルロードと CDC タスクを作成します。

1. ソーステーブルと一致するように、ターゲットで外部キーとインデックスを含むテーブルを手動で作成します。
2. 次の ECA を AWS DMS ターゲットエンドポイントに追加します。

   ```
   Initstmt=SET FOREIGN_KEY_CHECKS=0;
   ```

3. TargetTablePrepMode を DO_NOTHING に設定して AWS DMS タスクを作成します。

4. Stop task after full load completes 設定を「StopTaskCachedChangesApplied」に設定します。
5. タスクを開始します。AWS DMS はフルロードが完了したらタスクを自動的に停止し、キャッシュされた変更を適用します。
6. 以前追加した SET FOREIGN_KEY_CHECKS ECA を削除します。
7. タスクを再開します。タスクは CDC フェーズに入り、ソースデータベースからの継続的な変更がターゲットに適用されます。

フルロードタスクと CDC タスクを個別に作成する

次の手順では、フルロードと CDC に別々のタスクを使用して外部キーとインデックスを移行する方法について説明します。

フルロードタスクを作成します。

1. ソーステーブルと一致するように、ターゲットで外部キーとインデックスを含むテーブルを手動で作成します。
2. 次の ECA を AWS DMS ターゲットエンドポイントに追加します。

```
Initstmt=SET FOREIGN_KEY_CHECKS=0;
```

3. TargetTablePrepMode パラメータを DO_NOTHING、EnableValidation を FALSE に設定して、AWS DMS タスクを作成します。
4. タスクを開始します。AWS DMS はフルロードが完了したらタスクを自動的に停止し、キャッシュされた変更を適用します。
5. タスクが完了したら、CDC のみのタスクを開始するために、フルロードタスクの UTC での開始時刻、またはバイナリログファイルの名前と位置をメモしておきます。ログを参照して、最初のフルロード開始時刻からの UTC 単位のタイムスタンプを取得します。

CDC のみのタスクを作成する

1. 以前に設定した SET FOREIGN_KEY_CHECKS ECA を削除します。
2. 開始位置を前の手順でメモしておいたフルロード開始時刻に設定して、CDC のみのタスクを作成します。前のステップで記録しておいたバイナリログ位置を使用することもできます。TargetTablePrepMode 設定を「DO_NOTHING」に設定します。必要に応じて、EnableValidation 設定を TRUE に設定して、データ検証を有効にします。

3. CDC のみのタスクを開始し、ログでエラーをモニタリングします。

> **Note**
>
> この回避策は MySQL 間移行にのみ適用されます。バッチ適用の場合は、ターゲットテーブルにアクティブな外部キーがない必要があるため、この方法はバッチ適用機能では使用できません。

PostgreSQL に関する問題のトラブルシューティング

以下では、AWS DMS を PostgreSQL データベースと使用する際に固有の問題のトラブルシューティングについて学習できます。

トピック

切り捨てられる JSON データ型

AWS DMS では、PostgreSQL の JSON データ型は LOB データ型の列として処理されます。つまり、制限付き LOB モードを使用するときの LOB のサイズ制限が、JSON データに適用されます。

例えば、制限付き LOB モードが 4,096 KB に設定されているとします。この場合、4,096 KB よりも大きい JSON データは 4,096 KB の制限で切り捨てられ、PostgreSQL での検証テストに失格します。

以下の口グ情報は、制限付き LOB モードにより切り捨てられた JSON と、失敗した検証を示しています。

```
03:00:49
2017-09-19T03:00:49 [TARGET_APPLY ]E: Failed to execute statement:
  'UPDATE "public"."delivery_options_quotes" SET "id"=? , "enabled"=? ,
  "new_cart_id"=? , "order_id"=? , "user_id"=? , "zone_id"=? , "quotes"=? ,
  "start_at"=? , "end_at"=? , "last_quoted_at"=? , "created_at"=? ,
  "updated_at"=? WHERE "id"=? ' [1022502] (ar_odbc_stmt
2017-09-19T03:00:49 [TARGET_APPLY ]E: Failed to execute statement:
  'UPDATE "public"."delivery_options_quotes" SET "id"=? , "enabled"=? ,
  "new_cart_id"=? , "order_id"=? , "user_id"=? , "zone_id"=? , "quotes"=? ,
  "start_at"=? , "end_at"=? , "last_quoted_at"=? , "created_at"=? ,
  "updated_at"=? WHERE "id"=? ' [1022502] (ar_odbc_stmt.c:2415)
#
03:00:49
2017-09-19T03:00:49 [TARGET_APPLY ]E: RetCode: SQL_ERROR SqlState:
  22P02 NativeError: 1 Message: ERROR: invalid input syntax for type json;,
  Error while executing the query [1022502] (ar_odbc_stmt.c:2421)
2017-09-19T03:00:49 [TARGET_APPLY ]E: RetCode: SQL_ERROR SqlState:
  22P02 NativeError: 1 Message: ERROR: invalid input syntax for type json;,
  Error while executing the query [1022502] (ar_odbc_stmt.c:2421)
```

ユーザー定義のデータ型の列が正しく移行されない

PostgreSQL ソースからレプリケートする場合、AWS DMS はユーザー定義のデータ型を含む列とは別に、すべての列に対して同じデータ型のターゲットテーブルを作成します。このような場合、データ型はターゲットで「character varying」として作成されます。

エラー: 作成用のスキーマが選択されていません

場合によっては、「SQL_ERROR SqlState: 3F000 NativeError: 7 Message: ERROR: no schema has been selected to create in」というエラーが表示されることがあります。

このエラーは、JSON テーブルマッピングがスキーマのワイルドカード値を含み、ソースデータベースがその値をサポートしていない時に発生することがあります。

CDC を使用してテーブルへの削除や更新がレプリケートされない

変更データ キャプチャ (CDC) 中の削除オペレーションと更新オペレーションは、ソーステーブルにプライマリ キーがない場合は無視されます。AWS DMS は、プライマリ キーを持つ PostgreSQL テーブルの変更データ キャプチャ (CDC) をサポートしています。

テーブルにプライマリ キーがない場合、先書きログ (WAL) にはデータベース行の前イメージが含まれていません。この場合は、AWS DMS はテーブルを更新できません。削除オペレーションをレプリケートする場合は、ソーステーブルにプライマリ キーを作成します。

TRUNCATE ステートメントが反映されない

変更データ キャプチャ (CDC) を使用する場合、AWS DMS は TRUNCATE オペレーションをサポートしません。

PostgreSQL の DDL キャプチャを防止する

PostgreSQL ターゲットエンドポイントが DDL ステートメントをキャプチャしないようにするには、次の [エンドポイント設定] ステートメントを追加します。

```
"CaptureDDLs": "N"
```

DDL キャプチャ用のデータベースオブジェクトを作成するスキーマの選択

DDL キャプチャに関連したデータベースオブジェクトをどのスキーマで作成するかを制御できます。次の [エンドポイント設定] ステートメントを追加します。この [エンドポイント設定] パラメータは、ソースエンドポイントのタブで提供されています。

```
"DdlArtifactsSchema: "xyzddlschema"
```

PostgreSQL に移行した後 Oracle テーブルが存在しない

この場合、通常、テーブルとデータには引き続きアクセスできます。

Oracle のデフォルトは大文字テーブル名ですが、PostgreSQL のデフォルトは小文字テーブル名です。Oracle から PostgreSQL に移行するときは、タスクのテーブル マッピングセクションの下に特定の変換ルールを指定することをお勧めします。これらは、テーブル名の大文字と小文字を変換するための変換ルールです。

テーブル名のケースを変換する変換ルールを使わずにテーブルを移行した場合、それらを参照するときテーブル名を引用符で囲みます。

ReplicationSlotDiskUsage ETL ワークロードなどの長いトランザクション中に が増加し、restart_lsn が進行しなくなる

論理レプリケーションが有効な場合、トランザクションあたりのメモリに保持される変更の最大数は 4MB です。その後、変更がディスクにこぼれます。その結果 ReplicationSlotDiskUsage が増加し、restart_lsnは、トランザクションが完了/中止され、ロールバックが完了するまで進みません。トランザクションが長いので、ロールバックに長時間かかることがあります。

従って、論理レプリケーションが有効になっている場合は、トランザクションの長期実行を避けてください。代わりに、トランザクションをいくつかの小さなトランザクションに分割してください。

ソースとしてビューを使用したタスクで行がコピーされない

ビューを移行するには、table-type を all または view に設定します。詳細については、「コンソールからテーブル選択および変換を指定する」を参照してください。

ビューをサポートするソースには、次のようなものがあります。

- Oracle
- Microsoft SQL Server
- MySQL
- PostgreSQL
- IBM Db2 LUW
- SAP Adaptive Server Enterprise (ASE)

Microsoft SQL Server 問題のトラブルシューティング

以下では、AWS DMS を Microsoft SQL Server データベースと使用する際に固有の問題のトラブルシューティングについて学習できます。

トピック

- SQL Server データベースの変更キャプチャエラー
- IDENTITY 列が存在しない
- エラー: SQL Server は公開をサポートしていません
- ターゲットに変更が表示されない

- パーティションにまたがってマッピングされる不均一テーブル

SQL Server データベースの変更キャプチャエラー

変更データ キャプチャ (CDC) 中のエラーは、前提条件の 1 つが満たされていないことを示している可能性があります。たとえば、最も一般的に見過ごされる前提条件は、完全データベースバックアップです。タスクログは以下のエラーで、この欠落を示します。

```
SOURCE_CAPTURE E: No FULL database backup found (under the 'FULL' recovery model).
To enable all changes to be captured, you must perform a full database backup.
120438 Changes may be missed. (sqlserver_log_queries.c:2623)
```

のソースとしての Microsoft SQL Server データベースの使用 AWS DMS で、SQL Server をソースとして使用するための前提条件を確認します。

IDENTITY 列が存在しない

AWS DMS では、ターゲットスキーマを作成する場合の IDENTITY 列がサポートされていません。初期ロードの完了後に追加する必要があります。

エラー: SQL Server は公開をサポートしていません

以下のエラーは、SQL Server Express をソースエンドポイントとして使用する際に生成されます。

```
RetCode: SQL_ERROR SqlState: HY000 NativeError: 21106
Message: This edition of SQL Server does not support publications.
```

AWS DMS では現在、ソースまたはターゲットとして SQL Server Express はサポートされていません。

ターゲットに変更が表示されない

AWS DMS では、変更を一貫してキャプチャするために、ソース SQL Server データベースが「FULL」または「BULK LOGGED」データ復旧モデルのいずれかであることが必要です。「SIMPLE」モデルはサポートされていません。

SIMPLE 復旧モデルは、ユーザーがデータベースを復旧するのに必要な最低限の情報を記録します。すべての非アクティブのログエントリは、チェックポイントが発生すると自動的に切り捨てられます。

すべてのオペレーションは引き続きログに記録されます。ただし、チェックポイントが発生するとすぐに、ログは自動的に切り捨てられます。この切り捨ては、ログが再利用可能になり、古いログエントリを上書きできることを意味します。ログエントリが上書きされると、変更をキャプチャできません。この問題は AWS DMS が、SIMPLE データ復旧モデルをサポートしていないのが理由です。SQL Server をソースとして使用するためのその他の前提条件については、「のソースとしての Microsoft SQL Server データベースの使用 AWS DMS」をご参照ください。

パーティションにまたがってマッピングされる不均一テーブル

変更データ キャプチャ (CDC) 中に、AWS DMS がテーブルで CDC を適切に実行できない場合、特殊な構造を持つテーブルの移行は中断されます。次のようなメッセージが発行されます。

```
[SOURCE_CAPTURE ]W: Table is not uniformly mapped across partitions. Therefore - it is
 excluded from CDC (sqlserver_log_metadata.c:1415)
[SOURCE_CAPTURE ]I: Table has been mapped and registered for CDC.
 (sqlserver_log_metadata.c:835)
```

SQL Server テーブルで CDC を実行する場合、AWS DMS は SQL Server の tlog を解析します。各 tlog レコードで、AWS DMS は変更中に挿入、更新、または削除されたカラムのデータを含む 16 進値を解析します。

16 進レコードを解析するために、AWS DMS は SQL Server システムテーブルからテーブルメタデータを読み取ります。これらのシステムテーブルは、特別に構造化されたテーブルカラムが何であるかを識別し、「xoffset」や「null bit position」などの内部プロパティの一部を表示します。

AWS DMS では、メタデータがテーブルのすべての未加工パーティションで同じであることを想定しています。しかし、場合によっては、特別に構造化されたテーブルのすべてのパーティションで同じメタデータが保持されないことがあります。この場合、AWS DMS は、そのテーブルの CDC を中断して、変更を誤って解析したり、ターゲットに誤ったデータを提供したりするのを防ぐことができます。回避策には、次のものが含まれます：

- テーブルにクラスター化されたインデックスがある場合は、インデックスの再構築を実行します。
- テーブルにクラスター化されたインデックスがない場合は、クラスター化されたインデックスをテーブルに追加します (必要に応じて後で削除できます)。

Amazon Redshift に関する問題のトラブルシューティング

以下では、AWS DMS を Amazon Redshift データベースと使用際に固有の問題のトラブルシューティングについて学習できます。

トピック

異なる AWS リージョンの Amazon Redshift クラスターにロードする

AWS DMS レプリケーション インスタンスとは異なる AWS リージョンの Amazon Redshift クラスターにロードすることはできません。DMS では、レプリケーション インスタンスと Amazon Redshift クラスターが同じリージョンにあることが前提です。

エラー: リレーション「awsdms_apply_exceptions」がすでに存在します

エラー "リレーション「awsdms_apply_exceptions」がすでに存在します" は、Redshift エンドポイントが PostgreSQL エンドポイントとして指定されている場合に頻繁に発生します。この問題を解決するには、エンドポイントを変更し、[Target engine] を「redshift」に変更します。

名前が「awsdms_changes」で始まるテーブルのエラー

名前が[awsdms_changes]で始まるテーブルに関連するエラーメッセージは、同一の Amazon Redshift クラスターにデータをロードしようとする 2 つのタスクが同時に実行される場合に頻繁に発生します。一時テーブルに名前が付けられる方法のため、同じテーブルを更新する場合、同時に実行されるタスクは競合する場合があります。

dms.awsdms_changes000000000XXXX のような名前のクラスターのテーブルを参照する

AWS DMS では、Amazon S3 に保存されるファイルからデータがロードされると、一時テーブルが作成されます。このような一時テーブルの名前には、プレフィックス `dms.awsdms_changes` が付

きます。AWS DMS では、データの初回ロード時と最終ターゲットテーブルへの保存前にデータを保存できるように、このようなテーブルが必要です。

Amazon Redshift での作業に必要なアクセス許可

Amazon Redshift で AWS DMS を使用するには、Amazon Redshift にアクセスするのに使うユーザーアカウントは、以下のアクセス許可を必要とします：

- CRUD (選択、挿入、更新、削除)
- [BULK Load] (一括ロード)
- CREATE、ALTER、DROP (タスクの定義によって必要な場合)

Amazon Redshift をターゲットとして使用するための前提条件を確認するには、「AWS Database Migration Service のターゲットとしての Amazon Redshift データベースの使用」をご参照ください。

Amazon Aurora MySQL に関する問題のトラブルシューティング

以下では、AWS DMS を Amazon Aurora MySQL データベースと使用する際に固有の問題のトラブルシューティングについて学習できます。

トピック

- エラー: CHARACTER SET UTF8 フィールドが「,」で切り取られています。行が「\n」で切り取られています

エラー: CHARACTER SET UTF8 フィールドが「,」で切り取られています。行が「\n」で切り取られています

ターゲットとして Amazon Aurora MySQL を使用している場合、ログに次のようなエラーが表示されることがあります。通常、このタイプのエラーは SQL_MODE パラメータの一部に ANSI_QUOTES があることを示しています。SQL_MODE パラメータの一部として ANSI_QUOTES がある場合、二重引用符が引用符として処理され、タスクの実行時に問題が起きることがあります。

このエラーを修正するには、SQL_MODE パラメータから ANSI_QUOTES を削除します。

```
2016-11-02T14:23:48 [TARGET_LOAD ]E: Load data sql statement. load data local infile
```

```
"/rdsdbdata/data/tasks/7XO4FJHCVON7TYTLQ6RX3CQHDU/data_files/4/LOAD000001DF.csv" into
 table
`VOSPUSER`.`SANDBOX_SRC_FILE` CHARACTER SET UTF8 fields terminated by ','
enclosed by '"' lines terminated by '\n'( `SANDBOX_SRC_FILE_ID`,`SANDBOX_ID`,
`FILENAME`,`LOCAL_PATH`,`LINES_OF_CODE`,`INSERT_TS`,`MODIFIED_TS`,`MODIFIED_BY`,
`RECORD_VER`,`REF_GUID`,`PLATFORM_GENERATED`,`ANALYSIS_TYPE`,`SANITIZED`,`DYN_TYPE`,
`CRAWL_STATUS`,`ORIG_EXEC_UNIT_VER_ID` ) ; (provider_syntax_manager.c:2561)
```

SAP ASE に関する問題のトラブルシューティング

以下では、AWS DMS を SAP ASE データベースと使用する際に固有の問題のトラブルシューティングについて学習できます。

エラー:ソースに NULL 値を持つコンポジット一意インデックスがある場合、LOB 列には NULL 値があります

NULL 値を許可する複合一意インデックスで構成されたテーブルで SAP ASE をソースとして使用すると、進行中のレプリケーション中に LOB 値が移行されないことがあります。通常、この動作は、DMS レプリケーション インスタンス クライアントで ANSI_NULL をデフォルトで 1 に設定した結果です。

LOB フィールドが正しく移行されるようにするには、タスクの AWS DMS ソースエンドポイントにエンドポイント設定 `'AnsiNull=0'` を含めます。

IBM Db2 に関する問題のトラブルシューティング

AWS DMS で IBM Db2 データベースを使用する場合に特有の問題のトラブルシューティングについての説明は次のとおりです。

エラー: Resume from timestamp is not supported Task

継続的なレプリケーション (CDC) で、特定のタイムスタンプからレプリケーションを開始する場合は、`StartFromContext` 接続属性を必要なタイムスタンプに設定する必要があります。詳細については、「Endpoint settings when using Db2 LUW」を参照してください。`StartFromContext` を必要なタイムスタンプに設定すると、次の問題が防止されます。

```
Last Error Resume from timestamp is not supported Task error notification received
 from
```

```
subtask 0, thread 0 [reptask/replicationtask.c:2822] [1020455] 'Start from timestamp'
 was blocked to prevent Replicate from
scanning the log (to find the timestamp). When using IBM DB2 for LUW, 'Start from
 timestamp' is only supported if an actual
change was captured by this Replicate task earlier to the specified timestamp.
```

AWS Database Migration Service のレイテンシーに関する問題のトラブルシューティング

このセクションでは、継続的なレプリケーションフェーズ (CDC) 中に AWS DMS タスクのレイテンシーが発生する一般的な原因の概要を説明します。AWS DMS はデータを非同期的にレプリケートします。レイテンシーは、変更がソースにコミットされてから、その変更がターゲットにレプリケートされるまでの遅延を指します。レイテンシーは、次のとおりのレプリケーションコンポーネントの設定ミスが原因で発生する可能性があります。

- ソースエンドポイントまたはデータソース
- ターゲットエンドポイントまたはデータソース
- レプリケーション インスタンス
- 上記コンポーネント間のネットワーク

レプリケーションに関する情報を収集するための概念実証としてテスト移行を使用することをお勧めします。その後、この情報を使用してレプリケーション設定を調整して、レイテンシーを最小限に抑えることができます。概念実証の移行の実行については、「PoC (概念実証) を実行する」を参照してください。

トピック

- CDC レイテンシーのタイプ
- CDC レイテンシーの一般的な原因
- レイテンシーに関する問題のトラブルシューティング

CDC レイテンシーのタイプ

このセクションでは、CDC 中に発生する可能性のあるレプリケーションレイテンシーのタイプについて説明します。

ソースのレイテンシー

ソースエンドポイントからキャプチャされた最後のイベントのコミット時刻とレプリケーションインスタンスの現在のシステムタイムスタンプ間のレイテンシー (秒単位)。CDCLatencySource CloudWatch メトリクスを使用して、データソースとレプリケーションインスタンス間のレイテンシーをモニタリングできます。CDCLatencySource メトリクスが高いと、ソースからの変更をキャプチャするプロセスが遅延していることを意味します。例えば、アプリケーションが 10:00 にソースへの挿入をコミットして、AWS DMS が 10:02 に変更を適用した場合、CDCLatencySource メトリクスは 120 秒になります。

の CloudWatch メトリクスの詳細についてはAWS DMS、「」を参照してくださいレプリケーションのタスクメトリクス。

ターゲットのレイテンシー

ターゲットへのコミットを待機している最初のイベントのソースでのコミット時間とDMS レプリケーションインスタンスの現在のタイムスタンプとの間のレイテンシー (秒単位)。CDCLatencyTarget CloudWatch メトリクスを使用して、データソースとデータターゲットのコミット間のレイテンシーをモニタリングできます。つまり、CDCLatencyTarget にはソースからの読み取り遅延も含まれます。その結果、CDCLatencyTarget は常に CDCLatencySource 以上になります。

例えば、アプリケーションが 10:00 にソースへの挿入をコミットして、AWS DMS が 10:05 にターゲットに書き込む場合、CDCLatencyTarget メトリクスは 300 秒になります。

CDC レイテンシーの一般的な原因

このセクションでは、CDC 中のレプリケーションで発生する可能性のあるレイテンシーの原因について説明します。

トピック

- エンドポイントのリソース
- レプリケーションインスタンスのリソース
- ネットワーク速度と帯域幅
- DMS の設定
- レプリケーションのシナリオ

エンドポイントのリソース

レプリケーションのパフォーマンスとレイテンシーに多大な影響を及ぼすのは次の要因です。

- ソースデータベースとターゲットデータベースの設定
- インスタンスサイズ
- ソースデータストアまたはターゲットデータストアがプロビジョニング不足または設定の誤り

AWSホストされたソースとターゲットのエンドポイントの問題によるレイテンシーの原因を特定するには、次の CloudWatch メトリクスをモニタリングします。

- `FreeMemory`
- `CPUUtilization`
- `WriteIOPS`、`WriteThroughput`、または `ReadLatency` などのスループットと I/O メトリクス
- `CDCIncomingChanges` などのトランザクションボリュームメトリクス

CloudWatch メトリクスのモニタリングについては、「」を参照してくださいAWS Database Migration Service のメトリクス。

レプリケーションインスタンスのリソース

レプリケーションインスタンスのリソースはレプリケーションに不可欠であり、ソースとターゲットの両方のレイテンシーにつながる可能性があるため、リソースのボトルネックがないことを確認する必要があります。

レプリケーションインスタンスのリソースのボトルネックを特定するには、次を確認します。

- CPU、メモリ、1 秒あたりの I/O、ストレージなどの重要な CloudWatch メトリクスでスパイクが発生していないか、値が大きくなっています。
- レプリケーションインスタンスのサイズがワークロードに応じて適切に設定されていること。レプリケーションインスタンスの適切なサイズを決定する方法の詳細については、「レプリケーションインスタンス向けの最適なサイズの選択」を参照してください。

ネットワーク速度と帯域幅

ネットワーク帯域幅は、データ送信に影響を与える要因となります。レプリケーションのネットワークパフォーマンスを分析するには、次のいずれかを実行します。

- インスタンスレベルで ReadThroughput メトリクスと WriteThroughput メトリクスを確認します。 CloudWatch メトリクスのモニタリングについては、「」を参照してくださいAWS Database Migration Service のメトリクス。
- AWS DMS Diagnostic Support AMI を使用します。Diagnostic Support AMI が現在のリージョンで利用できない場合は、サポートされている任意のリージョンからダウンロードして現在のリージョンにコピーすると、ネットワーク分析を実行できます。Diagnostic Support AMI の詳細については、「AWS DMS 診断サポート AMI の使用」を参照してください。

AWS DMS の CDC は、データの整合性が確保されるようにシングルスレッドとなっています。そのため、シングルスレッドのデータ転送速度を計算すると、ネットワークがサポートできるデータ量を決定できます。例えば、タスクが 100 Mbps (メガビット/秒) のネットワークを使用してソースに接続する場合、理論上、レプリケーションの最大帯域幅割り当ては 12.5 Mbps (メガバイト/秒) です。これは 1 時間あたり 45 ギガビットに相当します。ソースでのトランザクションログの生成速度が 45 ギガビット/時を超える場合は、タスクで CDC レイテンシーが発生することを意味します。100 Mbps のネットワークでは、上記の速度は理論上の最大値です。ネットワークトラフィックやソースとターゲットのリソースのオーバーヘッドなどのその他の要因によって、実際に使用できる帯域幅は減少します。

DMS の設定

このセクションでは、レイテンシーの軽減に役立つレプリケーションの推奨設定を説明します。

- エンドポイント設定: ソースエンドポイントとターゲットエンドポイント設定がレプリケーションインスタンスのパフォーマンス低下の原因となる場合があります。リソースを大量に利用する機能を有効にするエンドポイント設定は、パフォーマンスに影響します。例えば、Oracle エンドポイントの場合、 LogMiner はリソースを大量に消費するため、Binary Reader を無効にして使用する LogMiner とパフォーマンスが向上します。Oracle エンドポイントのパフォーマンスを向上するエンドポイント設定は次のとおりです。

  ```
  useLogminerReader=N;useBfile=Y
  ```

  エンドポイント設定の詳細については、「AWS DMS エンドポイントの使用」トピックのソースエンドポイントとターゲットエンドポイントエンジンのドキュメントを参照してください。
- タスク設定: 特定のレプリケーションシナリオのタスク設定によっては、レプリケーションインスタンスのパフォーマンスが低下する場合があります。例えば、AWS DMS は、Amazon Redshift 以外のすべてのエンドポイントの CDC にデフォルトでトランザクション適用

モード (BatchApplyEnabled=false) を使用します。ただし、変更が多数あるソースでは、BatchApplyEnabled を true に設定するとパフォーマンスが向上する可能性があります。

タスク設定の詳細については、「AWS Database Migration Service タスクのタスク設定の指定」をご参照ください。

- CDC のみのタスクの開始位置: 以前の位置またはタイムスタンプから CDC のみのタスクを開始すると、タスク開始時に CDC ソースのレイテンシーが増大します。ソースの変更量によっては、タスクのレイテンシーが低減するまで時間がかかります。
- LOB 設定: ラージオブジェクトデータ型の場合、ラージバイナリデータを AWS DMS がレプリケートする方法が原因でレプリケーションのパフォーマンスが低下する可能性があります。詳細については、次のトピックを参照してください。
  - AWS DMS タスクでのソースデータベースの LOB サポートの設定
  - ラージバイナリオブジェクト (LOB) の移行.

レプリケーションのシナリオ

このセクションでは、特定のレプリケーションシナリオと、レイテンシーへの影響について説明します。

トピック

- タスクの長時間にわたる停止
- キャッシュされた変更
- クロスリージョンレプリケーション

タスクの長時間にわたる停止

タスクを停止すると、AWS DMS はソースから読み取った最後のトランザクションログの位置を保存します。タスクを再開すると、DMS は同じトランザクションログ位置からの読み取りを続けようとします。タスクの再開が数時間または数日後の場合、DMS がトランザクションバックログの処理を完了するまで CDC ソースのレイテンシーが増大します。

キャッシュされた変更

キャッシュされた変更 は、AWS DMS がフルロードレプリケーションフェーズを実行している間にアプリケーションがデータソースに書き込む変更です。DMS は、フルロードフェーズが完了して CDC フェーズが開始されるまで、このような変更を適用しません。多数のトランザクションがあるソースの場合、キャッシュされた変更の適用には時間がかかるため、CDC フェーズが開始されると

ソースのレイテンシーが増加します。キャッシュされた変更数を最小限に抑えるため、フルロードフェーズは、トランザクション量が少ないケースで実行することをお勧めします。

クロスリージョンレプリケーション

DMS エンドポイントまたはレプリケーションインスタンスを別の AWS リージョンに配置すると、ネットワークレイテンシーが増大します。これにより、レプリケーションレイテンシーも長くなります。最高のパフォーマンスを得るには、ソースエンドポイント、ターゲットエンドポイント、AWS レプリケーションインスタンスを同じリージョンに配置します。

レイテンシーに関する問題のトラブルシューティング

このセクションでは、レプリケーションレイテンシーのトラブルシューティングの手順を説明します。

レイテンシーのトラブルシューティングを行うには、次を実行します。

- まず、タスクのレイテンシーのタイプと量を判断します。DMS コンソールまたは CLI でタスクのテーブル統計セクションを確認します。カウンターが変化している場合は、データ転送が進行中です。CDCLatencySource メトリクスと CDCLatencyTarget メトリクスを同時に確認して、CDC 中にボトルネックが発生していないかを判断します。
- CDCLatencySource メトリクスまたは CDCLatencyTarget メトリクスが高い値またはメトリクスがレプリケーションのボトルネックを示している場合は、次の点を確認します。
  - CDCLatencySource の値が高く、CDCLatencyTarget が CDCLatencySource に等しい場合、ソースエンドポイントでボトルネックが発生し、AWS DMS はデータをターゲットに円滑に書き込んでいることを示しています。次の「ソースのレイテンシーに関する問題のトラブルシューティング」を参照してください。
  - CDCLatencySource が低く、CDCLatencyTarget が高い場合、ターゲットエンドポイントでボトルネックが発生し、AWS DMS はソースからはデータを円滑に読み取っていることを示しています。次の「ターゲットのレイテンシーに関する問題のトラブルシューティング」を参照してください。
  - CDCLatencySource が高く、CDCLatencySource が CDCLatencyTarget よりも大幅に高い場合、ソース読み取りとターゲット書き込みの両方でボトルネックが発生していることを示しています。まずソースのレイテンシーを調べてから、ターゲットのレイテンシーを調査します。

DMS タスクメトリクスのモニタリングの詳細については、「AWS DMS タスクのモニタリング」を参照してください。

ソースのレイテンシーに関する問題のトラブルシューティング

次のトピックでは、ソースエンドポイントタイプに固有のレプリケーションシナリオについて説明します。

トピック

- Oracle エンドポイントのトラブルシューティング
- MySQL エンドポイントのトラブルシューティング
- PostgreSQL エンドポイントのトラブルシューティング
- SQL Server エンドポイントのトラブルシューティング

Oracle エンドポイントのトラブルシューティング

このセクションでは、Oracle に固有のレプリケーションシナリオについて説明します。

ソースの読み取りが停止した

AWS DMS は、次のシナリオで Oracle ソースからの読み取りを停止します。この動作は仕様です。原因はタスクログを使用して調査できます。タスクログで次のメッセージを調べます。タスクログのオペレーションの詳細については、「AWS DMS タスクログの表示と管理」を参照してください。

- SORTER メッセージ: DMS がレプリケーションインスタンスにトランザクションをキャッシュしていることを示します。詳細については、「タスクログの SORTER メッセージ」を参照してください。
- デバッグタスクログ: DMS が読み取りプロセスを中断した場合、タスクはコンテキストフィールドやタイムスタンプを変更せずに、次のメッセージをデバッグタスクログに繰り返し書き込みます。
  - Binary Reader:

```
[SOURCE_CAPTURE  ]T:  Produce CTI event:
context '00000020.f23ec6e5.00000002.000a.00.0000:190805.3477731.16'
xid [00000000001e0018] timestamp '2021-07-19 06:57:55'
thread 2  (oradcdc_oralog.c:817)
```

  - Logminer:

```
[SOURCE_CAPTURE  ]T:  Produce INSERT event:
object id 1309826 context
 '000000000F2CECAA010000010005A8F500000275016C0000000000000F2CEC58'
```

```
xid [000014e06411d996] timestamp '2021-08-12 09:20:32' thread 1
 (oracdc_reader.c:2269)
```

- AWS DMS は、新しい REDO またはアーカイブされたログオペレーションごとに次のメッセージを記録します。

```
00007298: 2021-08-13T22:00:34 [SOURCE_CAPTURE ]I: Start processing archived
 Redo log sequence 14850 thread 2 name XXXXX/XXXXX/ARCHIVELOG/2021_08_14/
thread_2_seq_14850.22977.1080547209 (oradcdc_redo.c:754)
```

  ソースに新しい REDO オペレーションまたはアーカイブされたログオペレーションがあり、AWS DMS がこのようなメッセージをログに書き込んでいない場合は、タスクがイベントを処理していないことを意味します。

大量の REDO 生成

タスクが REDO ログまたはアーカイブされたログを処理しているものの、ソースのレイテンシーが依然として高い場合は、REDO ログの生成速度と生成パターンを特定します。REDO ログの生成レベルが高い場合は、レプリケートしたテーブルに関連する変更をフェッチするタスクが REDO ログとアーカイブログをすべて読み取るため、ソースのレイテンシーが増大します。

REDO 生成率を判断するには、次のクエリを使用します。

- 1 日あたりの REDO 生成率:

```
select trunc(COMPLETION_TIME,'DD') Day, thread#,
round(sum(BLOCKS*BLOCK_SIZE)/1024/1024/1024) GB,
count(*) Archives_Generated from v$archived_log
where completion_time > sysdate- 1
group by trunc(COMPLETION_TIME,'DD'),thread# order by 1;
```

- 1 時間あたりの REDO 生成率:

```
Alter session set nls_date_format = 'DD-MON-YYYY HH24:MI:SS';
select trunc(COMPLETION_TIME,'HH') Hour,thread# ,
round(sum(BLOCKS*BLOCK_SIZE)/1024/1024) "REDO PER HOUR (MB)",
count(*) Archives from v$archived_log
where completion_time > sysdate- 1
group by trunc(COMPLETION_TIME,'HH'),thread#  order by 1 ;
```

このシナリオでレイテンシーのトラブルシューティングを行うには、次の点を確認します。

- レプリケーションのネットワーク帯域幅とシングルスレッドのパフォーマンスを調べて、基盤となるネットワークがソースの REDO 生成速度をサポートできることを確認します。ネットワーク帯域幅によるレプリケーションのパフォーマンスへの影響の詳細については、前の「ネットワーク速度と帯域幅」を参照してください。
- 補足ログが適切に設定されているかを確認します。テーブルのすべての列でログを有効にするなど、ソースでの余分なログ記録は避けます。補足ログの設定の詳細については、「サプリメンタルロギングの設定」を参照してください。
- REDO ログまたはアーカイブされたログを読み取るために適切な API を使用していることを確認します。Oracle LogMiner または AWS DMS Binary Reader のいずれかを使用できます。はオンライン REDO ログとアーカイブ REDO ログファイル LogMiner を読み込みますが、Binary Reader は raw REDO ログファイルを直接読み取り、解析します。このため、Binary Reader を使用するとパフォーマンスが向上します。REDO ログの生成が 1 時間あたり 10 GBを超える場合は、Binary Reader を使用することをお勧めします。詳細については、「CDC での Oracle LogMiner または AWS DMS Binary Reader の使用」を参照してください。
- `ArchivedLogsOnly` を Y に設定しているかを確認します。このエンドポイント設定が指定されている場合、AWS DMS はアーカイブされた REDO ログから読み取ります。AWS DMS はオンライン REDO ログがアーカイブされるまで待機してから読み取りを行うため、これによりソースのレイテンシーが増大します。詳細については、「」を参照してくださいArchivedLogsOnly。
- Oracle ソースで自動ストレージ管理 (ASM) を使用している場合にデータストアを適切に設定する方法については、「のソースとして Oracle を使用する場合の Oracle ASM への REDO の保存 AWS DMS」を参照してください。追加接続属性(ECA) `asmUsePLSQLArray` を使用して、読み取りパフォーマンスをさらに最適化できる場合もあります。`asmUsePLSQLArray` の使用の詳細については、「のソースとして Oracle を使用する場合のエンドポイント設定 AWS DMS」を参照してください。

MySQL エンドポイントのトラブルシューティング

このセクションでは、MySQL 固有のレプリケーションシナリオについて説明します。AWS DMS は、MySQL バイナリログを定期的にスキャンして変更をレプリケートします。このプロセスにより、次のシナリオでレイテンシーが増大する可能性があります。

トピック

- ソースでのトランザクション実行時間が長い
- ソースでの高ワークロード

- バイナリログの競合

ソースでのトランザクション実行時間が長い

MySQL はコミットされたトランザクションのみをバイナリログに書き込むため、トランザクションが長時間実行されると、クエリの実行時間に比例してレイテンシーのスパイクが発生します。

実行時間の長いトランザクションを特定するには、次のクエリを使用するか、スロークエリログを使用します。

```
SHOW FULL PROCESSLIST;
```

スロークエリログの使用については、「MySQL ドキュメント」の「The Slow Query Log」を参照してください。

長時間実行されるトランザクションによるレイテンシーのスパイクを回避するには、ソーストランザクションを再構築してクエリの実行時間を短縮するか、コミット頻度を増やします。

ソースでの高ワークロード

DMS CDC はシングルスレッドであるため、トランザクション数が多いとソースのレイテンシーが増大する可能性があります。ワークロードが重いことがソースのレイテンシーの原因であるかを特定するには、レイテンシー中に生成されたバイナリログの数とサイズをレイテンシー以前に生成されたログと比較します。バイナリログと DMS CDC スレッドのステータスを確認するには、次のクエリを使用します。

```
SHOW BINARY LOGS;
SHOW PROCESSLIST;
```

CDC バイナリログのダンプスレッドの状態の詳細については、「Replication Source Thread States」を参照してください。

ソースで生成された最新のバイナリログの位置と DMS が現在処理しているイベントを比較すると、レイテンシーを判断できます。ソースの最新のバイナリログを特定するには、次を実行します。

- SOURCE_CAPTURE コンポーネントのデバッグログを有効にします。
- DMS の処理のバイナリログと位置の詳細をタスクデバッグログから取得します。
- 次のクエリを使用して、ソースの最新のバイナリログを特定します。

```
SHOW MASTER STATUS;
```

パフォーマンスをさらに最適化するには、EventsPollInterval を調整します。デフォルトでは、DMS は 5 秒ごとにバイナリログをポーリングします。この値を減らすとパフォーマンスが向上する可能性があります。EventsPollInterval の設定の詳細については、MySQL をソースとして使用する場合のエンドポイント設定 AWS DMSを参照してください。

バイナリログの競合

大量のデータを含む複数のテーブルを移行する場合、MySQL 5.7.2 以降ではテーブルを個別のタスクに分割することをお勧めします。MySQL バージョン 5.7.2 以降では、マスターダンプスレッドによるロック競合が低減し、スループットが向上しています。その結果、ダンプスレッドはイベントを読み取る都度バイナリログをロックすることがなくなりました。つまり、複数のダンプスレッドがバイナリログファイルを同時に読み取ることができるようになりました。これは、クライアントがバイナリログに書き込みを行っている間も、ダンプスレッドがバイナリログを読み取ることができることも意味します。ダンプスレッドの詳細については、「Replication Threads」と「MySQL 5.7.2 リリースノート」を参照してください。

5.7.2 以前のバージョンの MySQL ソースのレプリケーションのパフォーマンスを向上するには、CDC コンポーネントを使用してタスクを統合することを検討します。

PostgreSQL エンドポイントのトラブルシューティング

このセクションでは、PostgreSQL に固有のレプリケーションシナリオについて説明します。

トピック

- ソースでのトランザクション実行時間が長い
- ソースでの高ワークロード
- 高いネットワークスループット
- Aurora PostgreSQL のスピルファイル

ソースでのトランザクション実行時間が長い

ソースデータベースで長時間実行されるトランザクション (単一トランザクションでの数千回の挿入など) がある場合、トランザクションが完了するまで DMS CDC イベントカウンターとトランザクションカウンターは増加しません。この遅延は、レイテンシーの問題につながる可能性があり、CDCLatencyTarget メトリクスで測定できます。

長時間実行されているトランザクションを確認するには、次のいずれかを実行します。

- pg_replication_slots ビューを使用します。restart_lsn 値が更新されない場合、アクティブなトランザクションが長時間実行されているため、PostgreSQL は ログ先行書き込み (WAL) を解放できない可能性があります。pg_replication_slots ビューの詳細については、「PostgreSQL 15.4 ドキュメント」の「pg_replication_slots」を参照してください。
- 次のクエリを使用すると、データベース内のすべてのアクティブなクエリのリストと関連情報が返されます。

```
SELECT pid, age(clock_timestamp(), query_start), usename, query
FROM pg_stat_activity WHERE query != '<IDLE>'
AND query NOT ILIKE '%pg_stat_activity%'
ORDER BY query_start desc;
```

  クエリ結果の age フィールドには、各クエリのアクティブ期間が表示されます。長時間実行されているクエリは、これを使用して特定できます。

ソースでの高ワークロード

ソースの PostgreSQL のワークロードが高い場合は、次の点を確認してレイテンシーを低減します。

- 1 秒あたりのトランザクション (TPS) 値が高いソースデータベースからテーブルのサブセットを移行する際に test_decoding プラグインを使用すると、レイテンシーが増大する可能性があります。これは、test_decoding プラグインがデータベースのすべての変更をレプリケーションインスタンスに送信し、DMS がタスクのテーブルマッピングに基づいてフィルタリングするためです。タスクのテーブルマッピングに含まれていないテーブルのイベントは、ソースレイテンシーを増大につながる可能性があります。
- 次のいずれかの方法を使用して TPS スループットを確認します。
  - Aurora PostgreSQL ソースの場合は、 CommitThroughput CloudWatch メトリクスを使用します。
  - Amazon RDS またはオンプレミスで実行されている PostgreSQL の場合は、バージョン 11 以降の PSQL クライアントを使用して次のクエリを実行します (クエリ中に **enter** を押すと結果の表示を先に進めることができます)。

```
SELECT SUM(xact_commit)::numeric as temp_num_tx_ini FROM pg_stat_database; \gset
select pg_sleep(60);
SELECT SUM(xact_commit)::numeric as temp_num_tx_final FROM pg_stat_database; \gset
```

```
select (:temp_num_tx_final - :temp_num_tx_ini)/ 60.0 as "Transactions Per Second";
```

- test_decoding プラグインを使用する際のレイテンシーを低減するには、代わりに pglogical プラグインの使用を検討します。test_decoding プラグインとは異なり、pglogical プラグインはソースでログ先行書き込み (WAL) の変更をフィルターして、関連する変更のみをレプリケーションインスタンスに送信します。AWS DMS で pglogical プラグインを使用する方法については、「pgglogical プラグインの設定」を参照してください。

高いネットワークスループット

test_decoding プラグインを使用すると、特に大量のトランザクションがある場合、レプリケーションのネットワーク帯域幅の使用量が増大する可能性があります。これは、test_decoding プラグインが変更を処理し、元のバイナリ形式よりもサイズが大きくなる判別しやすい形式に変換するためです。

パフォーマンスを向上するには、代わりにバイナリプラグインの pglogical プラグインの使用を検討します。test_decoding プラグインとは異なり、pglogical プラグインはバイナリ形式の出力を生成するため、圧縮されたログ先行書き込み (WAL) ストリーム変更が作成されます。

Aurora PostgreSQL のスピルファイル

PostgreSQL バージョン 13 以降では、 logical_decoding_work_memパラメータによってデコードとストリーミングのメモリ割り当てが決まります。logical_decoding_work_mem パラメータの詳細については、PostgreSQL 13.13 ドキュメントの「PostgreSQL でのリソース消費」を参照してください。 PostgreSQL

論理レプリケーションは、すべてのトランザクションの変更をメモリ内のトランザクションがコミットされるまで蓄積します。すべてのトランザクションに保存されているデータの量がデータベースパラメータ で指定された量を超えるとlogical_decoding_work_mem、DMS はトランザクションデータをディスクにスピルして新しいデコードデータのメモリを解放します。

トランザクションの実行時間が長い、またはサブトランザクションが多いと、DMS が論理デコードメモリを消費する可能性があります。このメモリ使用量の増加により、DMS はディスク上にスピルファイルを作成し、レプリケーション中のソースレイテンシーが大きくなります。

ソースワークロードの増加による影響を減らすには、次の操作を行います。

- 長時間実行されるトランザクションを減らします。
- サブトランザクションの数を減らします。

- 1 つのトランザクションでテーブル全体を削除または更新するなど、大量のログレコードを生成するオペレーションは実行しないでください。代わりに小さなバッチで操作を実行します。

次の CloudWatch メトリクスを使用して、ソースのワークロードをモニタリングできます。

- `TransactionLogsDiskUsage`: 論理 WAL によって現在占有されているバイト数。この値は、論理レプリケーションスロットが新しい書き込みのペースに追いつくことができない場合、または長時間実行されるトランザクションが古いファイルのガベージコレクションを妨げる場合、一定間隔で増加します。
- `ReplicationSlotDiskUsage`: 論理レプリケーションスロットが現在使用しているディスク容量。

`logical_decoding_work_mem` パラメータを調整することで、ソースのレイテンシーを低減できます。このパラメータのデフォルト値は 64 MB です。このパラメータは、各論理ストリーミングレプリケーション接続で使用されるメモリの量を制限します。DMS がディスクに書き込むデコードされた変更の量を減らすために、 `logical_decoding_work_mem`値を `work_mem`値よりも大幅に高く設定することをお勧めします。

特に移行アクティビティやレイテンシーが重い時間帯は、スピルファイルを定期的に確認することをお勧めします。DMS が大量のスピルファイルを作成している場合、論理デコードが効率的に機能せず、レイテンシーが増加する可能性があります。これを軽減するには、`logical_decoding_work_mem`パラメータ値を増やします。

現在のトランザクションオーバーフローは、 `aurora_stat_file`関数で確認できます。詳細については、「Amazon Relational Database Service デベロッパーガイド」の「論理デコードの作業メモリの調整」を参照してください。

SQL Server エンドポイントのトラブルシューティング

このセクションでは、SQL Server に固有のレプリケーションシナリオについて説明します。SQL Server からレプリケートする変更を決定するには、AWS DMS トランザクションログを読み取り、ソースデータベースで定期的にスキャンを実行します。レプリケーションのレイテンシーは通常、リソースの制約により SQL Server がこれらのスキャンをスロットリングすることが原因で発生します。また、短時間でトランザクションログに書き込まれるイベントの数が大幅に増加したことが原因である場合もあります。

トピック

- インデックスの再構築
- 大きいトランザクション
- Amazon RDS SQL Server の MS-CDC ポーリング間隔が適切に設定されていない
- 同じソースデータベースからレプリケートする複数の CDC タスク

インデックスの再構築

サイズが大きいインデックスを再構築する場合、SQL Server は単一のトランザクションを使用します。これにより大量のイベントが生成されます。SQL Server が複数のインデックスを一度に再構築すると、大量のログ領域が使用される可能性があります。このような状況が生じた場合、レプリケーションで短期間のスパイクが発生することが予想されます。SQL Server ソースでログのスパイクが継続して発生している場合は、次の点を確認します。

- まず、 `CDCLatencySource`および `CDCLatencySource` CloudWatch メトリクスを使用するか、タスクログのスループットモニタリングメッセージをチェックして、レイテンシースパイクの時間を確認します。の CloudWatch メトリクスの詳細についてはAWS DMS、「」を参照してくださいレプリケーションのタスクメトリクス。
- レイテンシーのスパイク中にアクティブなトランザクションログまたはログバックアップのサイズが増加したかを確認します。この間にメンテナンスジョブや再構築が実行されたかも確認します。トランザクションログのサイズの確認の詳細については、「SQL Server 技術ドキュメント」の「ログ領域の使用量の監視」を参照してください。
- メンテナンスプランが SQL Server のベストプラクティスに従っていることを確認します。SQL Server メンテナンスのベストプラクティスについては、「SQL Server 技術ドキュメント」の「インデックスのメンテナンスの戦略」を参照してください。

インデックスの再構築中のレイテンシーの問題を解決するには、次を試します。

- オフラインの再構築に `BULK_LOGGED` 復旧モデルを使用して、タスクが処理する必要があるイベントを減らします。
- 可能な場合は、インデックスの再構築中はタスクを停止します。または、レイテンシーのスパイクの影響を軽減するために、インデックスの再構築をピーク時以外の時間帯でスケジュールします。
- ディスクのレイテンシーや I/O スループットなど、DMS の読み取りを遅延させるリソースのボトルネックを特定し、対処します。

大きいトランザクション

大量のイベントがあるトランザクションや実行時間の長いトランザクションにより、トランザクションログのサイズが増大します。これにより DMS の読み取りに時間がかかり、レイテンシーが発生します。これは、インデックスの再構築がレプリケーションのパフォーマンスに与える影響と類似しています。

ソースデータベースの一般的なワークロードに慣れていない場合、この問題を特定するのが難しい場合があります。このエラーを解決するには、次を実行します。

- まず、 `ReadThroughput`および `WriteThroughput` CloudWatch メトリクスを使用するか、タスクログのスループットモニタリングメッセージをチェックして、レイテンシーが急増した時間を特定します。
- レイテンシーのスパイク中にソースデータベースで長時間実行されているクエリがないかを確認します。実行時間の長いクエリの詳細については、「SQL Server 技術ドキュメント」の「SQL Serverでの実行時間の遅いクエリのトラブルシューティング」を参照してください。
- アクティブなトランザクションログまたはログバックアップのサイズが増加していないかを確認します。詳細については、「SQL Server 技術ドキュメント」の「ログ領域の使用量の監視」を参照してください。

この問題を解決するには、次のいずれかを実行します。

- トランザクションが迅速に完了するようにアプリケーション側でトランザクションを再構築することが、最善の解決策です。
- トランザクションを再構築できない場合の短期的な回避策は、ディスク待機や CPU 競合などのリソースのボトルネックがないかを確認することです。ソースデータベースにボトルネックが見つかった場合は、ソースデータベースのディスク、CPU、メモリのリソースを増やすとレイテンシーを短縮できます。これにより、システムリソースの競合が低減し、DMS クエリの完了を迅速化できます。

Amazon RDS SQL Server の MS-CDC ポーリング間隔が適切に設定されていない

Amazon RDS インスタンスのポーリング間隔の設定に誤りがあると、トランザクションログが増大する可能性があります。これは、レプリケーションがログの切り捨てを回避するためです。実行中のタスクは最小限のレイテンシーでレプリケーションを続行する可能性があるとはいえ、タスクの停止と再開、または CDC のみのタスクの開始によりタスクが失敗する可能性があります。これは、サイズの大きいトランザクションログのスキャン中のタイムアウトが原因です。

ポーリング間隔の設定が適切でない場合のトラブルシューティングを行うには、次を実行します。

- アクティブなトランザクションログのサイズが増えているか、ログの使用率が 100% に近づいているかを確認します。詳細については、「SQL Server 技術ドキュメント」の「ログ領域の使用量の監視」を参照してください。
- ログの切り捨てが遅延していないかを REPLICATION の `log_reuse_wait_desc value` で確認します。詳細については、「SQL Server 技術ドキュメント」の「トランザクション ログ (SQL Server)」を参照してください。

上記のリストの項目のいずれかで問題が見つかった場合は、MS-CDC ポーリング間隔を調整します。ポーリング間隔の調整については、「のソースとして Amazon RDS for SQL Server を使用する場合の推奨設定 AWS DMS」を参照してください。

同じソースデータベースからレプリケートする複数の CDC タスク

フルロードフェーズでは、パフォーマンス向上、依存テーブルの論理的分離、タスク障害の影響の軽減のために、テーブルをタスク間で分割することをお勧めします。ただし、CDC フェーズでは、DMS スキャンを最小限に抑えるためにタスクを統合することをお勧めします。CDC フェーズでは、各 DMS タスクが 1 分あたり数回トランザクションログをスキャンして新しいイベントを検索します。各タスクは独立して実行されるため、各タスクは各トランザクションログを個別にスキャンします。これにより、ソースの SQL Server データベースのディスクと CPU の使用量が増加します。この結果、多数のタスクが並列実行されると、SQL Server が DMS の読み取りのスロットリングを行い、レイテンシーが増大する可能性があります。

複数のタスクが徐々に開始されると、この問題を特定するのが難しくなる可能性があります。この問題の最も一般的な兆候は、ほとんどのタスクスキャンに時間がかかるようになっていくことです。これにより、このようなスキャンのレイテンシーが増大します。SQL Server ではいくつかのタスクスキャンが優先されるため、少数のタスクによっては通常のレイテンシーが表示されます。この問題を解決するには、すべてのタスクの `CDCLatencySource` メトリクスを確認します。タスクによっては `CDCLatencySource` が増加しており、`CDCLatencySource` が減少しているタスクもいくつかある場合は、SQL Server が一部のタスクの DMS 読み取りをスロットリングしている可能性があります。

SQL Server が CDC 中にタスクの読み取りをスロットリングしている場合は、タスクを統合して DMS スキャンの数を最小限に抑えます。競合を発生させずにソースデータベースに接続できるタスクの最大数は、ソースデータベースのキャパシティ、トランザクションログの増加率、テーブル数などの要因によって異なります。レプリケーションシナリオに最適なタスク数を決定するには、本番環境と同様のテスト環境でレプリケーションをテストします。

ターゲットのレイテンシーに関する問題のトラブルシューティング

このセクションでは、ターゲットのレイテンシーの要因となるシナリオを説明しています。

トピック

インデックス作成の問題

CDC フェーズでは、AWS DMS はターゲットで DML ステートメント (insert、update、delete) を実行することにより、ソースの変更をレプリケートします。DMS を使用する異種移行の場合、ソースとターゲットのインデックス最適化の違いにより、ターゲットへの書き込みに時間がかかる場合があります。これは、ターゲットのレイテンシーとパフォーマンスの問題が発生する原因となります。

インデックス作成の問題のトラブルシューティングを行うには、次を実行します。このようなステップの手順は、データベースエンジンにより異なります。

- ターゲットデータベースのクエリ時間をモニタリングします。ターゲットとソースのクエリ実行時間を比較すると、最適化が必要なインデックスが判明します。
- 実行速度が遅いクエリのログ記録を有効にします。

長時間実行されるレプリケーションのインデックス作成の問題を修正するには、次を実行します。

- クエリの実行時間がソースとターゲットで同等になるように、ソースデータベースとターゲットデータベースのインデックスを調整します。
- ソースとターゲットの DML クエリで使用されるセカンダリインデックスを比較します。ターゲットの DML パフォーマンスがソースの DML パフォーマンスと同等かそれ以上であることを確認します。

インデックスを最適化するプロシージャはデータベースエンジンに固有であることに注意します。ソースとターゲットのインデックスを調整するための DMS 機能はありません。

タスクログの SORTER メッセージ

ターゲットエンドポイントが、AWS DMS が書き込む変更の量に遅れを取る場合、タスクは変更をレプリケーションインスタンスにキャッシュします。キャッシュが内部しきい値を超えると、タスクはそれ以上のソースからの変更の読み取りを停止します。DMS は、レプリケーションインスタンスでのストレージ不足や、大量の保留中のイベントを読み取っている際のタスクのスタックを避けるためにこれを実行します。

この問題のトラブルシューティングを行うには、次のいずれかのメッセージがないか CloudWatch ログを確認してください。

```
[SORTER ]I: Reading from source is paused. Total disk usage exceeded the limit 90%
 (sorter_transaction.c:110)
[SORTER ]I: Reading from source is paused. Total storage used by swap files exceeded
 the limit 1048576000 bytes  (sorter_transaction.c:110)
```

ログに最初のメッセージと同様のメッセージが含まれている場合は、タスクのトレースログを無効にして、レプリケーションインスタンスのストレージを増やします。レプリケーションインスタンスのストレージを増やす方法については、「レプリケーション インスタンスの変更」を参照してください。

ログに 2 番目のメッセージと同様のメッセージが含まれている場合は、次を実行します。

- 多数のトランザクションまたは長時間実行される DML オペレーションを含むテーブルは、タスクでその他のテーブルへの依存関係がない場合、別のタスクに移動します。
- トランザクションをメモリに保持する時間を延長するように、`MemoryLimitTotal` と `MemoryKeepTime` の設定を増やします。この方法は、レイテンシーが持続する場合には役に立たないとはいえ、トランザクション量が短時間集中している間のレイテンシーを低減できます。上記のタスク設定の詳細については、「変更処理のチューニング設定」を参照してください。
- `BatchApplyEnabled` を `true` に設定して、トランザクションにバッチ適用を使用できるかどうかを評価します。`BatchApplyEnabled` 設定の詳細については、「ターゲットメタデータのタスク設定」を参照してください。

データベースのロック

AWS DMS がレプリケーションターゲットとして使用しているデータベースにアプリケーションがアクセスすると、DMS がアクセスしようとしているテーブルをアプリケーションがロックする場合があります。これにより、ロックの競合が発生します。DMS は、ソースで発生した順序でターゲッ

トデータベースに変更を書き込むため、ロック競合が原因で単一のテーブルへの書き込みが遅延すると、すべてのテーブルへの書き込みの遅延が発生します。

この問題のトラブルシューティングを行うには、ターゲットデータベースにクエリを実行して、ロック競合が DMS 書き込みトランザクションをブロックしていないかを確認します。ターゲットデータベースが DMS 書き込みトランザクションをブロックしている場合は、次の単一または複数の手順を実行します。

- クエリを再構築して、より頻繁に変更をコミットします。
- ロックのタイムアウト設定を変更します。
- テーブルをパーティション分裂して、ロック競合を最小限に抑えます。

ロックの競合を最適化するプロシージャは、データベースエンジンによって異なることに注意します。ロックの競合を調整するための DMS 機能はありません。

LOB ルックアップが遅い

ラージオブジェクト (LOB) 列のレプリケートの際、AWS DMS はターゲットに変更を書き込む直前にソースのルックアップを実行します。通常、このルックアップによってターゲットにレイテンシーが発生することはありません。ただし、ソースデータベースでのロックでルックアップが遅延した場合、ターゲットのレイテンシーのスパイクが発生する可能性があります。

この問題を診断するのは通常、困難です。この問題のトラブルシューティングを行うには、タスクログで詳細なデバッグを有効にして、DMS LOB ルックアップコールのタイムスタンプを比較します。詳細なデバッグを有効にする方法の詳細については、「AWS DMS タスクログの表示と管理」を参照してください。

この問題を解決するには、次を試します。

- ソースデータベースの SELECT クエリのパフォーマンスを向上します。
- DMS LOB 設定を調整します。LOB 設定の調整の詳細については、「ラージバイナリオブジェクト (LOB) の移行」を参照してください。

マルチ AZ、監査ログ、バックアップ

Amazon RDS ターゲットの場合、ターゲットレイテンシーは次の期間に増大する可能性があります。

- バックアップ

- 複数のアベイラビリティゾーン (マルチ AZ) を有効にした後
- 監査ログやスロークエリログなどのデータベースログ記録を有効にした後。

このような問題を診断するのは通常、困難です。このような問題のトラブルシューティングを行うには、Amazon RDS のメンテナンス期間やデータベースの負荷が高い期間に周期的にスパイクが発生しないか、レイテンシーをモニタリングします。

このような問題を修正するには、次を実行します。

- 可能な場合、短期間の移行中は、マルチ AZ、バックアップ、またはロギングを無効にします。
- メンテナンスの時間帯をアクティビティが少ない時間帯に再スケジュールします。

AWS DMS での診断サポート スクリプトの操作

AWS DMS での作業中に問題が発生した場合は、サポート エンジニアがソースデータベースまたはターゲットデータベースに関する詳細情報を必要とする場合があります。AWS Support が可能な限り短時間で必要な情報を可能な限り多く取得できるようにします。そのため、いくつかの主要なリレーショナル データベースエンジンでこの情報を照会するスクリプトを開発しました。

サポートスクリプトがデータベースで使用できる場合は、後述する対応するスクリプトトピックのリンクを使用して、サポートスクリプトをダウンロードできます。スクリプトを検証して確認した後 (以下で説明)、スクリプトのトピックで説明されている手順に従ってスクリプトを実行できます。スクリプトの実行が完了したら、その出力を AWS Support ケース (繰り返しますが、以下で説明します)。

スクリプトを実行する前に、サポートスクリプトをダウンロード時または保存するときに生じた可能性があるエラーを検出できます。これを行うには、スクリプトファイルのチェックサムを AWS から取得した値と比較します。AWS では、チェックサムに SHA256 アルゴリズムを使用します。

チェックサムを使用してサポート スクリプトファイルを検証するには

1. 提供された最新のチェックサム ファイルを開き、https://d2pwp9zz55emqw.cloudfront.net/sha256Check.txt のサポート スクリプトを確認します。例えば、ファイルに次のようなコンテンツがあるとします。

```
MYSQL  dfafd0d511477c699f96c64693ad0b1547d47e74d5c5f2f2025b790b1422e3c8
ORACLE  6c41ebcfc99518cfa8a10cb2ce8943b153b2cc7049117183d0b5de3d551bc312
POSTGRES  6ccd274863d14f6f3146fbdbbba43f2d8d4c6a4c25380d7b41c71883aa4f9790
```

```
SQL_SERVER  971a6f2c46aec8d083d2b3b6549b1e9990af3a15fe4b922e319f4fdd358debe7
```

2. Support ファイルが含まれているディレクトリで、オペレーティングシステムの SHA256 検証コマンドを実行します。例えば、macOS オペレーティングシステムでは、このトピックで後述する Oracle サポート スクリプトに対して次のコマンドを実行できます。

```
shasum -a 256 awsdms_support_collector_oracle.sql
```

3. コマンドの結果を、最新の開いた `sha256Check.txt` ファイルに表示されている値と比較します。2 つの値は一致する必要があります。そうでない場合は、不一致と、クリーンな Support スクリプトファイルの入手方法について、サポートエンジニアに連絡してください。

クリーンな Support スクリプトファイルがある場合は、スクリプトを実行する前に、パフォーマンスとセキュリティの両方の観点から SQL を読み、理解してください。このスクリプトで任意の SQL を楽に実行しにくければ、問題の SQL をコメントアウトするか、削除することができます。許容可能な回避策については、サポートエンジニアに相談することもできます。

正常に完了すると、特に指定されていない限り、スクリプトは読み取り可能な HTML 形式で出力を返します。このスクリプトは、ビジネスを危険にさらす可能性のあるデータやセキュリティの詳細をこの HTML からは除外するように設計されています。また、データベースやその環境は変更されません。ただし、HTML 内に共有したくない情報がある場合は、HTML をアップロードする前に該当の情報を削除してください。出力ファイルが許容できる内容となったら、サポートケースの[ケースの詳細] にある[添付ファイル] を使用して出力ファイルをアップロードします。

次の各トピックでは、サポートされている AWS DMS データベースで使用できるスクリプトとその実行方法について説明します。サポートエンジニアは、以下に記載された特定のスクリプトドキュメントを参照するよう指示します。

トピック

- Oracle 診断 Support スクリプト
- SQL Server 診断 Support スクリプト
- MySQL 互換データベースの診断サポート スクリプト
- PostgreSQL 診断 Support スクリプト

Oracle 診断 Support スクリプト

次に、 AWS DMS マイグレーション設定でオンプレミスデータベースまたは Amazon RDS for Oracle Database の分析に使用できる診断サポート スクリプトについて説明します。これらのスクリプトは、ソース エンドポイントまたはターゲット エンドポイントで使用できます。スクリプトはすべて SQL*Plus コマンドライン ユーティリティで実行するように記述されています。ユーティリティの詳しい使用方法については、Oracle ドキュメントの「A SQL コマンドラインの使用」をご参照ください。

スクリプトを実行する前に、使用するユーザーアカウントに Oracle データベースへのアクセス許可があるか確認してください。表示される許可設定は、ユーザーが次のように作成されたことを前提としています。

```
CREATE USER script_user IDENTIFIED BY password;
```

オンプレミス データベースの場合、以下に示す *script_user* におけるように最小許可を設定します。

```
GRANT CREATE SESSION TO script_user;
GRANT SELECT on V$DATABASE to script_user;
GRANT SELECT on V$VERSION to script_user;
GRANT SELECT on GV$SGA to script_user;
GRANT SELECT on GV$INSTANCE to script_user;
GRANT SELECT on GV$DATAGUARD_CONFIG to script_user;
GRANT SELECT on GV$LOG to script_user;
GRANT SELECT on DBA_TABLESPACES to script_user;
GRANT SELECT on DBA_DATA_FILES to script_user;
GRANT SELECT on DBA_SEGMENTS to script_user;
GRANT SELECT on DBA_LOBS to script_user;
GRANT SELECT on V$ARCHIVED_LOG to script_user;
GRANT SELECT on DBA_TAB_MODIFICATIONS to script_user;
GRANT SELECT on DBA_TABLES to script_user;
GRANT SELECT on DBA_TAB_PARTITIONS to script_user;
GRANT SELECT on DBA_MVIEWS to script_user;
GRANT SELECT on DBA_OBJECTS to script_user;
GRANT SELECT on DBA_TAB_COLUMNS to script_user;
GRANT SELECT on DBA_LOG_GROUPS to script_user;
GRANT SELECT on DBA_LOG_GROUP_COLUMNS to script_user;
GRANT SELECT on V$ARCHIVE_DEST to script_user;
GRANT SELECT on DBA_SYS_PRIVS to script_user;
GRANT SELECT on DBA_TAB_PRIVS to script_user;
```

```
GRANT SELECT on DBA_TYPES to script_user;
GRANT SELECT on DBA_CONSTRAINTS to script_user;
GRANT SELECT on V$TRANSACTION to script_user;
GRANT SELECT on GV$ASM_DISK_STAT to script_user;
GRANT SELECT on GV$SESSION to script_user;
GRANT SELECT on GV$SQL to script_user;
GRANT SELECT on DBA_ENCRYPTED_COLUMNS to script_user;
GRANT SELECT on DBA_PDBS to script_user;

GRANT EXECUTE on dbms_utility to script_user;
```

Amazon RDS データベースでは、以下に示すように最小限のアクセス許可を設定します。

```
GRANT CREATE SESSION TO script_user;
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$DATABASE','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$VERSION','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('GV_$SGA','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('GV_$INSTANCE','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('GV_
$DATAGUARD_CONFIG','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('GV_$LOG','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TABLESPACES','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_DATA_FILES','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_SEGMENTS','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_LOBS','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$ARCHIVED_LOG','script_user','SELECT');
exec
 rdsadmin.rdsadmin_util.grant_sys_object('DBA_TAB_MODIFICATIONS','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TABLES','script_user','SELECT');
exec
 rdsadmin.rdsadmin_util.grant_sys_object('DBA_TAB_PARTITIONS','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_MVIEWS','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_OBJECTS','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TAB_COLUMNS','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_LOG_GROUPS','script_user','SELECT');
exec
 rdsadmin.rdsadmin_util.grant_sys_object('DBA_LOG_GROUP_COLUMNS','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$ARCHIVE_DEST','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_SYS_PRIVS','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TAB_PRIVS','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TYPES','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_CONSTRAINTS','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$TRANSACTION','script_user','SELECT');
```

```
exec rdsadmin.rdsadmin_util.grant_sys_object('GV_
$ASM_DISK_STAT','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('GV_$SESSION','script_user','SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('GV_$SQL','script_user','SELECT');
exec
 rdsadmin.rdsadmin_util.grant_sys_object('DBA_ENCRYPTED_COLUMNS','script_user','SELECT');

exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_PDBS','script_user','SELECT');

exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_UTILITY','script_user','EXECUTE');
```

Oracle で使用可能な各 SQL*Plus サポート スクリプトのダウンロード、確認、および実行方法について説明します。また、AWS Support ケースでの出力を確認してアップロードする方法も確認できます。

トピック

- awsdms_support_collector_oracle.sql スクリプト

awsdms_support_collector_oracle.sql スクリプト

awsdms_support_collector_oracle.sql スクリプトをダウンロードします。

このスクリプトは、Oracle データベース構成に関する情報を収集します。スクリプトのチェックサムを必ず検証し、チェックサムが検証する場合は、スクリプト内の SQL コードを確認して、実行しにくいコードをコメントアウトします。スクリプトの整合性と内容に納得できたら、スクリプトを実行できます。

スクリプトを実行して結果をサポートケースにアップロードするには

1. 次の SQL*Plus コマンドラインを使用して、データベース環境からスクリプトを実行します。

   ```
   SQL> @awsdms_support_collector_oracle.sql
   ```

   <result>

   スクリプトには、簡単な説明と、実行を続行または中止するプロンプトが表示されます。[Enter] を押して確認します。
   </result>

2. 次のプロンプトで、移行するスキーマの名前を 1 つだけ入力します。

3. 次のプロンプトで、データベースに接続するように定義したユーザーの名前 (*script_user*) を入力します。
4. 次のプロンプトで、検査するデータの日数を入力するか、デフォルトをそのまま使用します。スクリプトは、指定されたデータをデータベースから収集します。
   <result>

   スクリプトが完了すると、例えば dms_support_oracle-2020-06-22-13-20-39-ORCL.html などの出力 HTML ファイルの名前が表示されます。スクリプトは、このファイルを作業ディレクトリに保存します。
   </result>
5. この HTML ファイルを確認し、共有が不快な情報をすべて削除します。HTML の共有に納得できたら、ファイルを AWS Support ケースにアップロードします。ファイルのアップロードの詳細については、「AWS DMS での診断サポート スクリプトの操作」をご参照ください。

SQL Server 診断 Support スクリプト

以下は、オンプレミスまたは AWS DMS マイグレーション設定において Amazon RDS for SQL Server データベースの分析に使用できる診断サポート スクリプトの説明です。これらのスクリプトは、ソース エンドポイントまたはターゲット エンドポイントで使用できます。オンプレミス データベースの場合は、sqlcmd コマンドライン ユーティリティでこれらのスクリプトを実行します。ユーティリティの使用方法の詳細については、Microsoft ドキュメントの「sqlcmd-ユーティリティを使用する」をご参照ください。

Amazon RDS データベースの場合、sqlcmd コマンドライン ユーティリティを使用して接続することはできません。代わりに、Amazon RDS SQL Server に接続する任意のクライアント ツールを使用して、これらのスクリプトを実行します。

スクリプトを実行する前に、使用するユーザーアカウントに SQL Server データベースへのアクセス許可があるか確認してください。オンプレミス データベースと Amazon RDS データベースの両方で、SQL Server データベースへの SysAdmin ロールなしのアクセスに使用するのと同じアクセス許可を使用できます。

トピック

- オンプレミスの SQL Server データベースに対する最小限のアクセス許可の設定
- Amazon RDS SQL Serverデータベースに対する最小限のアクセス許可の設定
- スタンドアロン SQL Server での継続的なレプリケーションのセットアップ: sysadmin ロールなし

- 可用性グループ環境の SQL Server での継続的なレプリケーションのセットアップ: sysadmin ロールなし
- SQL Server サポートスクリプト

オンプレミスの SQL Server データベースに対する最小限のアクセス許可の設定

オンプレミス SQL Server データベースに対して実行する最小限のアクセス許可を設定するには

1. SQL Server Management Studio (SSMS) を使用してパスワード認証する *on-prem-user* など新しい SQL Server アカウントを作成します。
2. SSMS の[User Mappings] (ユーザーマッピング)セクションで、MSDB と MASTER データベース (公開許可を付与します) を選択し、すくリプを実行するデータベースに DB_OWNER ロールを割り当てます。
3. 新しいアカウントのコンテキストメニュー (右クリック) を開き、[Security] (セキュリティ) を選択して Connect SQL 権限を明示的に付与します。
4. 以下の付与コマンドを実行します。

```
GRANT VIEW SERVER STATE TO on-prem-user;
USE MSDB;
GRANT SELECT ON MSDB.DBO.BACKUPSET TO on-prem-user;
GRANT SELECT ON MSDB.DBO.BACKUPMEDIAFAMILY TO on-prem-user;
GRANT SELECT ON MSDB.DBO.BACKUPFILE TO on-prem-user;
```

Amazon RDS SQL Serverデータベースに対する最小限のアクセス許可の設定

Amazon RDS SQL Server データベースの最小許可で実行するには

1. SQL Server Management Studio (SSMS) を使用してパスワード認証する *rds-user* など新しい SQL Server アカウントを作成します。
2. SSMS の [ユーザーマッピング] セクションで、[MSDB] データベース (パブリックアクセス権限を付与) を選択して、スクリプトを実行するデータベースに DB_OWNER ロールを割り当てます。
3. 新しいアカウントのコンテキストメニュー (右クリック) を開き、[Security] (セキュリティ) を選択して Connect SQL 権限を明示的に付与します。
4. 以下の付与コマンドを実行します。

```
GRANT VIEW SERVER STATE TO rds-user;
```

```
USE MSDB;
GRANT SELECT ON MSDB.DBO.BACKUPSET TO rds-user;
GRANT SELECT ON MSDB.DBO.BACKUPMEDIAFAMILY TO rds-user;
GRANT SELECT ON MSDB.DBO.BACKUPFILE TO rds-user;
```

スタンドアロン SQL Server での継続的なレプリケーションのセットアップ: sysadmin ロールなし

このセクションでは、ユーザーアカウントで sysadmin アクセス権限を必要としないスタンドアロン SQL Server データベースソースの継続的なレプリケーションをセットアップする方法について説明します。

> **Note**
>
> このセクションのステップを実行すると、システム管理者以外の DMS ユーザーには、以下を実行するアクセス許可が付与されます。
>
> - オンライントランザクションログファイルからの変更の読み取り
> - トランザクションログのバックアップファイルから変更を読み取るためのディスクアクセス
> - DMS が使用するパブリケーションの追加または変更
> - パブリケーションへの記事の追加

1. オンプレミスまたは Amazon EC2 上のセルフマネージド型 SQL Server のデータ変更のキャプチャ の説明に従って、レプリケーション向けに Microsoft SQL Server をセットアップします。
2. ソースデータベースで MS-REPLICATION を有効にします。これは手動で実行することも、sysadmin ユーザーとしてタスクを 1 回実行することによっても行うことができます。
3. 次のスクリプトを使用して、ソースデータベースに `awsdms` スキーマを作成します。

```
use master
go
create schema awsdms
go


-- Create the table valued function [awsdms].[split_partition_list] on the Master
 database, as follows:
```

```
USE [master]
GO

set ansi_nulls on
go

set quoted_identifier on
go

if (object_id('[awsdms].[split_partition_list]','TF')) is not null

drop function [awsdms].[split_partition_list];

go

create function [awsdms].[split_partition_list]

(

@plist varchar(8000), –A delimited list of partitions

@dlm nvarchar(1) –Delimiting character

)

returns @partitionsTable table –Table holding the BIGINT values of the string
 fragments

(

pid bigint primary key

)

as

begin

declare @partition_id bigint;

declare @dlm_pos integer;

declare @dlm_len integer;
```

```
set @dlm_len = len(@dlm);

while (charindex(@dlm,@plist)>0)

begin

set @dlm_pos = charindex(@dlm,@plist);

set @partition_id = cast( ltrim(rtrim(substring(@plist,1,@dlm_pos-1))) as bigint);

insert into @partitionsTable (pid) values (@partition_id)

set @plist = substring(@plist,@dlm_pos+@dlm_len,len(@plist));

end

set @partition_id = cast (ltrim(rtrim(@plist)) as bigint);

insert into @partitionsTable (pid) values ( @partition_id );

return

end

GO
```

4. 次のスクリプトを使用して Master データベースに [awsdms].[rtm_dump_dblog] プロシージャを作成します。

```
use [MASTER]

go

if (object_id('[awsdms].[rtm_dump_dblog]','P')) is not null drop procedure
 [awsdms].[rtm_dump_dblog];
go


set ansi_nulls on
go

set quoted_identifier on
GO
```

```
CREATE procedure [awsdms].[rtm_dump_dblog]
(
@start_lsn varchar(32),
@seqno integer,
@filename varchar(260),
@partition_list varchar(8000), — A comma delimited list: P1,P2,... Pn
@programmed_filtering integer,
@minPartition bigint,
@maxPartition bigint
)
as begin
declare @start_lsn_cmp varchar(32); — Stands against the GT comparator
SET NOCOUNT ON — - Disable "rows affected display"
set @start_lsn_cmp = @start_lsn;
if (@start_lsn_cmp) is null
set @start_lsn_cmp = '00000000:00000000:0000';
if (@partition_list is null)
begin
RAISERROR ('Null partition list waspassed',16,1);
return
end
```

```
if (@start_lsn) is not null

set @start_lsn = '0x'+@start_lsn;

if (@programmed_filtering=0)



SELECT

[Current LSN],

[operation],

[Context],

[Transaction ID],

[Transaction Name],

[Begin Time],

[End Time],

[Flag Bits],

[PartitionID],

[Page ID],

[Slot ID],

[RowLog Contents 0],

[Log Record],

[RowLog Contents 1]

FROM

fn_dump_dblog (

@start_lsn, NULL, N'DISK', @seqno, @filename,
```

```
default, default, default, default, default, default, default,

default, default, default, default, default, default, default,

default, default, default, default, default, default, default,

default, default, default, default, default, default, default,

default, default, default, default, default, default, default,

default, default, default, default, default, default, default,

default, default, default, default, default, default, default,

default, default, default, default, default, default, default,

default, default, default, default, default, default, default)

where [Current LSN] collate SQL_Latin1_General_CP1_CI_AS > @start_lsn_cmp collate
 SQL_Latin1_General_CP1_CI_AS

and

(

( [operation] in ('LOP_BEGIN_XACT','LOP_COMMIT_XACT','LOP_ABORT_XACT') )

or

( [operation] in ('LOP_INSERT_ROWS','LOP_DELETE_ROWS','LOP_MODIFY_ROW')

and

( ( [context] in ('LCX_HEAP','LCX_CLUSTERED','LCX_MARK_AS_GHOST') ) or ([context] =
 'LCX_TEXT_MIX' and (datalength([RowLog Contents 0]) in (0,1))))

and [PartitionID] in ( select * from master.awsdms.split_partition_list
 (@partition_list,','))

)

or
```

```
([operation] = 'LOP_HOBT_DDL')

)


else


SELECT

[Current LSN],

[operation],

[Context],

[Transaction ID],

[Transaction Name],

[Begin Time],

[End Time],

[Flag Bits],

[PartitionID],

[Page ID],

[Slot ID],

[RowLog Contents 0],

[Log Record],

[RowLog Contents 1] — After Image

FROM

fn_dump_dblog (

@start_lsn, NULL, N'DISK', @seqno, @filename,
```

```
default, default, default, default, default, default, default,

default, default, default, default, default, default, default,

default, default, default, default, default, default, default,

default, default, default, default, default, default, default,

default, default, default, default, default, default, default,

default, default, default, default, default, default, default,

default, default, default, default, default, default, default,

default, default, default, default, default, default, default)

where [Current LSN] collate SQL_Latin1_General_CP1_CI_AS > @start_lsn_cmp collate
 SQL_Latin1_General_CP1_CI_AS

and

(

( [operation] in ('LOP_BEGIN_XACT','LOP_COMMIT_XACT','LOP_ABORT_XACT') )

or

( [operation] in ('LOP_INSERT_ROWS','LOP_DELETE_ROWS','LOP_MODIFY_ROW')

and

( ( [context] in ('LCX_HEAP','LCX_CLUSTERED','LCX_MARK_AS_GHOST') ) or ([context] =
 'LCX_TEXT_MIX' and (datalength([RowLog Contents 0]) in (0,1))))

and ([PartitionID] is not null) and ([PartitionID] >= @minPartition and
 [PartitionID]<=@maxPartition)

)

or

([operation] = 'LOP_HOBT_DDL')
```

```
)



SET NOCOUNT OFF – Re-enable "rows affected display"

end

GO
```

5. 次のスクリプトを使用して、Master データベースに証明書を作成します。

```
Use [master]
Go

CREATE CERTIFICATE [awsdms_rtm_dump_dblog_cert] ENCRYPTION BY PASSWORD =
 N'@5trongpassword'

WITH SUBJECT = N'Certificate for FN_DUMP_DBLOG Permissions';
```

6. 次のスクリプトを使用して、証明書からログインを作成します。

```
Use [master]
Go

CREATE LOGIN awsdms_rtm_dump_dblog_login FROM CERTIFICATE
 [awsdms_rtm_dump_dblog_cert];
```

7. 次のスクリプトを使用して、ログインを sysadmin サーバーロールに追加します。

```
ALTER SERVER ROLE [sysadmin] ADD MEMBER [awsdms_rtm_dump_dblog_login];
```

8. 次のスクリプトを使用して、証明書を使って署名を [master].[awsdms].[rtm_dump_dblog] に追加します。

```
Use [master]
GO
ADD SIGNATURE
TO [master].[awsdms].[rtm_dump_dblog] BY CERTIFICATE [awsdms_rtm_dump_dblog_cert]
 WITH PASSWORD = '@5trongpassword';
```

> **Note**
>
> ストアドプロシージャを再作成する場合は、署名を再度追加する必要があります。

9. 次のスクリプトを使用して、マスターデータベースに [awsdms].[rtm_position_1st_timestamp] を作成します。

```
use [master]
    if object_id('[awsdms].[rtm_position_1st_timestamp]','P') is not null
    DROP PROCEDURE [awsdms].[rtm_position_1st_timestamp];
    go
    create procedure [awsdms].[rtm_position_1st_timestamp]
    (
    @dbname                sysname,      -- Database name
    @seqno                 integer,      -- Backup set sequence/position number
 within file
    @filename              varchar(260), -- The backup filename
    @1stTimeStamp          varchar(40)   -- The timestamp to position by
    )
    as begin

    SET NOCOUNT ON       -- Disable "rows affected display"

    declare @firstMatching table
    (
    cLsn varchar(32),
    bTim datetime
    )

    declare @sql nvarchar(4000)
    declare @nl                       char(2)
    declare @tb                       char(2)
    declare @fnameVar                 nvarchar(254) = 'NULL'

    set @nl  = char(10); -- New line
    set @tb  = char(9)   -- Tab separator

    if (@filename is not null)
    set @fnameVar = ''''+@filename +''''

    set @sql='use ['+@dbname+'];'+@nl+
    'select top 1 [Current LSN],[Begin Time]'+@nl+
```

```
    'FROM fn_dump_dblog (NULL, NULL, NULL, '+ cast(@seqno as varchar(10))+','+
 @fnameVar+','+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default)'+@nl+
    'where operation=''LOP_BEGIN_XACT''' +@nl+
    'and [Begin Time]>= cast('+''''+@1stTimeStamp+''''+' as datetime)'+@nl

    --print @sql
    delete from  @firstMatching
    insert into @firstMatching  exec sp_executesql @sql    -- Get them all

    select top 1 cLsn as [matching LSN],convert(varchar,bTim,121) as [matching
 Timestamp] from @firstMatching;

    SET NOCOUNT OFF      -- Re-enable "rows affected display"

    end
    GO
```

10. 次のスクリプトを使用して、Master データベースに証明書を作成します。

```
Use [master]
Go
CREATE CERTIFICATE [awsdms_rtm_position_1st_timestamp_cert]
ENCRYPTION BY PASSWORD = '@5trongpassword'
WITH SUBJECT = N'Certificate for FN_POSITION_1st_TIMESTAMP Permissions';
```

11. 次のスクリプトを使用して、証明書からログインを作成します。

```
Use [master]
Go
CREATE LOGIN awsdms_rtm_position_1st_timestamp_login FROM CERTIFICATE
 [awsdms_rtm_position_1st_timestamp_cert];
```

12. 次のスクリプトを使用して、sysadmin ロールにログインを追加します。

```
ALTER SERVER ROLE [sysadmin] ADD MEMBER [awsdms_rtm_position_1st_timestamp_login];
```

13. 次のスクリプトに従い、証明書を使用して [master].[awsdms].[rtm_position_1st_timestamp] に署名を追加します。

```
Use [master]
    GO
    ADD SIGNATURE
    TO [master].[awsdms].[rtm_position_1st_timestamp]
    BY CERTIFICATE [awsdms_rtm_position_1st_timestamp_cert]
    WITH PASSWORD = '@5trongpassword';
```

14. 次のスクリプトを使用して、新しいストアドプロシージャへの実行アクセス権を DMS ユーザーに付与します。

```
use master
go
GRANT execute on [awsdms].[rtm_position_1st_timestamp] to dms_user;
```

15. 次の各データベースに、次の権限とロールを持つユーザーを作成します。

> **Note**
>
> dmsnosysadmin ユーザーアカウントは、各レプリカで同じ SID を使用して作成する必要があります。次の SQL クエリは、各レプリカの dmsnosysadmin アカウントの SID 値を確認するのに役立ちます。ユーザー作成の詳細については、「Microsoft SQL Server ドキュメント」の「 CREATE USER (Transact-SQL)」を参照してください。Azure SQL Database の SQL ユーザーアカウントの作成の詳細については、「アクティブな地理的レプリケーション」を参照してください。

```
use master
go
grant select on sys.fn_dblog to [DMS_user]
grant view any definition to [DMS_user]
grant view server state to [DMS_user]–(should be granted to the login).
grant execute on sp_repldone to [DMS_user]
grant execute on sp_replincrementlsn to [DMS_user]
grant execute on sp_addpublication to [DMS_user]
```

```
grant execute on sp_addarticle to [DMS_user]
grant execute on sp_articlefilter to [DMS_user]
grant select on [awsdms].[split_partition_list] to [DMS_user]
grant execute on [awsdms].[rtm_dump_dblog] to [DMS_user]
```

```
use MSDB
go
grant select on msdb.dbo.backupset to [DMS_user]
grant select on msdb.dbo.backupmediafamily to [DMS_user]
grant select on msdb.dbo.backupfile to [DMS_user]
```

ソースデータベースで次のスクリプトを実行します。

```
EXEC sp_addrolemember N'db_owner', N'DMS_user'
use Source_DB
go
```

16. 最後に、追加の接続属性 (ECA) をソースの SQL Server エンドポイントに追加します。

```
enableNonSysadminWrapper=true;
```

可用性グループ環境の SQL Server での継続的なレプリケーションのセットアップ: sysadmin ロールなし

このセクションでは、ユーザーアカウントに sysadmin 権限を必要としない可用性グループ環境で、SQL Server データベースソースの継続的なレプリケーションをセットアップする方法について説明します。

> **Note**
>
> このセクションのステップを実行すると、システム管理者以外の DMS ユーザーには、以下を実行するアクセス許可が付与されます。
>
> - オンライントランザクションログファイルからの変更の読み取り
> - トランザクションログのバックアップファイルから変更を読み取るためのディスクアクセス
> - DMS が使用するパブリケーションの追加または変更

- パブリケーションへの記事の追加

可用性グループ環境で sysadmin ユーザーを使用せずに継続的なレプリケーションをセットアップするには

1. オンプレミスまたは Amazon EC2 上のセルフマネージド型 SQL Server のデータ変更のキャプチャ の説明に従って、レプリケーション向けに Microsoft SQL Server をセットアップします。
2. ソースデータベースで MS-REPLICATION を有効にします。これは手動で実行することも、sysadmin ユーザーを使用してタスクを 1 回実行することによっても行うことができます。

   > **Note**
   >
   > MS-REPLICATION ディストリビューターをローカルとして設定するか、関連するリンクサーバーを経由して sysadmin 以外のユーザーがアクセスできるように設定する必要があります。

3. [Exclusively use sp_repldone within a single task] エンドポイントオプションが有効になっている場合は、MS-REPLICATION Log Reader ジョブは停止します。
4. 各レプリカで次のステップを実行します。

   1. master データベースに [awsdms] [awsdms] スキーマを作成します。

      ```
      CREATE SCHEMA [awsdms]
      ```

   2. Masterデータベースに [awsdms].[split_partition_list] テーブル値関数を作成します。

      ```
      USE [master]
      GO

      SET ansi_nulls on
      GO

      SET quoted_identifier on
      GO

      IF (object_id('[awsdms].[split_partition_list]','TF')) is not null
        DROP FUNCTION [awsdms].[split_partition_list];
      GO
      ```

```
CREATE FUNCTION [awsdms].[split_partition_list]
(
  @plist varchar(8000),    --A delimited list of partitions
  @dlm nvarchar(1)    --Delimiting character
)
RETURNS @partitionsTable table --Table holding the BIGINT values of the string
 fragments
(
  pid bigint primary key
)
AS
BEGIN
  DECLARE @partition_id bigint;
  DECLARE @dlm_pos integer;
  DECLARE @dlm_len integer;
  SET @dlm_len = len(@dlm);
  WHILE (charindex(@dlm,@plist)>0)
  BEGIN
    SET @dlm_pos = charindex(@dlm,@plist);
    SET @partition_id = cast( ltrim(rtrim(substring(@plist,1,@dlm_pos-1))) as
 bigint);
    INSERT into @partitionsTable (pid) values (@partition_id)
    SET @plist = substring(@plist,@dlm_pos+@dlm_len,len(@plist));
  END
  SET @partition_id = cast (ltrim(rtrim(@plist)) as bigint);
  INSERT into @partitionsTable (pid) values (  @partition_id  );
  RETURN
END
GO
```

3. Masterデータベースに [awsdms].[rtm_dump_dblog] プロシージャを作成します。

```
USE [MASTER]
GO

IF (object_id('[awsdms].[rtm_dump_dblog]','P')) is not null
  DROP PROCEDURE [awsdms].[rtm_dump_dblog];
GO

SET ansi_nulls on
GO

SET quoted_identifier on
```

```
GO

CREATE PROCEDURE [awsdms].[rtm_dump_dblog]
(
  @start_lsn            varchar(32),
  @seqno                integer,
  @filename             varchar(260),
  @partition_list       varchar(8000), -- A comma delimited list: P1,P2,... Pn
  @programmed_filtering integer,
  @minPartition         bigint,
  @maxPartition         bigint
)
AS
BEGIN

  DECLARE @start_lsn_cmp varchar(32); -- Stands against the GT comparator

  SET NOCOUNT ON  -- Disable "rows affected display"

  SET @start_lsn_cmp = @start_lsn;
  IF (@start_lsn_cmp) is null
    SET @start_lsn_cmp = '00000000:00000000:0000';

  IF (@partition_list is null)
    BEGIN
      RAISERROR ('Null partition list was passed',16,1);
      return
      --set @partition_list = '0,';    -- A dummy which is never matched
    END

  IF (@start_lsn) is not null
    SET @start_lsn = '0x'+@start_lsn;

  IF (@programmed_filtering=0)
    SELECT
      [Current LSN],
      [operation],
      [Context],
      [Transaction ID],
      [Transaction Name],
      [Begin Time],
      [End Time],
      [Flag Bits],
      [PartitionID],
```

```
      [Page ID],
      [Slot ID],
      [RowLog Contents 0],
      [Log Record],
      [RowLog Contents 1] -- After Image
    FROM
      fn_dump_dblog (
        @start_lsn, NULL, N'DISK', @seqno, @filename,
        default, default, default, default, default, default, default,
        default, default, default, default, default, default, default,
        default, default, default, default, default, default, default,
        default, default, default, default, default, default, default,
        default, default, default, default, default, default, default,
        default, default, default, default, default, default, default,
        default, default, default, default, default, default, default,
        default, default, default, default, default, default, default,
        default, default, default, default, default, default, default)
    WHERE
      [Current LSN] collate SQL_Latin1_General_CP1_CI_AS > @start_lsn_cmp collate
SQL_Latin1_General_CP1_CI_AS -- This aims for implementing FN_DBLOG based on GT
comparator.
      AND
      (
        (  [operation] in ('LOP_BEGIN_XACT','LOP_COMMIT_XACT','LOP_ABORT_XACT') )
        OR
        (  [operation] in ('LOP_INSERT_ROWS','LOP_DELETE_ROWS','LOP_MODIFY_ROW')
          AND
          ( ( [context]   in ('LCX_HEAP','LCX_CLUSTERED','LCX_MARK_AS_GHOST') )
or ([context] = 'LCX_TEXT_MIX') )
          AND
          [PartitionID] in ( select * from master.awsdms.split_partition_list
(@partition_list,','))
        )
      OR
      ([operation] = 'LOP_HOBT_DDL')
    )
    ELSE
      SELECT
        [Current LSN],
        [operation],
        [Context],
        [Transaction ID],
        [Transaction Name],
        [Begin Time],
```

```
        [End Time],
        [Flag Bits],
        [PartitionID],
        [Page ID],
        [Slot ID],
        [RowLog Contents 0],
        [Log Record],
        [RowLog Contents 1] -- After Image
      FROM
        fn_dump_dblog (
          @start_lsn, NULL, N'DISK', @seqno, @filename,
          default, default, default, default, default, default, default,
          default, default, default, default, default, default, default,
          default, default, default, default, default, default, default,
          default, default, default, default, default, default, default,
          default, default, default, default, default, default, default,
          default, default, default, default, default, default, default,
          default, default, default, default, default, default, default,
          default, default, default, default, default, default, default)
      WHERE [Current LSN] collate SQL_Latin1_General_CP1_CI_AS > @start_lsn_cmp
 collate SQL_Latin1_General_CP1_CI_AS -- This aims for implementing FN_DBLOG
 based on GT comparator.
      AND
      (
        (  [operation] in ('LOP_BEGIN_XACT','LOP_COMMIT_XACT','LOP_ABORT_XACT') )
        OR
        (  [operation] in ('LOP_INSERT_ROWS','LOP_DELETE_ROWS','LOP_MODIFY_ROW')
          AND
          ( ( [context]   in ('LCX_HEAP','LCX_CLUSTERED','LCX_MARK_AS_GHOST') )
 or ([context] = 'LCX_TEXT_MIX') )
          AND ([PartitionID] is not null) and ([PartitionID] >= @minPartition and
 [PartitionID]<=@maxPartition)
        )
        OR
        ([operation] = 'LOP_HOBT_DDL')
      )
      SET NOCOUNT OFF -- Re-enable "rows affected display"
END
GO
```

4. Master データベースに次のとおり証明書を作成します。

```
USE [master]
```

```
GO
CREATE CERTIFICATE [awsdms_rtm_dump_dblog_cert]
  ENCRYPTION BY PASSWORD = N'@hardpassword1'
  WITH SUBJECT = N'Certificate for FN_DUMP_DBLOG Permissions'
```

5. CSR から次のとおり証明書を作成します。

```
USE [master]
GO
CREATE LOGIN awsdms_rtm_dump_dblog_login FROM CERTIFICATE
  [awsdms_rtm_dump_dblog_cert];
```

6. sysadmin サーバーロールにログイン情報を追加します。

```
ALTER SERVER ROLE [sysadmin] ADD MEMBER [awsdms_rtm_dump_dblog_login];
```

7. この証明書を使用して署名を [master].[awsdms].[rtm_dump_dblog] プロシージャに追加します。

```
USE [master]
GO

ADD SIGNATURE
  TO [master].[awsdms].[rtm_dump_dblog]
  BY CERTIFICATE [awsdms_rtm_dump_dblog_cert]
  WITH PASSWORD = '@hardpassword1';
```

> **Note**
>
> ストアドプロシージャを再作成する場合は、署名を再度追加する必要があります。

8. Masterデータベースに `[awsdms].[rtm_position_1st_timestamp]` プロシージャを作成します。

```
USE [master]
IF object_id('[awsdms].[rtm_position_1st_timestamp]','P') is not null
  DROP PROCEDURE [awsdms].[rtm_position_1st_timestamp];
GO
CREATE PROCEDURE [awsdms].[rtm_position_1st_timestamp]
(
  @dbname                sysname,      -- Database name
```

```
  @seqno                  integer,      -- Backup set sequence/position number
 within file
  @filename               varchar(260), -- The backup filename
  @1stTimeStamp           varchar(40)   -- The timestamp to position by
)
AS
BEGIN
  SET NOCOUNT ON       -- Disable "rows affected display"

  DECLARE @firstMatching table
  (
    cLsn varchar(32),
    bTim datetime
  )
  DECLARE @sql nvarchar(4000)
  DECLARE @nl                         char(2)
  DECLARE @tb                         char(2)
  DECLARE @fnameVar                   sysname = 'NULL'

  SET @nl  = char(10); -- New line
  SET @tb  = char(9)   -- Tab separator

  IF (@filename is not null)
    SET @fnameVar = ''''+@filename +''''
  SET @filename = ''''+@filename +''''
  SET @sql='use ['+@dbname+'];'+@nl+
    'SELECT TOP 1 [Current LSN],[Begin Time]'+@nl+
    'FROM fn_dump_dblog (NULL, NULL, NULL, '+ cast(@seqno as varchar(10))+','+
 @filename +','+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default,'+@nl+
    @tb+'default, default, default, default, default, default, default)'+@nl+
    'WHERE operation=''LOP_BEGIN_XACT''' +@nl+
    'AND [Begin Time]>= cast('+''''+@1stTimeStamp+''''+' as datetime)'+@nl

    --print @sql
    DELETE FROM @firstMatching
    INSERT INTO @firstMatching  exec sp_executesql @sql    -- Get them all
```

```
    SELECT TOP 1 cLsn as [matching LSN],convert(varchar,bTim,121) AS[matching
 Timestamp] FROM @firstMatching;

    SET NOCOUNT OFF      -- Re-enable "rows affected display"

END
GO
```

9. Master データベースに次のとおり証明書を作成します。

```
USE [master]
GO
CREATE CERTIFICATE [awsdms_rtm_position_1st_timestamp_cert]
  ENCRYPTION BY PASSWORD = N'@hardpassword1'
  WITH SUBJECT = N'Certificate for FN_POSITION_1st_TIMESTAMP Permissions';
```

10. CSR から次のとおり証明書を作成します。

```
USE [master]
GO
CREATE LOGIN awsdms_rtm_position_1st_timestamp_login FROM CERTIFICATE
  [awsdms_rtm_position_1st_timestamp_cert];
```

11. sysadmin サーバーロールにログイン情報を追加します。

```
ALTER SERVER ROLE [sysadmin] ADD MEMBER
 [awsdms_rtm_position_1st_timestamp_login];
```

12. この証明書を使用して、署名を [master].[awsdms].[rtm_position_1st_timestamp] プロシージャに追加します。

```
USE [master]
GO
ADD SIGNATURE
  TO [master].[awsdms].[rtm_position_1st_timestamp]
  BY CERTIFICATE [awsdms_rtm_position_1st_timestamp_cert]
  WITH PASSWORD = '@hardpassword1';
```

> **Note**
>
> ストアドプロシージャを再作成する場合は、署名を再度追加する必要があります。

13次の各データベースに次のアクセス権限またはロールを持つユーザーを作成します。

> **Note**
>
> dmsnosysadmin ユーザーアカウントは、各レプリカで同じ SID を使用して作成する必要があります。次の SQL クエリは、各レプリカの dmsnosysadmin アカウントの SID 値を確認するのに役立ちます。ユーザー作成の詳細については、「Microsoft SQL Server ドキュメント」の「CREATE USER (Transact-SQL)」を参照してください。Azure SQL Database の SQL ユーザーアカウントの作成の詳細については、「アクティブな地理的レプリケーション」を参照してください。

```
SELECT @@servername servername, name, sid, create_date, modify_date
  FROM sys.server_principals
  WHERE name = 'dmsnosysadmin';
```

14.各レプリカの master データベースに対するアクセス権限を付与します。

```
USE master
GO

GRANT select on sys.fn_dblog to dmsnosysadmin;
GRANT view any definition to dmsnosysadmin;
GRANT view server state to dmsnosysadmin -- (should be granted to the login).
GRANT execute on sp_repldone to dmsnosysadmin;
GRANT execute on sp_replincrementlsn to dmsnosysadmin;
GRANT execute on sp_addpublication to dmsnosysadmin;
GRANT execute on sp_addarticle to dmsnosysadmin;
GRANT execute on sp_articlefilter to dmsnosysadmin;
GRANT select on [awsdms].[split_partition_list] to dmsnosysadmin;
GRANT execute on [awsdms].[rtm_dump_dblog] to dmsnosysadmin;
GRANT execute on [awsdms].[rtm_position_1st_timestamp] to dmsnosysadmin;
```

15.各レプリカの msdb データベースに対するアクセス権限を付与します。

```
USE msdb
GO
GRANT select on msdb.dbo.backupset to dmsnosysadmin
GRANT select on msdb.dbo.backupmediafamily to dmsnosysadmin
GRANT select on msdb.dbo.backupfile to dmsnosysadmin
```

16.db_owner ロールをソースデータベースの dmsnosysadmin に追加します。データベースは同期しているため、プライマリレプリカにのみロールを追加します。

```
use <source DB>
GO
EXEC sp_addrolemember N'db_owner', N'dmsnosysadmin'
```

SQL Server サポートスクリプト

次のトピックでは、SQL Server で使用可能な各サポート スクリプトをダウンロード、確認、実行する方法について説明します。また、スクリプト出力を確認して AWS Support ケースにアップロードする方法についても説明します。

トピック

- awsdms_support_collector_sql_server.sql スクリプト

awsdms_support_collector_sql_server.sql スクリプト

awsdms_support_collector_sql_server.sql スクリプトをダウンロードします。

> **Note**
>
> この SQL Server 診断サポートスクリプトは、SQL Server 2014 以降のバージョンでのみ実行します。

このスクリプトは、SQL Server データベース構成に関する情報を収集します。スクリプトのチェックサムを必ず検証し、チェックサムが検証する場合は、スクリプト内の SQL コードを確認して、実行しにくいコードをコメントアウトします。スクリプトの整合性と内容に納得できたら、スクリプトを実行できます。

オンプレミス SQL Server データベースのスクリプトを実行するには

1. 次の sqlcmd コマンドを使用して、スクリプトを実行します。

```
sqlcmd -Uon-prem-user -Ppassword -SDMS-SQL17AG-N1 -y 0
-iC:\Users\admin\awsdms_support_collector_sql_server.sql -oC:\Users\admin
\DMS_Support_Report_SQLServer.html -dsqlserverdb01
```

指定された sqlcmd コマンドパラメータには、以下が含まれます。

- -U – データベースユーザー名。
- -P – データベース ユーザーパスワード。
- -S – SQL Server データベースサーバーの名前。
- -y – sqlcmd ユーティリティから出力される列の最大幅。値 0 は、無制限の幅を持つ列を指定します。
- -i – 実行するサポート スクリプトのパス (この場合は `awsdms_support_collector_sql_server.sql`)。
- -o – 収集されたデータベース構成情報を含む、指定したファイル名の出力 HTML ファイルのパス。
- -d – SQL Server データベース名。

2. スクリプトが完了したら、出力 HTML ファイルを確認し、共有しにくい情報をすべて削除します。HTML の共有に納得できたら、ファイルを AWS Support ケースにアップロードします。ファイルのアップロードの詳細については、「AWS DMS での診断サポート スクリプトの操作」をご参照ください。

Amazon RDS for SQL Server では、sqlcmd コマンドライン ユーティリティを使用して接続できないので、次の手順を使用します。

RDS SQL Server データベースのスクリプトを実行するには

1. RDS SQL Server に `Master` ユーザーとして接続できる任意のクライアント ツールを使用して、スクリプトを実行し、出力を HTML ファイルとして保存します。
2. 出力 HTML ファイルを確認し、共有しにくい情報をすべて削除します。HTML の共有に納得できたら、ファイルを AWS Support ケースにアップロードします。ファイルのアップロードの詳細については、「AWS DMS での診断サポート スクリプトの操作」をご参照ください。

MySQL 互換データベースの診断サポート スクリプト

以下に、オンプレミスまたは Amazon RDS for MySQL 互換データベースを分析するために使用できる診断サポート スクリプトを AWS DMS マイグレーション設定について説明します。これらのスクリプトは、ソース エンドポイントまたはターゲット エンドポイントで使用できます。スクリプトはすべて MySQL SQL コマンドラインで実行するように記述されています。

MySQL クライアントのインストールの詳細については、MySQL ドキュメントの「MySQL Shell のインストール」をご参照ください。MySQL クライアントの使用方法の詳細については、MySQL ドキュメントの「MySQL シェルコマンドを使用する」をご参照ください。

スクリプトを実行する前に、使用するユーザーアカウントに MySQL 互換データベースへのアクセス許可があるか確認してください。次の手順でユーザーアカウントを作成し、このスクリプトを実行するために必要な最小限のアクセス許可を提供します。

これらのスクリプトを実行するための最小許可を受けtあユーザーアカウントを設定するには

1. スクリプトを実行するユーザーを作成します。

```
create user 'username'@'hostname' identified by password;
```

2. select コマンドでデータベース分析します。

```
grant select on database-name.* to username;
grant replication client on *.* to username;
```

3.

```
grant execute on procedure mysql.rds_show_configuration to username;
```

次のトピックでは、MySQL 互換データベースで使用可能な各サポート スクリプトをダウンロード、確認、実行する方法について説明します。また、スクリプト出力を確認して AWS Support ケースにアップロードする方法についても説明します。

トピック

- awsdms_support_collector_MySQL.sql スクリプト

awsdms_support_collector_MySQL.sql スクリプト

awsdms_support_collector_MySQL.sql スクリプトをダウンロードします。

このスクリプトは、MySQL 互換データベース設定に関する情報を収集します。スクリプトのチェックサムを必ず検証し、チェックサムが検証する場合は、スクリプト内の SQL コードを確認して、実行しにくいコードをコメントアウトします。スクリプトの整合性と内容に納得できたら、スクリプトを実行できます。

コマンドラインを使用してデータベース環境に接続した後、スクリプトを実行します。

このスクリプトを実行して結果を Support ケースにアップロードするには

1. mysql コマンドを使用してデータベースに接続します。

```
mysql -h hostname -P port -u username database-name
```

2. 次の mysql source コマンドを使用してスクリプトを実行します。

```
mysql> source awsdms_support_collector_MySQL_compatible_DB.sql
```

生成されたレポートを確認し、共有しにくい情報をすべて削除します。コンテンツが共有できるようになったら、ファイルをAWS Support ケースにアップロードします。ファイルのアップロードの詳細については、「AWS DMS での診断サポート スクリプトの操作」をご参照ください。

> **Note**
>
> - MySQL 互換データベースの診断サポート スクリプト で説明する必要な権限を持つユーザーアカウントをすでにお持ちの場合、既存のユーザーアカウントを使用してスクリプトを実行することもできます。
> - スクリプトを実行する前に、必ずデータベースに接続してください。
> - スクリプトは、テキスト形式で出力を生成します。
> - セキュリティのベストプラクティスを念頭に置いて、この MySQL 診断 Support スクリプトの実行専用に新しいユーザーアカウントを作成する場合は、スクリプトが正常に実行された後、このユーザーアカウントを削除することをお勧めします。

PostgreSQL 診断 Support スクリプト

以下に、AWS DMS マイグレーション設定で PostgreSQL RDBMS (オンプレミス、Amazon RDS、または Aurora PostgreSQL) を分析するために使用できる診断サポート スクリプトについて説明します。これらのスクリプトは、ソース エンドポイントまたはターゲット エンドポイントで使用できます。スクリプトはすべて psql コマンドライン ユーティリティで実行するように記述されています。

これらのスクリプトを実行する前に、使用するユーザーアカウントに PostgreSQL RDBMS へのアクセス許可があることを確認してください。

- PostgreSQL 10.x 以降 – pg_catalog.pg_ls_waldir 関数の実行アクセス権限を持つユーザーアカウント
- PostgreSQL 9.x 以前 — デフォルト許可を持つユーザーアカウント。

これらのスクリプトを実行するには、適切な許可を持つ既存のアカウントを使用することをお勧めします。

これらのスクリプトを実行するために新しいユーザーアカウントを作成するか、既存のアカウントに許可を付与する必要がある場合は、PostgreSQL バージョンに基づく任意の PostgreSQL RDBMS に対して次の SQL コマンドを実行できます。

PostgreSQL データベースバージョン 10.x 以降でこのスクリプトを実行するアクセス権限をアカウントに付与するには

- 次のいずれかを行います:
  - 新しいユーザーアカウントの場合は、以下を実行します。

```
CREATE USER script_user WITH PASSWORD 'password';
GRANT EXECUTE ON FUNCTION pg_catalog.pg_ls_waldir TO script_user;
```

  - 既存のユーザーアカウントの場合は、以下を実行します。

```
GRANT EXECUTE ON FUNCTION pg_catalog.pg_ls_waldir TO script_user;
```

PostgreSQL 9.x 以前のデータベースに対してこれらのスクリプトを実行する許可をアカウントに付与するには

- 次のいずれかを行います:
  - 新しいユーザーアカウントでは、デフォルトのアクセス許可を使用して以下を実行します。

```
CREATE USER script_user WITH PASSWORD password;
```

  - 既存のユーザーアカウントの場合は、既存の許可を使用します。

> **Note**
>
> これらのスクリプトは、PostgreSQL 9.x 以前のデータベースの WAL サイズの検索に関連する特定の機能をサポートしていません。詳細については、AWS Supportをご参照ください。

次のトピックでは、PostgreSQL で使用可能な各 Support スクリプトのダウンロードおよび確認、実行方法について説明します。また、スクリプト出力を確認して、AWS Support ケースにアップロードする方法についても説明します。

トピック

- awsdms_support_collector_postgres.sql スクリプト

awsdms_support_collector_postgres.sql スクリプト

awsdms_support_collector_postgres.sql スクリプトをダウンロードします。

このスクリプトは PostgreSQL データベースの設定に関する情報を収集します。スクリプトのチェックサムを必ず確認してください。チェックサムが検証する場合は、スクリプト内の SQL コードを確認して、実行しにくいコードをコメントアウトします。スクリプトの整合性と内容に納得できたら、スクリプトを実行できます。

> **Note**
>
> このスクリプトは、psql クライアントバージョン 10 以降で実行できます。

次の手順を使用して、データベース環境またはコマンドラインからこのスクリプトを実行できます。いずれの場合も、後でファイルを AWS Support にアップロードできます。

このスクリプトを実行して結果を Support ケースにアップロードするには

1. 次のいずれかを行います:

   - 次の psql コマンドラインを使用して、データベース環境からスクリプトを実行します。

     ```
     dbname=# \i awsdms_support_collector_postgres.sql
     ```

     次のプロンプトで、移行するスキーマの名前を 1 つだけ入力します。

次のプロンプトで、データベースに接続するように定義しているユーザー名 (*script_user*) を入力します。

- コマンドラインから直接、次のスクリプトを実行します。このオプションは、スクリプトの実行前のプロンプトを回避します。

```
psql -h database-hostname -p port -U script_user -d database-name -f
 awsdms_support_collector_postgres.sql
```

2. 出力 HTML ファイルを確認し、共有しにくい情報をすべて削除します。HTML の共有に納得できたら、ファイルを AWS Support ケースにアップロードします。ファイルのアップロードの詳細については、「AWS DMS での診断サポート スクリプトの操作」をご参照ください。

AWS DMS 診断サポート AMI の使用

の使用時にネットワーク関連の問題が発生した場合 AWS DMS、サポートエンジニアはネットワーク設定に関する詳細情報が必要になる場合があります。サポートが必要な情報をできるだけ短い時間で AWS 取得できるようにしたいと考えています。そのため、 AWS DMS ネットワーク環境をテストするための診断ツールを備えた構築済みの Amazon EC2 AMI を開発しました。

Amazon マシンイメージ (AMI) にインストールされている診断テストには次があります。

- 仮想プライベートクラウド (VPC)
- ネットワークのパケットロス
- ネットワークのレイテンシー
- 最大送信単位 (MTU) のサイズ

トピック

> **Note**
>
> Oracle ソースでパフォーマンスの問題が発生した場合は、Oracle REDO ログまたはアーカイブログの読み取りパフォーマンスを評価して、パフォーマンスを改善する方法を探ることができます。詳細については、「Evaluating read performance of Oracle redo or archive logs」を参照してください。

新しい AWS DMS 診断用 Amazon EC2 インスタンスを起動する

このセクションでは、新しい Amazon EC2 インスタンスを起動します。Amazon EC2 インスタンスの起動方法の詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 Linux インスタンスの開始方法」チュートリアルを参照してください。

次の設定で Amazon EC2 インスタンスを起動します。

- [Application and OS Images (Amazon Machine Image)] で、DMS-DIAG-AMI AMI を検索します。コンソールにログオンしている場合は、このクエリで AMI を検索できます。リージョン内の AWS 診断 AMI の AMI ID については、リージョン別の AMI ID以下を参照してください。
- [インスタンスタイプ] では、[t2.micro] を選択することをお勧めします。
- [ネットワーク設定] では、レプリケーションインスタンスが使用しているのと同じ VPC を選択します。

インスタンスがアクティブになったら、インスタンスに接続します。Linux インスタンスに接続する方法については、「Linux インスタンスへの接続」を参照してください。

IAM ロールを作成する

必要最小限のアクセス権限を使用してレプリケーションインスタンスで診断テストを実行する場合は、次のアクセス権限ポリシーを使用する IAM ロールを作成します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
```

```
                "dms:DescribeEndpoints",
                "dms:DescribeTableStatistics",
                "dms:DescribeReplicationInstances",
                "dms:DescribeReplicationTasks",
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "*"
        }
    ]
}
```

このロールを新しい IAM ユーザーにアタッチします。IAM ロール、ポリシー、ユーザーの作成の詳細については、「IAM ユーザーガイド」の次のトピックを参照してください。

- IAM の使用開始
- IAM ロールの作成
- IAM ポリシーの作成

診断テストを実行する

Amazon EC2 インスタンスを作成して接続したら、次の手順を実行してレプリケーションインスタンスで診断テストを実行します。

1. AWS CLI を設定します。

   ```
   $ aws configure
   ```

   診断テストの実行に使用する AWS ユーザーアカウントのアクセス認証情報を指定します。VPC とレプリケーションインスタンスのリージョンを指定します。

2. リージョンで使用可能な AWS DMS タスクを表示します。サンプルリージョンを実際のリージョンに置き換えます。

   ```
   $ dms-report -r us-east-1 -l
   ```

   このコマンドは、タスクのステータスを表示します。

3. タスクエンドポイントと設定を表示します。*<DMS-Task-ARN>* をタスクの Amazon リソースネーム (ARN) に置き換えます。

```
$ dms-report -t <DMS-Task-ARN>
```

このコマンドは、タスクのエンドポイントと設定を表示します。

4. 診断テストを実行します。*<DMS-Task-ARN>* はタスク ARN に置き換えます。

```
$ dms-report -t <DMS-Task-ARN> -n y
```

このコマンドは、レプリケーションインスタンスの VPC、ネットワークパケット送信、ネットワークのレイテンシー、ネットワークの最大送信単位 (MTU) のサイズに関する診断データを表示します。


```
########
#
#
#       AWS DMS Diagnostic
#       Date: 07-13-2022
#
#
#       aws region: us-east-2
#
#
########
==== DMS DIAG Info ====
Public IP: 3.
Private IP: 172.30.0.240
Instance ID: i-04829b2beb8214602
Instance MAC: 02:58:04:b5:52:28
Instane Type: t2.micro
Instance Sec Group: D
Instance AWS Region: us-east-2
Instance VPC Id: vpc-08ba020355d8a952e


==== Network Packet Check ====
1.) Check DMS EC2 MetaData service
>>>>>Result: 10 packets transmitted, 10 packets received, 0% packet loss
     Looks good with no issue. <<<<<

2.) Check Source endpoint (d
>>>>>Result: 10 packets transmitted, 10 packets received, 0% packet loss
     Looks good with no issue. <<<<<

3.) Check Target endpoint (r
>>>>>Result: 10 packets transmitted, 10 packets received, 0% packet loss
     Looks good with no issue. <<<<<

==== End network packet check ====


==== Network Latency Check ====
1.) Check DMS MetaData Service
>>>>>Result: round-trip min/avg/max = 0.4/0.4/0.5 ms
     Looks good with no issue. <<<<<

2.) Check Source endpoint (d
>>>>>Result: round-trip min/avg/max = 1.0/1.1/1.2 ms
     Looks good with no issue. <<<<<

3.) Check Target endpoint (r
>>>>>Result: round-trip min/avg/max = 1.4/1.4/1.5 ms
     Looks good with no issue. <<<<<

==== End network latency check ====


==== Network MTU Check ====
1.) Check DMS MetaData Service
>>>>>Result: MTU setting looks good. Local MTU (9001) matches remote MTU (9001) <<<<<


2.) Check Source endpoint (d
>>>>>Result: MTU setting looks good. Local MTU (9001) matches remote MTU (9001) <<<<<


3.) Check Target endpoint (r
>>>>>Result: MTU setting looks good. Local MTU (9001) matches remote MTU (9001) <<<<<


==== End network MTU check ====
```

次のステップ

次のセクションでは、ネットワーク診断テストの結果に基づいたトラブルシューティング情報について説明します。

VPC テスト

このテストでは、診断 Amazon EC2 インスタンスがレプリケーションインスタンスと同じ VPC 内にあることを検証します。診断 Amazon EC2インスタンスがレプリケーションインスタンスと同じ VPC にない場合は、これを終了し、適切な VPC で再作成します。Amazon EC2インスタンスの VPC は、作成後に変更することはできません。

ネットワークパケットロスのテスト

このテストでは、10 個のパケットを次のエンドポイントに送信し、パケットロスを確認します。

- ポート 80 の AWS DMS Amazon EC2 メタデータサービス
- ソースエンドポイント
- ターゲットエンドポイント

すべてのパケットは正常に到着するはずです。パケットロスがあった場合は、ネットワークエンジニアに相談して問題を特定し、解決策を模索します。

ネットワークレイテンシーのテスト

このテストでは、以前のテストと同じエンドポイントに 10 個のパケットを送信し、パケットのレイテンシーを確認します。すべてのパケットのレイテンシーは 100 ミリ秒未満である必要があります。レイテンシーが 100 ミリ秒を超えるパケットがある場合は、ネットワークエンジニアに相談して問題を特定し、解決策を模索します。

最大送信単位 (MTU) サイズのテスト

このテストでは、以前のテストと同じエンドポイントで Traceroute ツールを使用して MTU のサイズを検出します。テスト範囲のすべてのパケットの MTU サイズが同じである必要があります。MTU サイズが異なるパケットがある場合は、ネットワークエンジニアに相談して問題を特定し、解決策を模索します。

リージョン別の AMI ID

AWS リージョンで使用可能な DMS Diagnostic AMIs のリストを表示するには、次の AWS CLI サンプルを実行します。

```
aws ec2 describe-images --owners 343299325021 --filters "Name=name, Values=DMS-DIAG*"
 --query "sort_by(Images, &CreationDate)[-1].[Name, ImageId, CreationDate]" --output
 text
```

出力に結果が表示されない場合は、DMS Diagnostic AMI がお客様の AWS リージョンで使用できないことを意味します。回避策は、以下の手順に従って、診断 AMI を別のリージョンからコピーすることです。詳細については、「AMI のコピー」を参照してください。

- 使用可能なリージョンでインスタンスを起動します。
- イメージを作成します。イメージはユーザーが所有します。
- AMI を中東 (UAE) リージョンなどのリージョンにコピーします。
- ローカルリージョンでインスタンスを起動します。

AWS DMS リファレンス

このリファレンスセクションには、データ型変換の情報など、AWS Database Migration Service (AWS DMS) を使用する際に必要になる可能性がある追加情報が記載されています。

AWS DMS は、ソースとターゲットの両方が同じエンジンタイプを使用する同種のデータベース移行を実行する際にデータ型を維持します。あるデータベースエンジンタイプから別のデータベースエンジンに移行する異種移行を実行する場合は、データ型は中間データ型に変換されます。ターゲットデータベースでデータ型がどのように表示されるかを確認するには、ソースデータベースエンジンとターゲットデータベースエンジンのデータ型テーブルを参照してください。

データベースを移行する際は、データ型に関するいくつかの重要な点に注意します。

- FLOAT データ型は本質的に近似値です。FLOAT 型のフィールドに特定の値を挿入すると、データベース内で値が別の方法で表現される可能性があります。この違いは、FLOAT が NUMBER や NUMBER(p,s) などの 10 進法のデータ型のように正確なデータ型ではないためです。そのため、データベースに格納される FLOAT の内部値は、挿入した値と異なる可能性があります。したがって、移行された FLOAT の値は、ソースデータベースの値と正確には一致しない可能性があります。

  この問題の詳細については、次の記事を参照してください。

  - Wikipedia の「IEEE 浮動小数点」
  - Microsoft Learn の「IEEE 浮動小数点表現」
  - Microsoft Learn の「浮動小数点数の精度の低下」

トピック

- AWS Database Migration Service のデータ型

AWS Database Migration Service のデータ型

AWS Database Migration Service は、組み込みデータ型を使用して、ソースデータベースエンジンタイプからターゲットデータベースエンジンタイプにデータを移行します。組み込みのデータ型と説明は、次の表のとおりです。

AWS DMS のデータ型	説明
STRING	文字列
WSTRING	ダブルバイト文字列
BOOLEAN	ブール値
BYTE	バイナリデータ値
DATE	日付値: 年、月、日
TIME	時間値: 時、分、秒
DATETIME	タイムスタンプ値: 年、月、日、時、分、秒、小数秒。小数点以下の秒の位取りは最大 9 桁です。サポートされる形式: YYYY:MM:DD HH:MM:SS.F(9)。Amazon S3 Select と Amazon S3 Glacier Select では、DATETIME データ型の形式が異なります。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「サポートされているデータ型」に記載の `timestamp` プリミティブデータ型の説明を参照してください。
INT1	1 バイトの符号付き整数
INT2	2 バイトの符号付き整数
INT4	4 バイトの符号付き整数
INT8	8 バイトの符号付き整数
NUMERIC	固定精度と位取りを持つ正確な数値
REAL4	単精度浮動小数点値
REAL8	倍精度浮動小数点値
UINT1	1 バイトの符号なし整数

AWS DMS のデータ型	説明
UINT2	2 バイトの符号なし整数
UINT4	4 バイトの符号なし整数
UINT8	8 バイトの符号なし整数
BLOB	バイナリラージオブジェクト
CLOB	文字ラージオブジェクト
NCLOB	ネイティブ文字ラージオブジェクト

> **Note**
>
> AWS DMS は、LOB データ型を Apache Kafka エンドポイントに移行できません。

AWS DMS リリースノート

Database AWS Migration Service () の現在および以前のバージョンのリリースノートを以下に示しますAWS DMS。

AWS DMS レプリケーションインスタンスの自動バージョンアップグレードを有効にしても、 はメジャーバージョンとマイナーバージョンを区別しません。DMS は、そのバージョンが廃止された場合、メンテナンスウィンドウ中にレプリケーションインスタンスのバージョンを自動的にアップグレードします。

レプリケーション インスタンスのバージョンを (API または CLI を使用して) バージョン 3.4.x から 3.5.x に手動でアップグレードするには、 `AllowMajorVersionUpgrade`パラメータを に設定する必要がありますtrue。`AllowMajorVersionUpgrade` パラメータの詳細については、DMS API ドキュメント ModifyReplicationInstanceの「」を参照してください。

> **Note**
>
> の現在のデフォルトエンジンバージョン AWS DMS は 3.5.1 です。

次の表は、アクティブな DMS バージョンの次の日付を示しています。

- バージョンのリリース日
- バージョンで新しいインスタンスを作成できない日付
- DMS がそのバージョンのインスタンスを自動的に更新する日付 (終了日)

バージョン	リリース日	新しいインスタンスの日付なし	サポート終了日
3.5.3	2024 年 5 月 17 日	2025 年 8 月 31 日	2025 年 10 月 31 日
3.5.2	2023 年 10 月 29 日	2025 年 3 月 30 日	2025 年 4 月 29 日
3.5.1	2023 年 6 月 30 日	2024 年 11 月 30 日	2025 年 1 月 30 日
3.4.7	2022 年 5 月 31 日	2024 年 7 月 30 日	2024 年 8 月 29 日
3.4.6	2021 年 11 月 30 日	2024 年 5 月 26 日	2024 年 6 月 27 日

AWS Database Migration Service 3.5.3 リリースノート

AWS DMS 3.5.3 の新機能

新機能または拡張機能	説明
Babelfish サポート用の拡張 PostgreSQL ソースエンドポイント	AWS DMS は、Babelfish データ型をサポートするように PostgreSQL ソースエンドポイントを拡張しました。詳細については、「AWS DMS ソースとしての PostgreSQL データベースの使用」を参照してください。
ソースとしての S3 Parquet のサポート	AWS DMS はソースとして S3 Parquet をサポートします。詳細については、「のソースとしての Amazon S3 の使用 AWS DMS」を参照してください。
PostgreSQL 16.x のサポート	AWS DMS は PostgreSQL バージョン 16.x をサポートしています。詳細については、「AWS DMS ソースとしての PostgreSQL データベースの使用」および「PostgreSQL データベースの AWS Database Migration Serviceのターゲットとしての使用」を参照してください。
フルロード Oracle から Amazon Redshift への移行のスループットの向上	AWS DMS Serverless は、Oracle から Amazon Redshift への全ロード移行のスループットパフォーマンスを大幅に向上させます。詳細については、「フルロード Oracle から Amazon Redshift への移行のスループットの向上」を参照してください。

AWS DMS バージョン 3.5.3 には、次の解決済みの問題が含まれています。

2024 年 5 月 17 日付けの DMS 3.5.3 リリースで解決された問題

解決済みの問題	説明
データ検証オーバーライド関数	`override-validation-function` テーブルマッピングでルールアクションが に設定されている場合、DMS がソースフィルタリングを受け付けないデータ検証機能の問題を修正しました。
MySQL ソース CDC エラー	ソースとしての MySQL で CDC 移行が UTF16 エンコーディングで失敗する問題を修正しました。

解決済みの問題	説明
データ検証照合の違い	列フィルタリングが使用されたときに DMS がHandleCollationDiff タスク設定を適切に適用しないデータ検証機能の問題を修正しました。
データ検証タスクのハング。	DMS タスクが null でハングするデータ検証機能の問題targetを修正しました。
PostgreSQL から PostgreSQL へのレプリケーションのタスク障害。	PostgreSQL から PostgreSQL への移行で、CDC レプリケーション中に LOB データをターゲットに挿入するときに DMS タスクが失敗する問題を修正しました。
PostgreSQL をソースとするデータ損失	特定のエッジケースシナリオでデータ損失が発生するソースとしての PostgreSQL の問題を修正しました。
MySQL 5.5 ソース CDC エラー	MySQL バージョン 5.5 で CDC レプリケーションが失敗するソースとしての MySQL の問題を修正しました。
Oracle ソース IOT テーブルの問題。	ソースとしての Oracle で、すべての列でサプリメンタルログが有効になっている IOT テーブルに対して DMS がUPDATEステートメントを正しくレプリケートしない問題を修正しました。
MySQL ソース LOBS	LOBs が Redshift で許可されている最大サイズを超えたために DMS タスクが失敗する MySQL から Redshift への移行の問題を修正しました。
での検証の問題 SkipLobColumns	プライマリキーがソーステーブルの最後の列にあったSkipLobColumns = true ときに DMS タスクが失敗するデータ検証機能の問題を修正しました。
一意のキーが である検証をスキップする null	DMS が null 一意キーを持つ行を適切にスキップしないデータ検証機能の問題を修正しました。
Oracle COLLATEオペレーターのデータ検証の改善。	データ検証機能で、12.2 より前のバージョンの Oracle で構文エラーが発生して検証が失敗する問題を修正しました。

解決済みの問題	説明
全ロード中のエラー処理	ターゲットとしての PostgreSQL で、無効なデータが原因でテーブルエラーが発生した後、全ロードフェーズ中にタスクがハングする問題を修正しました。
CDC 検証のみのタスクの再検証	データ検証機能を強化して、CDC 検証のみのタスクで再検証できるようにしました。
ターゲットCdcMaxBatchInterval Out of Memory問題としての S3	ターゲットとしての S3 で、 CdcMaxBatchInterval が設定されたメモリ不足状態で DMS タスクが失敗する問題を修正しました。
Oracle ソースドライバー	DMS Oracle ソースドライバーを v12.2 から v19.18 にアップグレードしました。
SQL Server ソースでの LOB 切り捨て警告	CDC 中の LOB 切り捨てに関する警告を表示するソースとしての SQL Server のログ記録を強化しました。
Oracle バイナリリーダーの機能強化	Oracle ソースバイナリリーダーが以下をサポートするように拡張されました。 • Big Endian プラットフォーム • HCC 圧縮による並列 DML ヒント • Golden Gate を有効にした高度な Oracle 圧縮

AWS Database Migration Service 3.5.2 リリースノート

AWS DMS 3.5.2 の新機能

新機能または拡張機能	説明
Redshift データ検証	AWS DMS で Redshift ターゲットでのデータの検証がサポートされるようになりました。
Microsoft SQL Server バージョン 2022 をソースおよ	AWS DMS では、Microsoft SQL Server バージョン 2022 をソースおよびターゲットとして使用できるようになりました。

新機能または拡張機能	説明
びターゲットとしてサポート	
IBM Db2 LUW をターゲットとしてサポート	AWS DMS が IBM Db2 LUW をターゲットとしてサポートするようになりました。を使用して AWS DMS、IBM Db2 LUW から IBM Db2 LUW へのライブ移行を実行できるようになりました。

AWS DMS バージョン 3.5.2 には、次の解決済みの問題が含まれています。

2024 年 4 月 29 日付けの DMS 3.5.2 メンテナンスリリースで解決された問題

解決済みの問題	説明
IBM Db2 ターゲットセグメント化全ロード	IBM Db2 をターゲットとするセグメント化された全ロードのサポートが追加されました。
ターゲット設定としての Amazon Timestream	Timestream をターゲットとする無効なタイムスタンプ設定とサポートされていないテーブルオペレーションの処理を強化しました。
列フィルターを使用したタスクのクラッシュ	DMS が変換ルールを使用して動的に追加した列でフィルターを使用しているときにタスクがクラッシュする問題を修正しました。
トランザクションスワップファイルの読み取りのログ記録	DMS がトランザクションスワップファイルからいつ読み取っているかを示すログを追加しました。
を使用したターゲットとしての S3 CdcInsertsAndUpdates	ターゲットとしての S3 で、 が で、 CdcInsertsAndUpdates が の場合にタスクtruePreserveTransactions がクラッシュする問題を修正しましたtrue。
ソースフィルターの負の演算子	負の演算子に設定すると、同じ列に変換ルールが定義されている場合に、ソースフィルターオペレーターの動作が正しくない問題を修正しました。

解決済みの問題	説明
DMS がソースからの読み取りを一時停止したときのログ記録を追加	パフォーマンスを向上させるために DMS がソースからの読み取りを一時的に一時停止するタイミングを示すログ記録を強化しました。
エスケープ文字を含むソースフィルター	ソースフィルターで、CDC 中に DMS がエスケープされた文字を新しく作成されたテーブルに適用する問題を修正しました。
ターゲットとしての PostgreSQL、誤ってレプリケートされた削除	ターゲットとしての PostgreSQL で、DMS が Null 値として削除をレプリケートする問題を修正しました。
ソースとしての Oracle のログ記録の改善	外部エラーコードを削除するためのソースとしての Oracle のログ記録を強化しました。
XMLTYPE 制限のログ記録を改善	DMS がXMLTYPEデータ型の完全な LOB モードをサポートしていないことを示すために、ソースとしての Oracle のログ記録が改善されました。
MySQL データ損失	ターゲットとしての MySQL で、列メタデータが破損してタスクがクラッシュしたり、データが失われたりする問題を修正しました。
新しい列に適用されるフィルター	フルロード中に、DMS が変換ルールが新しい列に追加するフィルターを無視する問題を修正しました。
ターゲットとしての S3: 検証の問題	S3 をターゲットとする場合に、検証パーティション定義が異なる複数のテーブルを移行する際にデータ検証が失敗する問題を修正しました。
CDC のみのタスククラッシュ	CDC のみのタスクで、 が のときにタスクがクラッシュする問題TaskRecoveryTableEnabled を修正しましたtrue。
MySQL と MariaDB の互換性のない照合順序	MySQL から MariaDB への移行で、DMS がtf8mb4_0900_ai_ci 照合で MySQL v8 テーブルを移行しない問題を修正しました。
でタスクがクラッシュするBatchApplyEnabled	特定の条件下でタスクが失敗するバッチ適用機能の問題を修正しました。

解決済みの問題	説明
Amazon DocumentDB の UTF-8 以外の文字	Amazon DocumentDB エンドポイントの UTF-8 以外の文字のサポートが追加されました。
バッチ適用タスクのクラッシュ	バッチ適用機能で、大きなトランザクションのレプリケート中に DMS タスクがクラッシュする問題を修正しました。
Db2 トランザクションのロールバック処理	ソースとして Db2 を使用する場合、DMS はソースにロールバックされているにもかかわらず、 INSERTをターゲットにレプリケートする問題を修正しました。
ソースフィルターによる検証	検証がソースフィルターを尊重しない問題を修正しました。

AWS Database Migration Service 3.5.1 リリースノート

AWS Database Migration Service (AWS DMS) バージョン 3.5.1 で導入された新機能と拡張機能は、次の表のとおりです。

新機能または拡張機能	説明
PostgreSQL 15.x の サポート	AWS DMS バージョン 3.5.1 は PostgreSQL バージョン 15.x をサポートしています。詳細については、「PostgreSQL のソースとしての使用」および「ターゲットとしての PostgreSQL の使用」を参照してください。
Amazon DocumentDB Elastic Clusters のシャードコレクションのサポート	AWS DMS バージョン 3.5.1 は、シャードコレクションを持つ Amazon DocumentDB Elastic Clusters をサポートしています。詳細については、「AWS Database Migration Service のターゲットとしての Amazon DocumentDB の使用」を参照してください。
ターゲットとしての Redshift Serverless	ターゲットエンドポイントとしての Amazon Redshift Serverless のサポートを追加しました。詳細については、「AWS Database Migration Service のターゲットとしての Amazon Redshift データベースの使用」を参照してください。

新機能または拡張機能	説明
Babelfish のエンドポイント設定	Babelfish のサポートを提供するための PostgreSQL ターゲットエンドポイント設定が強化されました。詳細については、「PostgreSQL データベースの AWS Database Migration Service のターゲットとしての使用」を参照してください。
Oracle ソースのオープントランザクション	AWS DMS 3.5.1 では、Oracle ソースの開始位置から CDC 専用タスクを開始するときに、オープントランザクションを処理する方法が改善されています。詳細については、`OpenTransactionWindow` セクションの のソースとして Oracle を使用する場合のエンドポイント設定 AWS DMS を参照してください。
ターゲットとしての Amazon Timestream	Amazon Timestream をターゲットエンドポイントとして使用するためのサポート。詳細については、「Amazon Timestream を AWS Database Migration Service のターゲットとして使用する」を参照してください。

AWS DMS バージョン 3.5.1 には、次の解決済みの問題が含まれています。

解決済みの問題	説明
ソースとしての Oracle が非アクティブセッションを増やす	Oracle ソースで、CDC のみのタスクで非アクティブなセッションが継続的に増加し、 の例外が発生する問題を修正しました`ORA-00020: maximum number of processes exceeded on the source database`。
UPDATE の変更を DocumentDB にレプリケートする	ターゲットとしての DocumentDB で、一部のシナリオで UPDATE ステートメントが適切にレプリケートされない問題を修正しました。
検証のみのタスク	検証のみのタスクでデータ検証が無効になっている場合に、データ検証機能が適切にタスクに失敗するようにエラー処理が改善されました。
接続終了後の Redshift レプリケーション	Redshift ターゲットで、接続終了後にターゲットが 0 より大きく`ParallelApplyThreads` 設定された場合に DMS タスクが

解決済みの問題	説明
	ターゲットへの変更の適用を再試行しない問題を修正しました。これにより、データが失われます。
MySQL テキストからミディアムテキストへのレプリケーション	フル MySQL LOB モードでの中文データ型の MySQL から MySQL へのレプリケーションの問題を修正しました。
CDC タスクがローテーションされたシークレットでレプリケートされない	Secrets Manager がパスワードを更新した後に DMS `true`がデータのレプリケーションを停止する を `BatchApplyEnabled` に設定した場合の DMS タスクの問題を修正しました。
MongoDB /DocumentDB セグメンテーションの問題	MongoDB / DocDB ソースで、プライマリキー列に大きな値が含まれている場合、範囲セグメンテーションが正しく機能しない問題を修正しました。
無制限の数値の Oracle データ検証	Oracle ターゲットで、データ検証`STRING`中に DMS がバインドされていないデータ型の値を `NUMERIC`として認識する問題を修正しました。
SQL Server データ検証	DMS データ検証で無効な SQL ステートメントが構築された SQL Server エンドポイントの問題を修正しました。
MongoDB 自動セグメンテーション	ソースとして MongoDB を使用する場合のドキュメントを並列移行する際のデータの自動パーティション分割機能を改善しました。
Amazon S3 Apache Parquet の形式	S3 にターゲットとして書き込まれた Apache Parquet ファイルを Apache Arrow C++ を使用した Python で表示できるように問題を修正しました。
PostgreSQL をソースとする場合の DDL の処理	サポートされていない DDL オペレーションが適切に無視されないという PostgreSQL をソースとする場合の問題を修正しました。
PostgreSQL の `timestamptz` データエラー	PostgreSQL 間移行で、CDC 中にバッチ適用が有効になっていると、タイムゾーンのデータなどのタイムスタンプが適切に移行されない問題を修正しました。

解決済みの問題	説明
Oracle から PostgreSQL の検証障害	Oracle から PostgreSQL への移行で、NUMERIC (38,30) データ型のデータ検証が失敗する問題を修正しました。
Oracle 拡張データ型エラー	Oracle ソースで、拡張 varchar データ型が切り捨てられていた問題を修正しました。
フィルター演算子の組み合わせ	NULL 列演算子をその他のタイプの演算子と組み合わせることができないという列フィルタリング機能の問題を修正しました。
過剰なロギングを原因とする CDC レイテンシー	PostgreSQL をソースとする場合に pglogical プラグイン警告の過剰なロギングがソース CDC レイテンシーを引き起こしていた問題を修正しました。
Create Table DDL の双方向レプリケーション処理	PostgreSQL 間の双方向レプリケーションで、Create Table DDL の変更が適切にレプリケートされなかった問題を修正しました。
フィルターの使用時の CDC 障害	CDC レプリケーションが失敗するフィルタリング機能の問題を修正しました。
Kafka エンドポイントの認証機関ホスト名検証	認証機関のホスト名検証を無効にするオプション (`SslEndpointIdentificationAlgorithm`) を追加して Kafka エンドポイントの機能を強化しました。
IBM Db2 LUW 検証	Db2 LUW ソースの date、timestamp、time のデータ型がデータ検証中に適切に処理されない問題を修正しました。
S3 の検証	Db2 LUW から S3 への移行で、検証機能が timestamp(0) データ型の処理が適切でなかった問題を修正しました。
DMS タスク再起動の障害	AWS DMS タスクの再起動に失敗し、pglogical プラグインの使用時にリレーショナルイベントを消費できなかった PostgreSQL ソースの問題を修正しました。
SQL Server の HIERARCHY データ型の検証	SQL Server ソースで HIERARCHY データ型の検証が失敗する問題を修正しました。

解決済みの問題	説明
制御文字がある SQL Server の文字列	SQL Server ソースで、制御文字がある文字列が適切にレプリケートされない問題を修正しました。
Redshift での Secrets Manager の使用	Redshift をターゲットとする場合に、Secrets Manager を使用するとエンドポイントのテストが失敗する問題を修正しました。
MySQL ParallelLoadThreads 設定の不整合	MySQL をターゲットとする場合に、タスク設定を変更すると `ParallelLoadThreads` 設定が適切に保持されない問題を修正しました。
PostgreSQL から Oracle へのデータ型マッピングエラー	PostgreSQL から Oracle への移行で、TEXT データ型から VARCHAR2(2000) データ型にレプリケートするとタスクが失敗する問題を修正しました。
Oracle から PostgreSQL へのデータ検証	Oracle から PostgreSQL への移行で、NULL 文字が SPACE 文字としてレプリケートされた場合にデータ検証で誤検出が報告される問題を修正しました。
AlwaysOn 設定中の SQL Server ソース	SQL Server ソースが AlwaysOn 混同で、レプリカ名が実際のサーバー名と正確に一致しないと AWS DMS タスクが失敗する問題を修正しました。
Oracle ソースエンドポイントのテスト障害	Oracle ソースで、Oracle セッション ID (SID) の取得中に権限が不十分なために AWS DMS エンドポイント接続テストが失敗する問題を修正しました。
CDC が新しいテーブルを取得しない	CDC のみのタスクで、タスクの開始後にソースで作成されたテーブルがレプリケートされない場合がある問題を修正しました。
ソースとしての Oracle でのオープントランザクション	Oracle ソースの開始位置から CDC のみのタスクを開始する際のオープントランザクションの処理方法が改善されています。

解決済みの問題	説明
データ欠落の問題	(StopTaskCachedChangesApplied オプションが true の場合) キャッシュした変更が適用された後にタスクが停止されてタスクを再開すると、データが欠落する問題を修正しました。この問題は、ソースの大量の変更により、 が AWS DMS AWS DMS レプリケーション インスタンス ディスクへのキャッシュされた変更を永続化した場合にまれに発生する可能性があります。
拡張データ型のデータ検証に関する問題	PostgreSQL から Oracle へのデータ検証で、拡張データ型の検証が失敗する問題を修正しました。
一貫性のない文字エンコーディングのデータ検証に関する問題	SQL Server から PostgreSQL への移行で、文字エンコーディングがソースとターゲットの間で一貫していない場合に検証が失敗するというデータ検証の問題を修正しました。
データ検証に関する問題 ORA-01455	PostgreSQL の integer を Oracle にマップしている場合の検証中に Oracle の number(10) エラーが発生する問題を修正しました。
SQL Server の IDENTITY のサポート	SQL Server 間のデータレプリケーションで、ターゲット列に IDENTITY プロパティがあると identity 列の移行が失敗する問題を修正しました。
ALTER ステートメントの文字セットの問題	MySQL から MySQL へのレプリケーションで AWS DMS 、CDC 中にALTERステートメントを移行するときに文字セットが UTF16 に変更される問題を修正しました。
PostgreSQL から Redshift への空間データ型のサポート	PostgreSQL から Amazon Redshift に移行する場合の spatial データ型のサポートが追加されました。
.parquet ファイルの GZIP 圧縮	が S3 をターゲットとする GZIP 圧縮で .parquet ファイルを生成 AWS DMS できない問題を修正しました。
MongoDB と DocDB のソースの移行	AWS DMS が一部のパーティションを MongoDB ソースから移行しない問題を修正しました。

解決済みの問題	説明
テーブル統計の問題	レプリケーションインスタンスの少なくとも 1 つのタスクに 1,001 を超えるテーブルが含まれている場合にテーブル統計が表示されない問題を修正しました。
IBM Db2 LUW バージョン 10.1.0 以前でのテーブル停止	Db2 LUW ソースで、ソースデータベースのバージョンが 10.1.0 以下の場合にテーブルの移行が `TYPESTRINGUNITS is not valid` エラーで停止する問題を修正しました。
MongoDB のパーティション分割の問題	ソースのパーティションが単一または複数のセグメントで欠落する MongoDB/DocDB の問題を修正しました。
MongoDB のパーティション分割の問題	NumberLong() 型の列に基づくセグメンテーションが、型変換のバグにより失敗する問題を修正しました。
MongoDB のパーティション分割の問題	MongoDB をソースとする大規模なデータセットの自動セグメント化のパフォーマンスが向上しました。
MongoDB ドライバーバージョン	MongoDB バージョン 3.6 以前を引き続きサポートするため、MongoDB ドライバーを 1.20.0 にダウングレードしました。
Amazon S3 Apache Parquet の timestamp データ型	Amazon S3 parquet target. AWS DMS now では、以前のバージョンの の動作と一致する`true`ように 形式パラメータが `isAdjustedToUTC` に設定される問題を修正しました AWS DMS。
Amazon Redshift ターゲットの copy コマンド	ターゲットとして Amazon Redshift を使用する場合、Amazon S3 から Amazon Redshift にデータをコピーする際、サイズが大きいテーブルに対して copy コマンドが失敗するという問題を修正しました。
PostgreSQL の geometry データ型	PostgreSQL 間の移行で、サイズが大きい geometry データ型で移行が失敗する問題を修正しました。
Oracle から PostgreSQL	Oracle から PostgreSQL へのレプリケーション時に移行により XML に余分なスペースが追加される問題を修正しました。

解決済みの問題	説明
サポートされているエンジンでのターゲットチェックポイントの更新	AWS DMS は、ターゲットデータベースの `awsdms_txn_state` テーブル内のターゲットチェックポイントを更新するようになりました。
MongoDB と DocDB のレコードの誤ったコレクションへの送信	MongoDB と DocDB でデータが誤ったターゲットコレクションに送信される問題を修正しました。
EscapeCharacter エンドポイント設定による Oracle ソースの新しいテーブルの選択	Oracle ソースで、`EscapeCharacter` エンドポイント設定の設定中にタスクが停止および再開されたときに、 AWS DMS がレプリケーション用の新しいテーブルのみを取得する問題を修正しました。
CDC 復旧チェックポイント	ターゲットデータストアと AWS DMS コンソールの間で確認された CDC 復旧チェックポイントの不整合を修正しました。
CDC 検証のみのタスク	CDC 検証のみのタスクにおいて、タスク内のすべてのテーブルに障害が発生してもタスクが失敗しない問題を修正しました。
ソースまたはターゲットでの接続の問題についての検証動作	接続が切断されたときに AWS DMS がソースまたはターゲットのテーブルを停止するデータ検証の問題を修正しました。
Oracle から PostgreSQL へのデータ検証の誤検出	Oracle から PostgreSQL へのデータ検証で、 が誤検出 AWS DMS を報告する問題を修正しました。これは、ターゲットでのソース NULL 文字の表現の違いが、VARCHAR 以外のテキストベースのデータ型では考慮されなかったことが原因です。
Oracle から PostgreSQL へのデータの切り捨て	ソースとしての Oracle とターゲットとしての PostgreSQL において、Oracle の `NLS_NCHAR_CHARACTERSET` 設定を `AL16UTF16` に設定した場合、NVARCHAR 列のデータが AWS DMS で切り捨てられていた問題を修正しました。
データ検証エラー	ソースフィルターと列追加変換ルールの両方が使用されている場合に `unable to create where filter clause` エラーが発生するデータ検証の問題を修正しました。

解決済みの問題	説明
Redshift ターゲットのエラー処理	CDC タスクの `ParallelApplyThreads` タスク設定が 0 より大きい値に設定されていると、エラー処理が設定どおりに機能しないターゲットとして Redshift を使用する場合の問題を修正しました。
ソースとしての Oracle の通信障害	ソースとして Oracle を使用する際に、タスクが `RUNNING` の状態のままであるのに、通信障害後にデータを移行できなくなる問題を修正しました。
列フィルターによる CDC テーブルの一時停止	フルロード + CDC タスクで、列フィルターが適用されている場合、CDC フェーズ中にテーブルが停止する問題を修正しました。
ターゲットデータとしての S3 での特殊文字の検証の失敗	テーブル名にアンダースコア以外の特殊文字が含まれているとタスクが失敗する S3 ターゲットデータ検証の問題を修正しました。
MongoDB ソースのフルロードと CDC の失敗	ソースとして MongoDB を使用する場合、大規模なコレクション移行時のキャッシュイベントの処理中に フルロード + CDC タスクが失敗する問題を修正しました。
`BatchApplyEnabled` が true に設定された場合のアップグレードに関する問題	タスク設定が true に設定されている`BatchApplyEnabled` タスクが AWS DMS 、バージョン 3.4.6 から 3.5.1 に移行した後に失敗する問題を修正しました。
大文字と小文字を区別する照合順序を持つ SQL Server AlwaysOn ソース	SQL Server をソース AlwaysOn とする場合に、大文字と小文字を区別する照合順序でタスクが失敗する問題を修正しました。
MySQL ソースのタスクのハング	ソースとして MySQL を使用する際、ソースが適切に設定されていない場合にタスクが失敗せずにハングする問題を修正しました。
S3 ソースのフルロードタスクの失敗	S3 をソースとする場合に、 AWS DMS バージョン 3.4.6 または 3.4.7 からバージョン 3.5.1 にアップグレードした後にタスクが再開時に失敗する問題を修正しました。

解決済みの問題	説明
CaptureDDLs が false に設定されている PostgreSQL ソース	ソースとして PostgreSQL を使用する場合に、CaptureDDLs エンドポイント設定が false に設定されていると DDL が適切に処理されない問題を修正しました。
再開時の Oracle ソースのタスクのクラッシュ	Oracle をソースとして使用している場合の、列名のデータが正しくないために再開時にタスクがクラッシュする問題を修正しました。
MySQL ソース LOB ルックアップの失敗	ソースとして MySQL を使用する場合、ParallelApplyThreads タスク設定がゼロより大きい値に設定されていると LOB ルックアップが失敗する問題を修正しました。
SQL Server ソースの LSN 論理エラー	SQL Server をソースとする場合に、 AWS DMS バージョン 3.4.7 からバージョン 3.5.1 にアップグレードした後にタスクがillogical LSN sequencing state error エラーで失敗する問題を修正しました。
pglogical を使用する PostgreSQL ソース	PostgreSQL ソースで pglogic プラグインを使用するタスクが、タスクが停止し、選択ルールからテーブルが削除された後、タスクが再開されて、削除されたテーブルに変更が加えられた場合に失敗する問題を修正しました。
Aurora MySQL の誤ったリカバリチェックポイント	ソースとしての Aurora MySQL において、Aurora フェイルオーバーまたは Aurora ソースの停止と起動の結果として誤ったリカバリチェックポイントが保存される問題を修正しました。
ソースとしての SQL Server におけるタスクのクラッシュ	ソースとしての SQL Server において、SafeguardPolicy を RELY_ON_SQL_SERVER_REPLICATION_AGENT に設定した場合、タスクがクラッシュする問題を修正しました。
ターゲットとしての MySQL における誤ったデータ型のキャスト	ターゲットとしての MySQL において、バッチ適用フェーズで誤ったデータ型をキャストした結果として、CDC レプリケーションが失敗する問題を修正しました。

解決済みの問題	説明
ソースとしての PostgreSQL で CaptureDDLs を false に設定した場合のタスクの失敗	ソースとしての PostgreSQL において、`CaptureDDLs` エンドポイント設定を `false` に設定した場合、DDL が DML として処理されるためにタスクが失敗する問題を修正しました。
MongoDB の空のコレクションによるクラッシュ	ソースとしての MongoDB において、空のコレクションが原因でタスクがクラッシュする問題を修正しました。
ターゲットとしての Redshift におけるフルロード時のタスククラッシュ	ターゲットとしての Redshift において、リカバリチェックポイント制御テーブルが有効になっていると、フルロードフェーズ中にタスクがクラッシュする問題を修正しました。
S3 から S3 へのデータ移動なし	が指定されていない場合、 がデータをレプリケート AWS DMS しないという S3 から S3 へのレプリケーションの問題を修正`bucketFolder` しました。
`GlueCatalogGeneration` を `true` に設定した場合の CDC レイテンシー	ターゲットとしての S3 において、`GlueCatalogGeneration` を `true` に設定すると、過剰なレイテンシーが発生する問題を修正しました。
ターゲットとしての Oracle におけるデータの切り捨て	ターゲットとして Oracle を使用すると、 が VARCHAR2 列のデータ AWS DMS を切り捨てる問題を修正しました。
PostgreSQL のアンダースコアワイルドカードの動作	ソースとしての PostgreSQL において、選択ルール内のアンダースコア (_) ワイルドカードがドキュメントの説明どおりに動作しないという問題を修正しました。
ソースとしての PostgreSQL における空の WAL ヘッダーの問題	ソースとしての PostgreSQL において、レプリケーションスロットから受信した空の WAL ヘッダーが原因でタスクが失敗する問題を修正しました。
ソースとしての MySQL または MariaDB における圧縮されたバイナリログ	ソースとしての MySQL と MariaDB で、BINLOG 圧縮 AWS DMS を検出したときに適切なエラーメッセージが出力されない問題を修正しました。

解決済みの問題	説明
S3 データ検証での特殊文字	S3 データ検証が改善され、プライマリキー列と非プライマリキー列の特殊文字を処理できるようになりました。
ターゲットとしての Redshift における誤解を招くタスクログエントリ	ターゲットとしての Redshift において、タスクログ内の誤解を招くエントリにより、UPDATES と DELETE でのバッチ適用ステートメントの失敗が報告される問題を修正しました。
SQL Server から S3 への移行タスクのクラッシュ	SQL Server から S3 への移行で、キャッシュされた変更を適用するとタスクがクラッシュする問題を修正しました。
バッチ適用エラーに伴うデータの喪失	バッチ適用時にエラーが発生するとデータが失われるというバッチ適用機能の問題を修正しました。

AWS Database Migration Service 3.5.0 ベータリリースノート

> **Important**
>
> AWS DMS 3.5.0 は、レプリケーション インスタンス エンジンのベータバージョンです。は、以前のすべてのリリースと同様にこのバージョン AWS DMS をサポートします。ただし、本番環境で使用する前に AWS DMS 3.5.0 ベータをテストすることをお勧めします。

次の表は、 AWS Database Migration Service (AWS DMS) バージョン 3.5.0 Beta で導入された新機能と機能強化を示しています。

新機能または拡張機能	説明
Oracle と Microsoft SQL Server 向けの Time Travel	DMS がサポートする Oracle、Microsoft SQL Server、PostgreSQL ソースエンドポイント、および DMS がサポートする PostgreSQL と MySQL ターゲットエンドポイントを持つすべての AWS リージョンで Time Travel を使用できるようになりました。
S3 の検証	AWS DMS で Amazon S3 ターゲットエンドポイントでのレプリケートされたデータの検証がサポートされるようになりました

新機能または拡張機能	説明
	。Amazon S3 ターゲットデータの検証については、「Amazon S3 ターゲットのデータ検証」を参照してください。
Glue Catalog の統合	AWS Glue は、データを分類する簡単な方法を提供するサービスで、 と呼ばれるメタデータリポジトリで構成されます AWS Glue Data Catalog。を Amazon S3 ターゲットエンドポイント AWS Glue Data Catalog と統合し、Amazon Athena などの他の サービスを通じて Amazon S3 データをクエリできるようになりました。 AWS Amazon Athena 詳細については、「AWS DMS のための Amazon S3 ターゲットでの AWS Glue Data Catalog データカタログの使用」を参照してください。
ターゲットとしての DocumentDB の並列適用	新しいParallelApply* タスク設定でターゲットとして DocumentDB を使用すると、CDC レプリケーション中に は 1 秒あたり最大 5000 レコードをサポートする AWS DMS ようになりました。詳細については、「AWS Database Migration Service のターゲットとしての Amazon DocumentDB の使用」を参照してください。
お客様中心の口グ記録	AWS DMS バージョン 3.5.0 を使用して、タスクログをより効果的に調査および管理できるようになりました。 AWS DMS タスクログの表示と管理については、「」を参照してくださいAWS DMS タスクログの表示と管理。
Kafka ターゲットエンドポイントの SASL_PLAIN メカニズム	SASL_PLAIN 認証を使用して Kafka MSK ターゲットエンドポイントをサポートできるようになりました。
MySQL の XA トランザクションのレプリケーション	MySQL DMS ソースで XA トランザクションを使用できるようになりました。DMS 3.5.0 以前のバージョンでは、XA トランザクションの一環として適用された DML の変更は適切にレプリケートされていませんでした。
Oracle 拡張データ型	AWS DMS は、Oracle バージョン 12.2 以降の拡張データ型のレプリケーションをサポートするようになりました。

新機能または拡張機能	説明
Db2 LUW PureScale 環境	AWS DMS は、Db2 LUW PureScale 環境からのレプリケーションをサポートするようになりました。この機能は、[Start processing changes from source change position] オプションを使用する場合にのみサポートされます。
`READ_COMMITTED_SNAPSHOT` を使用する SQL Server ソース	`READ_COMMITTED_SNAPSHOT` オプションを に設定して Microsoft SQL Server ソースデータベースを使用する場合`TRUE`、強制 Lookup 接続属性を設定することで DML DataRowの変更を正しくレプリケートできます。

AWS DMS 3.5.0 には、次の解決済みの問題が含まれています。

202 AWS DMS 3 年 3 17-March-2023で解決された問題

トピック	解決方法	
Oracle—数値から変換された文字列の特殊なケースの比較	Oracle をソースとする場合に、同じ列に文字列へのデータ型変換があると、数値列に対してフィルタールールが期待どおりに機能しない問題を修正しました。	
オンプレミスの SQL Server AG の機能強化	DMS で使用されていないレプリカへの不要な接続を排除することで、AlwaysOn設定中の SQL Server ソースとの接続処理の効率が向上しました。	
SQL Server の HIERARCHYID の内部変換	SQL Server ソースで HIERARCHYID データ型が HIERARCHYID ではなく VARCHAR(250) として SQL Server ターゲットにレプリケートされていた問題を修正しました。	
S3 ターゲットの移動タスクの修正	S3 ターゲットを使用したタスクの移動に非常に長い時間がかかったり、フリーズしているように見えたり、完了しなかったりする問題を修正しました。	
Kafka の SASL Plain メカニズム	Kafka MSK ターゲットエンドポイントの SASL Plain 認証方法のサポートを導入しました。	

トピック	解決方法	
Opensearch 2.x での _type パラメータを原因とする並列ロードまたは適用の障害	Opensearch 2.x ターゲットで _type パラメータがサポートされていないために並列ロードまたは並列適用が失敗する問題を修正しました。	
混合演算子を使用したテーブルマッピングフィルターのサポート	単一の列に単一のフィルターしか適用できないという制限がなくなりました。	
S3、Kinesis、Kafka エンドポイント — CDC フェーズでの alter ベースの LOB 列の移行	Kinesis、Kafka、S3 をターゲットとする場合に、CDC 中に追加された LOB 列のデータがレプリケートされないという 問題を修正しました。	
MongoDB ドライバーのアップグレード	MongoDB ドライバーを v1.23.2 にアップグレードしました。	
Kafka ドライバーの更新	Kafka ドライバーを 1.5.3 から 1.9.2 にアップグレードしました。	
S3 エンドポイント設定の不適切な動作	S3 ターゲットで、S3 ターゲットの区切り記号として指定された文字がデータに含まれている場合に `AddTrailingPaddingCharacter` エンドポイント設定が機能しなかった問題を修正しました。	
Kinesis ターゲットタスクのクラッシュ	Kinesis をターゲットとする場合に、PK 値が空で詳細なデバッグが有効になっているとタスクがクラッシュする問題を修正しました。	
S3 ターゲットの列名が 1 つずれるケース	S3 をターゲットとする場合に、`AddColumnName` を `true`、`TimestampColumnName` を "" に設定すると、列名が 1 つ位置ずれする問題を修正しました。	

トピック	解決方法	
LOB 切り捨て警告のログ記録の改善	SQL Server をソースとする場合の LOB 切り捨てに関する警告ログが改善され、LOB の取得に使用できる SELECT ステートメントが追加されました。	
TDE パスワードが間違っている場合に DMS タスクがクラッシュするのを避けるため、致命的なエラーが追加されました。	Oracle をソースとする場合に、TDE パスワードが間違っているために DMS タスクが失敗し、エラーメッセージが表示されない状況で、意味のあるエラーメッセージが導入され、タスクがクラッシュする問題が解消されました。	
CDC 中の PostgreSQL CTAS (CREATE TABLE AS SELECT) DDL の移行の許可	CDC 中に DMS が PostgreSQL CTAS (CREATE TABLE AS SELECT) DDL をレプリケートできないという制限がなくなりました。	
CDC でのテーブルの列削除時に pg_logical タスクがクラッシュする問題を修正しました。	S3 をターゲットとするソースの PostgreSQL で、LOB のサポートが無効で LOB が存在する場合にターゲットで列の位置がずれる問題を修正しました。	
MySQL 接続処理でのメモリリークの修正	MySQL をソースとする場合に、タスクメモリ消費量が継続的に増加していた問題を修正しました。	
Oracle のソースエンドポイント設定 – `ConvertTimestampWithZoneToUTC`	TIMESTAMP WITH TIME ZONE 列と TIMESTAMP WITH LOCAL TIME ZONE 列のタイムスタンプ値を UTC に変換するには、この属性を `true` に設定します。デフォルトでは、この属性の値は「false」で、データはソースデータベースのタイムゾーンを使用してレプリケートされます。	

トピック	解決方法	
Oracle ソース - SUSPEND_TABLE の DataTruncationErrorPolicy が機能しない	S3 ターゲットの Oracle ソースで、DataTruncationErrorPolicy タスク設定が SUSPEND_TABLE に設定されているとテーブルが停止されなかった問題を修正しました。	
クエリ句の作成中での SQL Server の長いスキーマまたはテーブルの障害	SQL Server ソースで選択ルールにカンマ区切りのテーブルのリストが含まれている場合、タスクが失敗するか応答しなくなる問題を修正しました。	
MongoDB エンドポイントでの Secret Manager 認証	MongoDB と DocumentDB エンドポイントの場合に Secret Manager ベースの認証が機能していない問題を修正しました。	
NLS_NCHAR_CHARACTERSET が UTF8 に設定されている場合、DMS の CDC 中にマルチバイトの VARCHAR 列のデータを切り捨ててしまう	Oracle ターゲットを使用する Oracle ソースで NLS_NCHAR_CHARACTERSET が UTF8 に設定されているマルチバイト VARCHAR 列のデータが切り捨てられていた問題を修正しました。	
filterTransactionsOfUser ECA for Oracle LogMiner	を使用して Oracle からレプリケートするときに、DMS が指定されたユーザーからのトランザクションを無視filterTransactionsOfUser できるようにする追加の接続属性 (ECA) を追加しました LogMiner。	
バックアップで LSN が欠落している場合の SQL Server 設定の回復可能なエラー	SQL Server で LSN がない場合でもタスクが失敗しない問題を修正しました。	

AWS Database Migration Service 3.4.7 リリースノート

次の表は、 AWS Database Migration Service (AWS DMS) バージョン 3.4.7 で導入された新機能と機能強化を示しています。

新機能または拡張機能	説明
ターゲットとしての Babelfish のサポート	AWS DMS がターゲットとして Babelfish をサポートするようになりました。を使用すると AWS DMS、 AWS DMS サポートされている任意のソースから Babelfish にライブデータを最小限のダウンタイムで移行できるようになりました。 詳細については、「AWS Database Migration Service のターゲットとしての Babelfish の使用」を参照してください。
IBM Db2 z/OS データベースをフルロードのみソースとしてサポート	AWS DMS は、IBM Db2 z/OS データベースをソースとしてサポートするようになりました。を使用して AWS DMS、Db2 メインフレームから AWS DMS サポートされている任意のターゲットへのライブ移行を実行できるようになりました。 詳細については、「AWS DMS のソースとしての IBM Db2 for z/OS データベースの使用」を参照してください。
ソースとしての SQL Server リードレプリカのサポート	AWS DMS で SQL Server リードレプリカがソースとしてサポートされるようになりました。を使用して AWS DMS、SQL Server リードレプリカから AWS DMS サポートされている任意のターゲットへのライブ移行を実行できるようになりました。 詳細については、「のソースとしての Microsoft SQL Server データベースの使用 AWS DMS」を参照してください。
EventBridge DMS イベントのサポート	AWS DMS は、 for EventBridge DMS イベントを使用したイベントサブスクリプションの管理をサポートします。 詳細については、「AWS Database Migration Service での Amazon EventBridge イベントと通知の使用」を参照してください。

新機能または拡張機能	説明
VPC ソースエンドポイントとターゲットエンドポイントのサポート	AWS DMS はAmazon Virtual Private Cloud (VPC) エンドポイントをソースおよびターゲットとしてサポートするようになりました。 AWS DMS は、 AWS サービスへの明示的に定義されたルートが VPC で定義されている場合に、VPC AWS DMS エンドポイントを持つ任意のサービスに接続できるようになりました。 **Note** AWS DMS バージョン 3.4.7 以降にアップグレードするには、まず VPC エンドポイントを使用する AWS DMS か、パブリックルートを使用するように を設定する必要があります。この要件は、Amazon S3、Amazon Kinesis Data Streams、Amazon DynamoDB AWS Secrets Manager、Amazon Redshift、および Amazon OpenSearch Service のソースエンドポイントとターゲットエンドポイントに適用されます。 DynamoDB 詳細については、「AWS DMS ソースエンドポイントとターゲットエンドポイントとしての VPC エンドポイントの設定」を参照してください。
新しい PostgreSQL バージョン	PostgreSQL バージョン 14.x がソースとターゲットとしてサポートされるようになりました。
ターゲットとしての Aurora Serverless v2 のサポート	AWS DMS がターゲットとして Aurora Serverless v2 をサポートするようになりました。を使用して AWS DMS、Aurora Serverless v2 へのライブ移行を実行できるようになりました。 サポートされている AWS DMS ターゲットの詳細については、「」を参照してください「データ移行のターゲット」。

新機能または拡張機能	説明
IBM Db2 for LUW の新しいバージョン	AWS DMS は、IBM Db2 for LUW バージョン 11.5.6 および 11.5.7 をソースとしてサポートするようになりました。を使用して AWS DMS、IBM DB2 for LUW の最新バージョンからライブ移行を実行できるようになりました。 AWS DMS ソースの詳細については、「」を参照してくださいデータの移行のソース。 サポートされている AWS DMS ターゲットの詳細については、「」を参照してください「データ移行のターゲット」。

AWS DMS 3.4.7 には、次の新規または変更された動作と解決された問題が含まれています。

- Amazon S3 をソースとして使用するときに、テーブル定義の日付形式を使用してデータ文字列を解析して日付オブジェクトに変換できるようになりました。
- 新しいテーブル統計カウンター、`AppliedInserts`、`AppliedDdls`、`AppliedDeletes`、`AppliedUpdates.` が使用できるようになりました。
- をターゲット OpenSearch として使用する場合、デフォルトのマッピングタイプを選択できるようになりました。
- Oracle、PostgreSQL、SQL Server のソースの新しい `TrimSpaceInChar` エンドポイント設定を使用すると、CHAR と NCHAR データ型のデータをトリミングするかどうかを指定できます。
- Amazon S3 の新しい `ExpectedBucketOwner` エンドポイント設定を使用すると、S3 をソースまたはターゲットとして使用する場合のスナイプ攻撃を防ぐことができます。
- RDS SQL Server、Azure SQL Server、セルフマネージド型 SQL Server の場合 — DMS は、PRIMARY KEY の有無を問わず、移行タスク向けに選択されたすべてのテーブルでの MS-CDC の自動セットアップを提供するようになりました。PRIMARY KEY を持つセルフマネージド型 SQL Server での MS-REPLICATION の優先順位の有効化を考慮した一意のインデックスを持つテーブルでも MS-CDC の自動セットアップが提供されます。
- Oracle 同種移行での Oracle パーティションとサブパーティション DDL オペレーションのレプリケーションがサポートされるようになりました。
- Oracle をソースとターゲットとして使用している場合に、データ検証タスクが複合プライマリキーでクラッシュする問題を修正しました。

- Redshift をターゲットとして使用する際、ターゲット列がブール型として事前作成されている場合、可変文字型を適切にブール型にキャストする問題を修正しました。
- PostgreSQL をターゲットとして使用する場合の既知の ODBC 問題により、varchar(255) として移行された varchar データ型でのデータ切り捨ての問題を修正しました。
- Oracle をターゲットとして使用する場合、BatchApplyEnabled を true に設定し、BatchApplyPreserveTransaction を false に設定すると、DELETE オペレーションの Parallel Hint が考慮されない問題を修正しました。
- Amazon S3 の新しい AddTrailingPaddingCharacter エンドポイント設定は、S3 をターゲットとして使用するときに文字列データにパディングを追加します。
- 新しい max_statement_timeout_seconds タスク設定は、エンドポイントクエリのデフォルトタイムアウトを延長します。この設定は現時点で、MySQL エンドポイントのメタデータクエリで使用されています。
- PostgreSQL をターゲットとして使用する際、CDC タスクがエラー処理タスク設定を適切に利用していなかった問題を修正しました。
- DMS が Redis Enterprise インスタンスの Redis モードを適切に識別できなかった問題を修正しました。
- S3 ターゲットの Parquet 形式の追加接続属性 (ECA) includeOpForFullLoad のサポートを強化しました。
- 新しい PostgreSQL エンドポイント設定 migrateBooleanAsBoolean が導入されました。PostgreSQL から Redshift への移行でこの設定が true に設定されている場合、ブール型が varchar(1) として移行されます。false に設定すると、ブール型は varchar(15) として移行されます。これがデフォルトの動作です。
- SQL Server ソースを使用する際の datetime データ型に関する移行の問題が修正されました。この修正により、精度がミリ秒単位の場合に Null が挿入される問題が解決されます。
- PGLOGICAL を使用する PostgresSQL ソースの場合、CDC フェーズで pglogical を使用してソーステーブルからフィールドを削除すると、削除されたフィールドの後の値がターゲットテーブルに移行されないという移行の問題が修正されました。
- 双方向レプリケーションでレコードが繰り返し取得される SQL Server の Loopback 移行の問題を修正しました。
- PostgresSQL をソースとする場合の新しい ECA mapBooleanAsBoolean を追加しました。この追加の接続属性 を使用すると、PostgresSQL ブール値のデフォルトのデータ型マッピングを RedShift ブールデータ型に上書きできます。
- SQL Server をソースとして使用する場合に、ALTER DECIMAL と NUMERIC SCALE がターゲットにレプリケートされないという移行の問題を修正しました。

- SQL Server 2005 での接続に関する問題を修正しました。
- 2022 年 10 月 17 日の時点で、DMS 3.4.7 はレプリケーションインスタンスの第 6 世代 Amazon EC2 インスタンスクラスをサポートするようになりました。
- 2022 年 11 月 25 日の時点で、DMS 3.4.7 では、DMS Schema Conversion を使用してデータベーススキーマとコードオブジェクトを変換でき切るようになりました。また、DMS Fleet Advisor を使用して、ネットワーク環境内のデータベースを移行候補として検出できるようになりました。
- 2022 年 11 月 25 日をもって、DMS Studio は廃止されました。
- 2023 年 1 月 31 日の時点で、DMS Schema Conversion はターゲットデータプロバイダーとして Aurora MySQL と Aurora PostgreSQL をサポートするようになりました。
- 2023 年 3 月 6 日の時点で、DMS Fleet Advisor を使用してソースデータベースに適切なサイズのターゲットレコメンデーションを生成できるようになりました。
- 2023 年 3 月 6 日現在、 は、メトリクスデータポイントを Amazon に発行することを許可する AWS マネージドポリシー AWS DMS をサポートしています CloudWatch。

2023 年 5 月 5 日の DMS 3.4.7 メンテナンスリリースで解決された問題

トピック	解決方法	
PostgreSQL ソースタスクの障害	PostgreSQL をソースとする場合に、単一のイベントで許可される DDL オペレーションの最大値を超えるとタスクが失敗する問題を修正しました。	
PostgreSQL ソースのデータ検証での誤検出	Oracle をターゲットとする PostgreSQL ソースの場合に timestamp フィールドのキャストが適切でないと、データ検証で誤検出エラーが発生する問題を修正しました。	
MySQL ソースエラーの処理	MySQL をソースとする際、次の BIN ログが利用できない場合に DMS タスクが失敗しない問題を修正しました。	
MySQL ソースの ROTATE_EVENT のログ記録	ROTATE_EVENT に関連する MySQL ソースのログ記録が改善され、読み込まれる BIN ログ名が含まれるようになりました。	

トピック	解決方法
データ検証のタイムアウトの問題	データ検証に関連するクエリで executeTimeout エンドポイント設定が考慮されないデータ検証機能の問題を修正しました。
PostgreSQL ターゲットの並列フルロードでの問題	PostgreSQL をターゲットとする場合に、セグメント化された (並列) フルロードが「接続ダウン」エラーにより失敗していた問題を修正しました。
DMS タスク移動の問題	S3 をターゲットとする場合に、DMS タスクの移動オペレーションが非常に時間がかかったり、完了しなかったりする問題を修正しました。
PostgreSQL ソースのレコード重複に関する問題	PostgreSQL をソースとする場合に、タスクを停止して再開した後に DMS タスクがターゲットの重複に関連するエラーを報告する問題を修正しました。
Oracle ターゲットのデータ検証の誤検出	Oracle をターゲットとする場合に、timestamp フィールドのタイムゾーンが適切にレプリケートされていないためにデータ検証で誤検出エラーが報告される問題を修正しました。

2023 年 2 月 22 日の DMS 3.4.7 メンテナンスリリースで解決された問題

トピック	解決方法
ソースとしての SQL Server AG レプリカ	リスナー TCP ポートがレプリカ TCP ポートと異なるAlwaysOn設定で SQL Server ソースのサポートが追加されました。
Amazon Redshift ターゲットのデータ損失	Redshift をターゲットとする場合に、まれに予期しない Redshift の再起動により、ターゲットのデータが失われる場合があった問題を修正しました。

トピック	解決方法	
SQL Server ソースの Safeguard のサポート	SQL Server をソースとする場合に、エンドポイント設定 "SafeguardPolicy": "EXCLUSIVE_AUTOMATIC_TRUNCATION" をソースとする場合の が指定されていると、DMS タスクがトランザクションログバックアップを読み取れないことを示すエラーで失敗する可能性がある問題を修正しました。	
ソースとして Oracle を使用する場合のデータ検証タスクの障害	Oracle をソースとする場合にプライマリキー値が誤って識別されたために DMS タスクがデータ検証に失敗することがあった問題を修正しました。	
Kinesis の変換前イメージデータの問題	"EnableBeforeImage" タスク設定が文字データ型に対してのみ機能していたストリーミングターゲット (Kinesis、Kafka) の問題を修正しました。	
Time Travel のログファイル	Time Travel 機能で、ソースがアイドル状態の場合に DMS が 0 バイトの Time Travel ログファイルを作成していた問題を修正しました。	

2022 年 12 月 16 日の DMS 3.4.7 メンテナンスリリースで解決された問題

トピック	解決方法	
BatchApply有効	が True に設定されている場合の過剰なログ記録の問題 BatchApplyEnabled を修正しました。	
新しい MongoDB エンドポイント設定 – FullLoadNoCursorタイムアウト	MongoDB エンドポイント設定 FullLoadNoCursorTimeout では、全ロードカーソル NoCursorTimeout に を指定します。 NoCursorTimeout は MongoDB 接続設定で、アイドル状態の場合にサーバーがカーソルを閉じることを防ぎます。	
MongoDB — 単一列セグメント化のためのフィルター関数	新しいフィルター関数により、セグメント化に単一列を使用して MongoDB データベースを移行するパフォーマンスが向上します。	

トピック	解決方法	
MongoDB から Redshift	MongoDB から Redshift での移行で MongoDB コレクションがバイナリデータ型の場合、DMS が Redshift でターゲットテーブルを作成していなかった問題が修正されました。	
新しい MongoDB SocketTimeoutMS 接続属性	新しい MongoDB SocketTimeoutMS 追加接続属性は、MongoDB クライアントの接続タイムアウトをミリ秒単位で設定します。値が 0 以下の場合、MongoDB クライアントのデフォルトが使用されます。	
Amazon Kinesis タスクのクラッシュにつながる問題の修正	Amazon Kinesis Data Streams をターゲットとして移行する際、テーブルにプライマリキーが存在しない場合の null 値の処理に関する問題が修正されました。	
Oracle NULL PK/UK データ検証のサポート	NULL PK/UK 値のデータ検証がサポートされないという制限がなくなりました。	
Oracle から Amazon S3	Oracle から Amazon S3 に移行する際に、数レコードが誤って NULL として移行される問題を修正しました。	
Oracle Standby	Oracle Standby をソースとして使用する場合に、DMS がオープントランザクションを処理する機能が追加されました。	
SDO_GEOMETRY 空間データ型がある Oracle 間移行	Oracle 間移行時に、DDL でテーブルに SDO_GEOMERY 列が存在するとタスクが失敗する問題が修正されました。	
ソースとしての Oracle	Oracle をソースとして使用する際、DMS が Oracle REDO ログのシーケンス番号をスキップする場合がある問題を修正しました。	
Oracle ソース — アーカイブまたはオンライン REDO ログの欠落	Oracle をソースとして使用する際、アーカイブログがないと DMS タスクが失敗するように問題を修正しました。	

トピック	解決方法	
修正済み — DMS での Oracle Standby の REDO ログのスキップ	Oracle をソースとして使用する際、DMS が Oracle REDO ログのシーケンス番号をスキップする場合がある問題を修正しました。	
修正済み — CDC 中に Oracle 間の空間データ型がレプリケートされない	Oracle 間レプリケーションの際、CDC 中に空間データ型がレプリケートされない問題を修正しました。	
ターゲットとしての Oracle	Oracle をターゲットとして使用する際、ターゲットの適用が ORA-01747 エラーで失敗する問題を修正しました。	
Amazon S3 — テーブルデータの再ロード時のデータ損失の修正	Amazon S3 をターゲットとして使用する際に、テーブルの再ロードオペレーションで CDC ファイルが生成されない問題を修正しました。	
修正済み — プライマリサーバーをソースとする場合の SQL Server Always On コンテキスト初期化	SQL Server Always On をソースとして使用する場合、ソースがプライマリで true AlwaysOnSharedSynchedBackupIsEnabled に設定されている場合、アベイラビリティグループ (AG) を初期化しないように問題を修正しました。	
SQL Server エンドポイント設定の更新	ソースエンドポイントが SQL Server Always On 可用性グループで、 がセカンダリレプリカである場合、 が True に設定されている場合、レプリケーションタスク AlwaysOnSharedSynchedBackupsIsEnabled が失敗する問題を修正しました。	
ソースとしての PostgreSQL	mapBooleanAsブール値をサポートする 3.4.7 で導入された PostgreSQL ソースで CDC が削除/更新オペレーションを移行できない問題を修正しました。	

AWS Database Migration Service 3.4.6 リリースノート

次の表は、 AWS Database Migration Service (AWS DMS) バージョン 3.4.6 で導入された新機能と機能強化を示しています。

新機能または拡張機能	説明
AWS DMS タイムトラベル	AWS DMS は Time Travel を導入しました。Time Travel は、ログ記録機能を柔軟に行える機能であり、トラブルシューティングエクスペリエンスを強化します。Time Travel を使用すると、Amazon S3 を使用して AWS DMS ログを保存および暗号化し、特定の期間内にログを表示、ダウンロード、難読化できます。
ソースとしての Microsoft Azure SQL Managed Instance のサポート	AWS DMS は、ソースとして Microsoft Azure SQL Managed Instance をサポートするようになりました。を使用して AWS DMS、Microsoft Azure SQL Managed Instance から AWS DMS サポートされている任意のターゲットへのライブ移行を実行できるようになりました。 AWS DMS ソースの詳細については、「」を参照してくださいデータの移行のソース。 サポートされている AWS DMS ターゲットの詳細については、「」を参照してください「データ移行のターゲット」。
ソースとしての Google Cloud SQL for MySQL のサポート	AWS DMS が Google Cloud SQL for MySQL をソースとしてサポートするようになりました。を使用して AWS DMS、Google Cloud SQL for MySQL から AWS DMS サポートされている任意のターゲットへのライブ移行を実行できるようになりました。 AWS DMS ソースの詳細については、「」を参照してくださいデータの移行のソース。 サポートされている AWS DMS ターゲットの詳細については、「」を参照してください「データ移行のターゲット」。
パーティション分割されたデータの S3 への並列ロードのサポート	AWS DMS では、パーティション化されたデータの Amazon S3 への並列ロードがサポートされるようになりました。これにより、サポートされているデータベースエンジンのソースデータか

新機能または拡張機能	説明
	ら Amazon S3 へのパーティション化されたデータの移行にかかるロード時間が短縮されます。この機能により、データベースソースのテーブルのパーティションごとに Amazon S3 サブフォルダーを作成し、 AWS DMS が並列プロセスを実行して各サブフォルダーにデータを取り込むことができます。
単一のタスクでの複数の Apache Kafka ターゲットトピックのサポート	AWS DMS は、単一のタスクで Apache Kafka マルチトピックターゲットをサポートするようになりました。 AWS DMSを使用して、同じタスクを使用して、単一のデータベースから異なる Apache Kafka ターゲットトピックに複数のスキーマをレプリケートできるようになりました。これにより、同じソースデータベースの多数のテーブルを異なる Kafka ターゲットトピックに移行する必要がある場合に、複数の個別のタスクを作成する必要がなくなります。

AWS DMS 3.4.6 で解決された問題は次のとおりです。

- Amazon S3 を CSV 形式のターゲットとして使用する場合に、プライマリキー列が最初の列でない場合、UPDATE ステートメントの列が誤った列に入力される問題を修正しました。
- PostgreSQL をソースとして使用する場合、制限付き LOB モードのBYTEA列NULLの値で pglogical プラグインを使用すると AWS DMS タスクがクラッシュすることがある問題を修正しました。
- PostgreSQL をソースとして使用する場合、多数のソーステーブルが削除されると AWS DMS タスクがクラッシュすることがある問題を修正しました。
- 新しい Amazon S3 設定 `DatePartitionTimezone` が導入されて UTC 以外の日付でのパーティション分割ができるようになり、Amazon S3 の日付ベースのフォルダーのパーティション分割が改善されました。
- Redshift をターゲットとして使用する場合に、ソースの `TIMESTAMP WITH TIME ZONE` データ型から `TIMESTAMPTZ` へのマッピングがサポートされるようになりました。
- MongoDB または Amazon DocumentDB をソースとして使用する際、ワイルドカード選択ルールを使用しないタスクの CDC パフォーマンスが向上しました。
- Db2 LUW をソースとして使用する際に、アンダースコアのワイルドカードを使用し、長さが 8 未満のスキーマ名が AWS DMS タスクでキャプチャされない問題を修正しました。
- Service をターゲット OpenSearch として使用すると、 AWS DMS インスタンスが大量のデータボリュームでメモリ不足になる問題を修正しました。

- フルロード検証のみのタスクがサポートされるようになり、データ検証のパフォーマンスが向上しました。
- Sybase をソースとして使用する場合、強制フェイルオーバー後に AWS DMS タスクが再開されない問題を修正しました。
- が警告`Invalid BC timestamp was encountered in column`を誤って AWS DMS 送信する問題を修正しました。

DMS 3.4.6 メンテナンスリリースで解決された問題:

- Oracle をソースとターゲットとして使用する場合に一括適用モードが有効になっているとタスクがクラッシュする問題が修正されました。
- PostgreSQL をソースとする `ExecuteTimeout` エンドポイント設定をフルロードタスクが適切に使用するように問題を修正しました。
- PostgreSQL をソースとして使用する場合に、タスクが制限付き LOB モードに設定されている場合の配列データ型列を移行する際の問題を修正しました。
- PostgreSQL をソースとして使用する場合に、1970 年 1 月 1 日以前のタイムゾーン付きタイムスタンプを移行する際の問題を修正しました。
- SQL Server をソースとターゲットとして使用する際、DMS がレプリケーション中に空の文字列を null として処理する問題を修正しました。
- ソースとターゲットに MySQL を使用する場合、セッションの読み取り/書き込みタイムアウトのエンドポイント設定が考慮されるように問題を修正しました。
- Amazon S3 をソースとして使用する場合に、DMS CDC タスクがフルロード関連ファイルをダウンロードしていた問題を修正しました。
- Amazon S3 をターゲットとして使用する際、`CdcInsertsAndUpdates` と `PreserveTransactions` が 両方とも `true` に設定されている場合にログがクラッシュする問題を修正しました。
- ParallelApply* 機能が有効になっているときにタスクがクラッシュする問題を修正しましたが、Amazon Kinesis Data Streams をソースとして使用する場合、一部のテーブルにはデフォルトのプライマリキーがありません。
- Amazon Kinesis Data Streams をソースとして StreamArn 使用する場合に、正しくない に対してエラーが表示されない問題を修正しました。
- をターゲットとして使用すると、プライマリキーの値が空の文字列 OpenSearch としてタスクがクラッシュする問題を修正しました。
- データ検証で使用されるディスク容量が多すぎる問題を修正しました。

2022 年 12 月 13 日の DMS 3.4.6 メンテナンスリリースで解決された問題

トピック	解決方法	
SAP ASE ODBC ドライバー	SAP ASE をソースとする場合に、ODBC ドライバーが文字セットをサポートできるように問題を修正しました。	
LOB ルックアップの SQL Server datetime プライマリキーのバグ	SQL Server をソースとする場合に、プライマリキーにミリ秒単位の精度の datetime データ型がある場合に LOB ルックアップが適切に機能しない問題を修正しました。	
SQL ServerからRedshift – 「datetimeoffset」の「timestamptz」へのマッピング	SQL Server から Redshift への移行の際に、SQL Server の「datetimeoffset」形式が Redshift の「timestamptz」形式にマッピングされるようにマッピングが改善されました。	
データ検証 - SkipLobColumns True	SkipLobColumns が True で、ソースに LOB があり、プライマリキーが最後の列にあり、検証によってデータの差が検出されるときに DMS タスクがクラッシュする問題を修正しました。	
MySQL ソースのデータ検証	データ検証が有効になっている MySQL をソースとする場合に、NULL 値を持つ複合一意キーを持つテーブルを使用すると DMS タスクがクラッシュする問題を修正しました。	
ソースとしての MySQL	MySQL をソースとする場合に、精度を追加して列が変更されると、テーブルがオーバーフローエラーで停止する問題を修正しました。	
MySQL ODBC ドライバーの 8.0.23 へのアップグレード	MySQL をソースとする場合に、「utf8mb4_0900_bin」照合順序が DMS で使用される mysql ドライバーとの互換性がない問題を修正しました。	

トピック	解決方法	
MySQL – パーティション分割テーブルでの DDL 変更	新しい MySQL エンドポイント設定を導入し skipTableSuspensionForPartitionDdl 、ユーザーが CDC 中にパーティション DDL 変更のテーブル停止をスキップできるようにしました。これにより、DMS はパーティション MySQL テーブルの DDL 変更をサポートできるようになりました。	
MongoDB から Redshift への移行	MongoDB から Redshift への移行で、MongoDB コレクションにバイナリデータ型がある場合、DMS が Redshift にターゲットテーブルを作成できない問題を修正しました。	
Redshift ターゲット – 一括適用でのTime Travel セグメント化	ターゲットとしての Redshift で、DMS タスクが を true BatchApplyEnabled に設定してクラッシュする問題を修正しました。	
ターゲットとしての Redshift	Redshift をターゲットとする場合に並列ロードが type=partitions-auto に設定されていると、並列セグメントが同じテーブルディレクトリに一括 CSV ファイルを書き込み、相互干渉していた問題を修正しました。	
ターゲットとしての Redshift	Redshift をターゲットとする際に、CDC でターゲット列のデータ型がブール型であるのに対してソースのデータ型が文字変化型となる問題を修正しました。	
ターゲットとしての Redshift	Redshift をターゲットとする際に、レプリケートに失敗した DDL 変更を特定できるようにタスクログを改善しました。	
PostgreSQL のデータ検証	PostgreSQL での検証で、ブール値のデータ型があると検証が失敗する問題を修正しました。	
ソースとしての PostgreSQL	PostgreSQL をソースとして使用する場合の問題を修正し、全ロードで追加の接続属性の ExecuteTimeout フィールドを使用するようにしました。	

トピック	解決方法	
ソースとしての PostgreSQL	PostgreSQL をソースとする場合に、リクエストされたタスク再開 LSN より大きい LSN を 60 分以上読み取る際にタスクが失敗し、使用されているレプリケーションスロットに問題があると表示するように問題を修正しました。	
ソースとしての PostgreSQL – 1970 年 1 月 1 日 以前の timestamptz	PostgreSQL をソースとする場合に、1970 年 1 月 1 日 以前の timestamptz が CDC 中に正しく移行されなかった問題を修正しました。	
ソースとしての PostgreSQL	PostgreSQL をソースとする場合に、CDC 中に DMS が文字のさまざまなデータ型値を切り捨てる問題を修正しました。	
ソースとしての PostgreSQL – 停止したタスクの再開	PostgreSQL をソースとする場合に、以前に停止したタスクのリプレイを再開すると、CDC 中に 単一または複数のトランザクションが欠落する問題を修正しました。	
ターゲットとしての Amazon S3	ターゲットとしての S3 で、 が true で が「」の場合、結果の CSV ファイルヘッダー AddColumnName が 1 列ずれる問題を修正 TimestampColumnName しました。	
ソースとしての Amazon S3 – タスクのフルロードフェーズでのメモリ使用動作	S3 をソースとする際に、フルロード DMS タスクが、ターゲットデータベースにテーブル全体がロードされた後にのみ使用済みメモリを解放していた問題を修正しました。	
ターゲットとしての Amazon S3 – テーブルの再ロードオペレーション	S3 をターゲットとする際に、テーブルの再ロードオペレーションで CDC ファイルが生成されない問題を修正しました。	

AWS Database Migration Service 3.4.5 リリースノート

次の表は、 AWS Database Migration Service (AWS DMS) バージョン 3.4.5 で導入された新機能と機能強化を示しています。

新機能または拡張機能	説明
ターゲットとしての Redis 対応	AWS DMS がターゲットとして Redis をサポートするようになりました。を使用すると AWS DMS、ダウンタイムを最小限に抑えながら、 AWS DMS サポートされている任意のソースから Redis データストアにライブデータを移行できるようになりました。AWS DMS ターゲットの詳細については、「」を参照してください「データ移行のターゲット」。
ソースとして MongoDB 4.2 と 4.4 に対応します	AWS DMS が MongoDB 4.2 および 4.4 をソースとしてサポートするようになりました。を使用して AWS DMS、最小限のダウンタイムで、MongoDB 4.2 および 4.4 クラスターから Amazon DocumentDB (MongoDB 互換) を含む AWS DMS サポートされている任意のターゲットにデータを移行できるようになりました。AWS DMS ソースの詳細については、「」を参照してくださいデータの移行のソース。
ソースとして MongoDB を使用する複数のデータベースに対応します	AWS DMS は、MongoDB をソースとして使用して、1 つのタスクで複数のデータベースの移行をサポートするようになりました。を使用して AWS DMS、MongoDB クラスターの複数のデータベースをグループ化し、1 つのデータベース移行タスクを使用して移行できるようになりました。Amazon DocumentDB (MongoDB 互換) を含む、 AWS DMS サポートされている任意のターゲットに最小限のダウンタイムで移行できます。
ソースとして MongoDB または Amazon DocumentDB (MongoDB 互換) を使用する自動セグメンテーションに対応します	AWS DMS は、ソースとして MongoDB または Amazon DocumentDB を使用した自動セグメンテーションをサポートするようになりました。を使用して AWS DMS、MongoDB または DocumentDB クラスターのコレクションを自動的にセグメント化するようにデータベース移行タスクを設定できます。その後、最小限のダウンタイムで、Amazon DocumentDB を含む AWS DMS サ

新機能または拡張機能	説明
	ポートされている任意のターゲットにセグメントを並行して移行できます。
Amazon Redshift フル ロードのパフォーマンス向上	AWS DMS は、全ロード中に Amazon Redshift をターゲットとして使用するときに並列スレッドの使用をサポートするようになりました。マルチスレッドのフルロードタスク設定を活用することで、 AWS DMS サポートされているソースから Amazon Redshift への初期移行のパフォーマンスを向上させることができます。AWS DMS ターゲットの詳細については、「」を参照してください「データ移行のターゲット」。

AWS DMS 3.4.5 で解決された問題は次のとおりです。

- PostgreSQL を並行トランザクションが多いソースとして使用すると、再開後にデータが欠落または重複する可能性がある問題を修正しました。
- pglogical プラグインを有効にして PostgreSQL をソースとして使用している場合データベース移行タスクがエラー[Could not find relation id …] (関係 ID が見つかりませんでした...)で失敗する問題を修正しました。
- PostgreSQL をソースとして使用し、Oracle をターゲットとして使用する場合に、VARCHAR 列が適切にレプリケートされない問題が修正されました。
- PostgreSQL をソースとして使用しているときに、テーブル定義の冒頭列がプライマリキーでない場合に、削除オペレーションが適切にキャプチャされない問題を修正しました。
- MySQL をソースとして使用するとき、特別なメタデータ設定で LOB の更新がデータベース移行タスクで欠落する問題を修正しました。
- MySQL バージョン 8 をソースとして使用する場合、TIMESTAMP 列が DATETIME のフル LOB モードのように扱われる問題が修正されました。。
- MySQL 5.6.4 以降をソースとして使用する際に、NULL DATETIME レコードの解析時にデータベース移行タスクが失敗する問題を修正しました。
- 並列適用を使用してターゲットに Amazon Redshift を使用しているとき、データベース移行タスクが [Thread is exiting] (スレッドが終了しています) のエラー発生後ハングする問題を解決しました。
- バッチ適用 CDC 中にデータベース移行タスクが Amazon Redshift ターゲット エンドポイントと切断されると、データが失われる可能性がある問題を修正しました。

- ターゲットとして Amazon Redshift を使用する場合 ACCEPTINVCHARS を呼び出すことによるフルロードのパフォーマンスが改善されました。
- Amazon Redshift をターゲットとして使用して one-by-one モードから並列適用モードに戻すと、重複レコードがレプリケートされる問題を修正しました。
- データベース移行タスクで Amazon S3 をターゲットとして使用する場合、cannedAclForObjects=bucket_owner_full_control によって Amazon S3 オブジェクトの所有権がバケット所有者に切り替わらない問題を修正しました。
- Oracle をソースとしてadditionalArchivedLogDestId使用する場合、ECA で複数のアーカイブ先をサポート AWS DMS することで改善されました。
- フル LOB モードで LOB カラムを更新中、データベース移行タスクがエラー OCI_INVALID_HANDLE で失敗する問題を修正しました。
- Oracle をソースとして使用している場合、NVARCHAR2 列がCDC 中に正しく移行されない問題が修正されました。
- RDS for SQL Server をソースとして使用するSafeguardPolicyときに を有効にする AWS DMS ことで改善されました。
- RDS 以外の SQL Server ソースを使用する場合、rdsadmin においてデータベース移行タスクがでエラーを報告する問題を修正しました。
- SQL Server をソースとして使用すると、パーティション設定で UUID を主キーとして使用すると、データ検証が失敗する問題を修正しました。
- Db2 LUW をソースとして使用しているとき、必要な LSN がデータベースログに見つからない場合、フルロードと CDC タスクが失敗することがある問題を修正しました。
- MongoDB をソースとして使用する際のカスタム CDC タイムスタンプをサポート AWS DMS することで改善されました。
- MongoDB ドライバが endSessions でエラーになったときに、MongoDB をソースとして使用して停止すると、データベース移行タスクが停止する問題を修正しました。
- DynamoDB をターゲットとして使用する場合、 がプライマリ以外のフィールドを更新 AWS DMS できない問題を修正しました。
- データ検証で、CLOB と NCLOB 列に誤検出の不一致がレポートされる問題を修正しました。
- Oracle をソースとして使用すると、空白のみレコードのデータ検証が失敗する問題を修正しました。
- パーティション テーブルを切り捨てると、データベース移行タスクがクラッシュする問題を修正しました。

- awsdms_apply_exceptions コントロール テーブルの作成時にデータベース移行タスクが失敗する問題を修正しました。。
- MySQL バージョン 8 を使用すると、caching_sha2_password の認証プラグインに対応するように拡張されました。

AWS Database Migration Service 3.4.4 リリースノート

次の表に、 AWS DMS バージョン 3.4.4 で導入された新機能と機能強化を示します。

新機能または拡張機能	説明
ターゲットとして Kafka を使用した TLS 暗号化と TLS または SASL 認証に対応します	AWS DMS は、Amazon MSK とオンプレミスの Kafka クラスターをターゲットとして使用して、TLS 暗号化と TLS または SASL 認証をサポートするようになりました。Kafka エンドポイントでの暗号化および認証の使用の詳細については、「Transport Layer Security (TLS) を使用した Kafka への接続」をご参照ください。

AWS DMS 3.4.4 で解決された問題は次のとおりです。

- Oracle エンドポイントを使用する場合のタスク失敗の AWS DMS ログ記録が改善されました。
- Oracle Data Guard のフェイルオーバー後に Oracle ソースエンドポイントがロールを切り替えると、 AWS DMS タスクの実行が処理を続行します。
- エラー処理の改善により、ORA—12561 は、Oracle エンドポイントの使用時に回復可能なエラーとして扱われます。
- Oracle をのソースとして使用する場合、EMPTY_BLOB( ) と EMPTY_CLOB( ) カラムが null として移行される問題を修正しました。
- SQL Server をソースとして使用する場合、列 DDL の変更を追加した後、 AWS DMS タスクがレコードを更新できない問題を修正しました。
- TIMESTAMP WITH TIME ZONE データ型に対応してPostgreSQL をソース移行として改善しました。
- PostgreSQL をターゲットとして使用している場合、全ロード中は afterConnectScript の設定が機能しない問題を修正しました。
- 新しい mapUnboundedNumericAsString 設定を導入し、PostgreSQL エンドポイントを使用する場合、精度とスケールを持たない NUMERIC の日付型をより適切に処理できるようにしました。

- PostgreSQL をソースとして使用する場合、 AWS DMS タスクを停止して再開した後、「影響を受ける行は 0 行」でタスクが失敗する問題を修正しました。
- PostgreSQL をソースとして使用する場合、 がBCサフィックス が付いたTIMESTAMPデータ型を移行 AWS DMS できない問題を修正しました。
- PostgreSQL をソースとして使用する場合、 がTIMESTAMP値「"infinity」を移行 AWS DMS できない問題を修正しました。
- 他の値に設定された csvNullValue 設定で S3 をソースとして使用する場合、空の文字列が NULL として処理される問題を修正しました。
- S3 をターゲットとして使用する場合、CDC で全ロードにより timestampColumnName の追加接続属性を CDC の間にソートできるように改善しました。
- S3 をソースとして使用する場合、BYTE、BINARY、BLOBといった 16 進形式のバイナリデータ型の処理が改善されました。
- S3 をターゲットとして使用するときに、削除されたレコードが特殊文字で移行される問題を修正しました。
- Amazon DocumentDB (MongoDB との互換性) をターゲットとして使用するときに、空のキーバリューを処理する問題を修正しました。
- MongoDB NumberDecimalまたは Amazon DocumentDB (MongoDB 互換) をソースとして使用する場合、 が または Decimal128列 MongoDB をレプリケート AWS DMS できない問題を修正しました。 MongoDB
- ソースとして MongoDB または Amazon DocumentDB (MongoDB との互換性) にフェイルオーバーしたときに CDC タスクの再試行を許可する問題を修正しました。
- Kinesis、Kafka、または をターゲット OpenSearch として使用するときに、RAWデータ型値に 16 進数の「0x」プレフィックスを削除するオプションを追加しました。
- ソースとして Db2 LUW を使用すると、固定長の文字列での検証が失敗する問題を修正しました。
- ソースデータ型またはターゲットデータ型のみが FLOAT または DOUBLE の場合、検証が失敗する問題を修正しました。
- ソースとして Oracle をの使用する場合、NULL 文字に検証が失敗する問題を修正しました
- ソースとして Oracle を使用すると、XML 列での検証が失敗する問題を修正しました。
- MySQL をソースとして使用する複合キーに NULL 可能な列がある場合に AWS DMS タスクがクラッシュする問題を修正しました。
- が SQL Server ソースエンドポイントのUNIQUEIDENTIFIER列と PostgreSQL ターゲットエンドポイントの UUID 列の両方を検証 AWS DMS できない問題を修正しました。

- CDC タスクが変更された後に、更新されたソーステーブル定義を使用しなくなる問題を修正しました。
- 無効なユーザー名またはパスワードによって発生したタスクの失敗を回復可能なエラーとして扱うように AWS DMS フェイルオーバーが改善されました。
- RDS for SQL Server をソースとして使用する場合、LSNs が欠落しているため、 AWS DMS タスクが失敗する問題を修正しました。

AWS Database Migration Service 3.4.3 リリースノート

次の表に、 AWS DMS バージョン 3.4.3 で導入された新機能と機能強化を示します。

新機能または拡張機能	説明
Amazon DocumentDB の二ューバージョン	Amazon DocumentDB バージョン 4.0 がソースとしてサポートされるようになりました。
MariaDB の二ューバージョン	MariaDB バージョン 10.4 はソースおよびターゲットとしてサポートされるようになりました。
AWS Secrets Manager 統合のサポート	サポートされているエンドポイントのデータベース接続の詳細(ユーザー認証情報)を AWS Secrets Managerに安全に保存することができます。その後、エンドポイントを作成または変更 AWS DMS するときに、プレーンテキストの認証情報の代わりに対応するシークレットを に送信できます。 AWS DMS その後、 はシークレットを使用してエンドポイントデータベースに接続します。 AWS DMS エンドポイントのシークレットの作成の詳細については、「」を参照してくださいシークレットを使用して AWS Database Migration Service エンドポイントにアクセスするには。
C5 および R5 レプリケーションインスタンスのオプションが増えました	これで、次のように従来より大きなレプリケーションインスタンスサイズを作成できます：C5 のサイズは最大 96 vCPUs と 192 GiB のメモリ、R5 サイズは最大 96 vCPUs および 768 GiB のメモリ。
Amazon Redshift のパフォーマンスの向上	AWS DMS は、Redshift をターゲットとして使用して継続的なレプリケーションのパフォーマンスを向上させるときに、並列適用をサポートするようになりました。詳細については、「Amazon Redshift のマルチスレッドタスク設定」を参照してください。

AWS DMS 3.4.3 で解決された問題は次のとおりです。

- ソースとして Db2 LUW を使用すると、遅延イベントのコミットタイムスタンプが「1970-01-01 00:00:00」になる問題を修正しました。
- SQL Server をフル LOB モードのソースとして使用する場合、 NVARCHAR列をプライマリキーとして AWS DMS タスクが失敗する問題を修正しました。
- ソースとして SQL Server を使用すると、キャッシュされた変更フェーズ中にレコードが欠落する問題を修正しました。
- RDS for SQL Server をソースとして使用する場合、 AWS DMS タスクが再開された後にレコードがスキップされる問題を修正しました。
- ASSERTION ログ記録コンポーネントが SQL Server AWS DMS の大きなログを生成する問題を修正しました。
- ソースとしてMySQL を使用すると、列の解析オーバーフローが原因で CDC フェーズ中にデータの検証が失敗する問題を修正しました。
- PostgreSQL をターゲットとして使用する場合、データ検証中にセグメンテーション障害が原因で AWS DMS タスクがクラッシュする問題を修正しました。
- ソースおよびターゲットとして PostgreSQL を使用するときに CDC 中に DOUBLE データ型でデータの検証が失敗する問題を修正しました。
- ソースとしてPostgreSQL を使用し、ターゲットとしてRedshift を使用すると、コピーコマンドによって挿入されたレコードのレプリケーションが正しくなくなる問題を修正しました。
- ソースとしてPostgreSQL を使用するときに、キャッシュされた変更フェーズ中のデータ損失の問題を修正しました。
- ソースとしてPostgreSQL を使用すると、データの損失やレコードの重複が発生する可能性がある問題を修正しました。
- ソースとしてPostgreSQL を使用しているときに、ケースが混在するスキーマが pglogical を使用して移行できない問題を修正しました。
- ソースとしてOracle を使用すると、最後の失敗メッセージに ORA エラーが含まれない問題を修正しました。
- Oracle をターゲットとして使用する場合、 AWS DMS タスクが UPDATE ステートメントの構築に失敗する問題を修正しました。
- Oracle 12.2 を ASM および Pluggable Database 設定でソースとして使用する場合、 AWS DMS タスクがデータをレプリケートしない問題を修正しました。
- ソースとしてS3 を使用する場合、RFC 4180 に準拠するように引用符を保持し、レコードの解析を改善しました。

- Full Load の列が CDC からの列でソート可能になるよう timestampColumnName の処理が改善されました。
- 新しいエンドポイント設定 を導入することでMessageMaxBytes、1MB を超える LOB 要素がある場合に AWS DMS タスクが失敗する問題を修正しました。
- Redshift をターゲットとして使用する場合、セグメンテーション障害が原因で AWS DMS タスクがクラッシュする問題を修正しました。
- Redshift テスト接続のエラーログを改良しました。
- フルロード中に がすべてのドキュメントを MongoDB から DocumentDB に転送しない問題を修正 AWS DMS しました。
- テーブルマッピングルールにテーブルが含まれていない場合に、 AWS DMS タスクが致命的なエラーを報告する問題を修正しました。
- AWS DMS タスクの再起動前に作成されたスキーマとテーブルが MySQL をソースとして使用している場合ターゲットへのレプリケーションがされなかった問題を修正しました。
- MySQL をソースとして使用するときに、除外ルールでワイルドカードエスケープ [_] がワイルドカード「_」をエスケープできない問題を修正しました。
- MySQL をソースとして使用しているときに、データ型 UNSIGNED BIGINT の列が正しくレプリケートされない問題を修正しました。

AWS Database Migration Service 3.4.2 リリースノート

次の表に、 AWS DMS バージョン 3.4.2. で導入された新機能と機能強化を示します。

新機能または拡張機能	説明
インターネットゲートウェイ、NAT デバイス、VPN 接続、または 接続を必要とせずに、Amazon Virtual Private Cloud (Amazon VPC) を AWS Database Migration Service (DMS) にプライベート AWS Direct Connect に接続するためのサポート。	作成した VPC インターフェイスエンドポイントを介して、 AWS DMS Amazon VPC に接続してアクセスできるようになりました。このインターフェイスエンドポイントを使用すると、Amazon ネットワークインフラストラクチャ内の AWS DMS レプリケーションインスタンスのすべてのネットワークアクティビティを分離できます。 AWS CLI または SDK AWS DMS を使用して へのすべての API コールにこのインターフェイスエンドポイントへの参照を含めることで、すべての AWS DMS アクティビティがパブリックインターネットに表示されなくなります。詳細については、「AWS

新機能または拡張機能	説明
	Database Migration Service でのインフラストラクチャのセキュリティ」を参照してください。 **Note** この機能は、サポートされているすべての AWS DMS エンジンバージョンで使用できます。
ターゲットとして Amazon S3 を使用する CDC 日付ベースフォルダーのパーティショニング	AWS DMS は、S3 をターゲットとして使用してデータをレプリケートするときに、日付ベースのフォルダパーティショニングをサポートするようになりました。詳細については、「日付ベースのフォルダパーティション分割を使用する」を参照してください。

AWS DMS 3.4.2 で解決された問題は次のとおりです。

- ターゲットとして Redshift を使用して移行を実行する場合の STATUPDATE オプションが追加されました。。
- 新しい設定が導入され、検証タスクが改善されました。ValidQueryCdcDelaySecond は、ソース エンドポイントとターゲット エンドポイントの両方で最初の検証クエリを遅延させ、移行レイテンシーが高い場合にリソースの競合を減らします。
- 検証タスクの開始に長い時間がかかる問題を修正 AWS DMS しました。
- S3 をターゲットとして使用してレプリケーション タスクを開始または停止するときに、空のレコードが生成される問題を修正しました。
- フルロードの完了後にタスクが停止する問題を修正しました。
- S3 をソースとして使用しているときに、ソーステーブルにデータエラーが発生するとタスクが停止する問題を修正しました。
- ソース エンドポイントのユーザーアカウントが無効になっているとき開始中にタスクが停止する問題を修正しました。
- REPLICA IDENTITY FULL でソースとして PostgreSQL を使用するとタスクがクラッシュする問題を修正しました。
- pglogical プラグインでソースとして PostgreSQL を使用すると、タスクがトランザクションに失敗する問題を修正しました。

- Redshift をターゲットとして使用する場合、 が圧縮されたソースファイルを削除しなかった問題を修正 AWS DMS しました。
- MySQL をデータ型 BIGINT UNSIGNED のソースとターゲットの両方として使用すると、検証タスクが偽陰性を報告する問題を修正しました。
- CHAR タイプとしてプライマリ キー列でソースとして SQL Server を使用すると、検証タスクで誤検出が報告される問題を修正しました。
- を使用して S3 をターゲットとして使用start-replicationしてレプリケーションタスクを開始するときに、 AWS DMS がターゲットオブジェクトをクリアしない問題を修正しました。
- ソースとしてDb2 を使用すると、データ検証に関して生じたいくつかの問題を修正しました。
- VARCHAR 列を主キーにし、ソースとしてSQL Server を使用すると、検証タスクが停止する問題を修正しました。
- ソースとしてPostgreSQL を使用している場合、タイムゾーンのデータ型 TIMESTAMP に対応するようになりました。

AWS Database Migration Service 3.4.1 ベータリリースノート

次の表に、 AWS DMS バージョン 3.4.1 ベータで導入された新機能と機能強化を示します。

新機能または拡張機能	説明
MongoDB 二ューバージョン	ソースとして MongoDB バージョン 4.0 に対応するようになりました。
SQL Server の TLS 1.2 対応	AWS DMS で SQL Server エンドポイントの TLS 1.2 がサポートされるようになりました。

AWS DMS 3.4.1 Beta で解決された問題は次のとおりです。

- Oracle 19c TDEのサポートが改善されました。
- ターゲットとしてRedshift を使用した utf8mb4 文字セットとアイデンティティデータ型のサポートが改善されました。
- ソースとしてMySQL を使用し、バイナリログが存在しない場合のレプリケーション タスクの失敗処理が改善されました。
- さまざまなデータ型と文字セットでのデータ検証のサポートが改善されました。

- ターゲットとして Kinesis と Kafka を使用する場合、IncludeNullAndEmpty を設定する新しいエンドポイントによるヌル値の処理が改善されましたののこと。
- ターゲットとしてKafka を使用する場合のエラー ログと処理が改善されました。
- ソースとして SQL Server を使用する場合の DST タイムオフセットが改善されました。
- レプリケーション タスクがターゲットとして Oracle の既存のテーブルを作成しようとする問題を修正しました。
- ソースとして Oracle を使用すると、データベース接続の強制終了後、レプリケーション タスクが停止する問題を修正しました。
- AlwaysOn 設定でソースとして SQL Server を使用しているとき、レプリケーション タスクが新しいプライマリを検出して再接続できない問題を修正しました。
- ターゲットとして S3 をする特定の条件の下でレプリケーション タスクが "D" を "OP" に追加できない問題を修正しました。

AWS Database Migration Service 3.4.0 ベータリリースノート

次の表に、 AWS DMS バージョン 3.4.0 で導入された新機能と機能強化を示します。

新機能または拡張機能	説明
MySQL の二ューバージョン	AWS DMS は、トランザクションペイロードが圧縮されている場合を除き、ソースとして MySQL バージョン 8.0 をサポートするようになりました。
MTLS 1.2 による MySQL 対応	AWS DMS が MySQL エンドポイントの TLS 1.2 をサポートするようになりました。
MariaDB の二ューバージョン	AWS DMS で MariaDB バージョン 10.3.13 がソースとしてサポートされるようになりました。
セルフマネージド Microsoft SQL Server ソースSysAdmin へのアクセス不可	AWS DMS は、非SysAdmin ユーザーによるオンプレミスおよびEC2-hostedソースエンドポイントへのアクセスをサポートするようになりました。

新機能または拡張機能	説明
	Note 現在、この機能はベータモードです。試してみたい場合は、 AWS サポートにお問い合わせください。
CREATE TABLE AS を使用して作成された CDC タスクおよび Oracle ソーステーブル	AWS DMS は、 CREATE TABLE ASステートメントを使用して作成された Oracle ソーステーブルに対して実行されるフルロードタスク、CDC タスク、CDC のみのタスクの両方をサポートするようになりました。

AWS DMS 3.4.0 で解決された問題は次のとおりです。

- 移行前のタスク評価が改善されました。詳細については、「タスクの移行前評価の有効化と操作」を参照してください。
- 浮動小数点データ型、実数、倍精度浮動小数点データ型のデータ検証が改善されました。
- [The specified key does not exist] (指定されたキーは存在しません) のエラーをより適切に処理することで、ターゲットとしての Amazon Redshift を改善しました。
- ParallelApplyQueuesPerThreadAmazon OpenSearch Service (OpenSearch Service) をターゲットとして、ParallelApplyThreads、ParallelApplyBufferSize、 などのマルチスレッド CDC ロードタスク設定をサポートします。
- 複合プライマリキーの使用をサポートすることで、ターゲットとしての OpenSearch サービスを改善しました。
- ソースとしてPostgreSQL を使用し、パスワードに特殊文字が含まれているとき、テスト接続が失敗する問題を修正しました。
- 一部の VARCHAR 列を切り詰めると SQL Server をソースとして使用する際の問題が修正されました。
- Amazon RDS SQL Server をソースとして使用する場合、 AWS DMS がオープントランザクションを閉じない問題を修正しました。この際ポーリング間隔パラメータが正しく設定されていないと、データが失われる可能性があります。推奨されるポーリング間隔値を設定する方法の詳細については、「のソースとしての Microsoft SQL Server データベースの使用 AWS DMS」をご参照ください。

- ソースとしての Oracle Standby ではバイナリリーダーの使用時に CDC タスクが予期せず停止する問題を修正しました。
- IBM DB2 for LUW で、「数値リテラル 0 は値が範囲外であるため有効ではありません」というメッセージでタスクが失敗する問題を修正しました。
- PostgreSQL ソースに新しい列が追加され、カラムがソースで最初に作成されたデータ型とは異なるデータ型で作成された PostgreSQL から PostgreSQL への移行に関する問題を修正しました。
- MySQL ソースで、バイナリログを取得できなかったときに移行タスクが予期せず停止する問題を修正しました。
- `BatchApply` が使用されていたとき Oracle ターゲットに関連する問題が修正されました。
- MySQL と MariaDB での `TIME` データ型移行時の問題を修正しました。
- IBM DB2 LUW ソース で、テーブルにプライマリキーまたは一意キーがない場合、LOB を含むテーブルの移行が失敗する問題を修正しました。

AWS Database Migration Service 3.3.4 リリースノート

AWS DMS 3.3.4 で解決された問題は次のとおりです。

- ソースとして PostgreSQL を使用すると、トランザクションがドロップまたは複製される問題を修正しました。
- スキーマ名でのドル記号 ($) 使用への対応が改善されました。
- ソースとしてRDS SQL Server を使用すると、レプリケーション インスタンスが開いているトランザクションを閉じない問題を修正しました。
- ソースとしてPostgreSQL を使用し、パスワードに特殊文字が含まれているとき、テスト接続が失敗する問題を修正しました。
- [The specified key does not exist] (指定されたキーは存在しません) のエラーをより適切に処理することで、ターゲットとしての Amazon Redshift を改善しました。
- さまざまなデータ型と文字セットでのデータ検証のサポートが改善されました。
- レプリケーション タスクがターゲットとして Oracle の既存のテーブルを作成しようとする問題を修正しました。
- レプリケーション タスクで、ターゲットとして Amazon S3 の特定の条件下で `"D"` が `"0P"` 列に追加されない問題を修正しました。

AWS Database Migration Service 3.3.3 リリースノート

次の表に、 AWS DMS バージョン 3.3.3 で導入された新機能と機能強化を示します。

新機能または拡張機能	説明
新しい PostgreSQL バージョン	PostgreSQL バージョン 12 がソースおよびターゲットとしてサポートされるようになりました。
Amazon OpenSearch Service をターゲットとする複合プライマリキーのサポート	AWS DMS 3.3.3 以降、複合プライマリキーの使用は OpenSearch サービスターゲットでサポートされています。
Oracle 拡張データ型のサポート	Oracle ソースとターゲットの両方に対して、Oracle 拡張データ型がサポートされました。
アカウントあたりの AWS DMS リソース数の増加	作成できる AWS DMS リソース数の制限が増加しました。詳細については、「AWS Database Migration Service のクォータ」を参照してください。

AWS DMS 3.3.3 で解決された問題は次のとおりです。

- Amazon Kinesis の Parallel Apply に特定の update ステートメントを使用するとタスクがクラッシュする問題を修正しました。
- Amazon S3　をターゲットとした ALTER TABLE ステートメントでタスクがクラッシュする問題を修正しました。
- Microsoft SQL Server をソースとして使用すると、ポリゴンの列の値が切り捨てられる問題を修正しました。
- Oracle をソースとして使用している場合の、JA16SJISTILDE と JA16EUCTILDE の Unicode コンバーターでの問題を修正しました。
- MEDIUMTEXT 列と LONGTEXT 列を MySQL から S3 カンマ区切り値 (CSV) 形式に移行できなかった問題を修正しました。
- Apache Parquet 出力でブール値の列が正しくない型に変換される問題を修正しました。
- Oracle の拡張 varchar 列に関する問題を修正しました。
- 特定のタイムスタンプの組み合わせが原因でデータ検証タスクが失敗する問題を修正しました。

- Sybase データ定義言語 (DDL) レプリケーションの問題を修正しました。
- Oracle Binary Reader での Oracle Real Application Clusters (RAC) ソースのクラッシュに関連する問題を修正しました。
- スキーマ名に大文字と小文字が使用されている Oracle ターゲットの検証に関する問題を修正しました。
- IBM Db2 バージョン 9.7 および 10 の検証に関する問題を修正しました。
- `StopTaskCachedChangesApplied` と `StopTaskCachedChangesNotApplied` が有効になっていると、タスクが 2 回停止しない問題を修正しました。

ドキュメント履歴

2018 年 1 月の後に AWS Database Migration Service ユーザーガイドに対して行われた重要な変更を次の表にまとめました。

RSS フィードにサブスクライブして、このドキュメントの更新に関する通知を受信できます。AWS DMS のバージョンリリースの詳細については、「AWS DMS リリースノート」を参照してください。

変更	説明	日付
AWS DMS でターゲットとしての RDS IBM DB2 のサポートを追加	AWS DMS は、Amazon RDS IBM DB2 をターゲットとしてサポートするようになりました。	2023 年 12 月 4 日
AWS DMS でターゲットとしての Timestream のサポートを追加	AWS DMS は、Timestream をターゲットとしてサポートするようになりました。	2023 年 11 月 17 日
AWS DMS で Redshift ターゲットデータ検証のサポートを追加	AWS DMS は、Redshift ターゲットのデータの検証をサポートするようになりました。	2023 年 11 月 14 日
4 つの新しいエンドポイントタイプのサポートが AWS DMS に追加されました	AWS DMS は、Microsoft Azure Database for PostgreSQL、Microsoft Azure Database for MySQL、OCI MySQL Heatwave、Google Cloud for PostgreSQL のソースとしての使用をサポートするようになりました。	2023 年 10 月 26 日
AWS DMS は、AWS サービスにリンクされたロールをサポートするようになりました。	AWS DMS は、AWS サービスにリンクされたロール `AWSServiceRoleForDMSServerless` をサポー	2023 年 5 月 22 日

	トするようになりました。これにより AWS DMS は、ユーザーに代わってリソースを作成および管理できるようになりました (メトリクスデータポイントを Amazon CloudWatch にパブリッシュするなど)。	
AWS DMS に、新しい AWS 管理ポリシーが追加されました。	AWS DMS は、サーバーレスレプリケーションログを CloudWatch Logs に公開できるようにする AWS 管理ポリシーをサポートするようになりました。	2023 年 5 月 22 日
AWS DMS に、新しい AWS 管理ポリシーが追加されました。	AWS DMS は、Amazon CloudWatch へのメトリクスデータポイントの公開を許可する AWS 管理ポリシーをサポートするようになりました。また AWS DMSは、AWS 管理ポリシーの変更の追跡を開始しました。	2023 年 3 月 6 日
VPC ソースエンドポイントとターゲットエンドポイントのサポート	AWS DMS は、ソースとターゲットとして仮想プライベートクラウド (VPC) のエンドポイントをサポートするようになりました。AWS DMS は、サービスへの明示的に定義されたルートが AWS DMS VPC で定義されている場合、VPC エンドポイントを使用して任意の AWS サービスに接続できるようになりました。	2022 年 6 月 30 日

ソースとしての SQL Server リードレプリカのサポート	AWS DMS は、SQL Server リードレプリカをソースとしてサポートするようになりました。AWS DMS を使用すると、SQL Server リードレプリカから AWS DMS がサポートしている任意のターゲットへのライブ移行を実行できるようになりました。	2022 年 6 月 30 日
IBM Db2 z/OS データベースをフルロードのみソースとしてサポート	AWS DMS は、IBM Db2 z/OS データベースをソースとしてサポートするようになりました。AWS DMS を使用すると、Db2 メインフレームから AWS DMS がサポートしている任意のターゲットへのライブ移行を実行できるようになりました。	2022 年 6 月 30 日
EventBridge DMS イベントのサポート	AWS DMS は、DMS イベントの EventBridge を使用したイベントサブスクリプションの管理をサポートします。	2022 年 6 月 30 日
ターゲットとしての Babelfish のサポート	AWS DMS は、Babelfish をターゲットとしてサポートするようになりました。AWS DMS を使用して、ダウンタイムを最小限に抑えならが、任意のサポートされている AWS DMS ソースからライブデータを Babelfish に移行できるようになりました。	2022 年 6 月 30 日

ターゲットとしての Aurora Serverless v2 のサポート	AWS DMS は、Aurora Serverless v2 をターゲットとしてサポートするようになりました。AWS DMS を使用して、Aurora Serverless v2 へのライブ移行を実行できるようになりました。	2022 年 6 月 30 日
開始方法のチュートリアル	AWS DMS の入門チュートリアルのアップデート。このチュートリアルでは、ソースとして MySQL データベース、ターゲットとして PostgreSQL データベースを使用します。	2021 年 5 月 20 日
ターゲットとして Amazon Neptune に対応	データ移行のターゲットとして Amazon Neptune のサポートが追加されました。	2020 年 6 月 1 日
ターゲットとしての Apache Kafka のサポート	データ移行のターゲットとしての Apache Kafka のサポートが追加されました。	2020 年 3 月 20 日
セキュリティコンテンツの更新	お客様のリクエストに応えて、セキュリティコンテンツを更新および標準化しました。	2019 年 12 月 20 日
AWS Snowball Edge を使用した移行	AWS Snowball Edge を使用した大規模なデータベースの移行のサポートを追加しました。	2019 年 1 月 24 日
ターゲットとして Amazon DocumentDB (MongoDB 互換) に対応	データ移行のターゲットとして　Amazon DocumentDB (MongoDB 互換) に対応	2019 年 1 月 9 日

ターゲットとしての Amazon OpenSearch Service と Amazon Kinesis Data Streams のサポート	データ移行のターゲットとして OpenSearch Service と Kinesis Data Streams のサポートが追加されました。	2018 年 11 月 15 日
CDC ネイティブ開始サポート	変更データキャプチャ (CDC) の使用時におけるネイティブの開始ポイントのサポートを追加しました。	2018 年 6 月 28 日
Db2 LUW のサポート	データ移行のソースとしての IBM Db2 LUW のサポートを追加しました。	2018 年 4 月 26 日
ターゲットとしての SQL Server のサポート	ソースとして Amazon RDS for Microsoft SQL Server のサポートを追加しました。	2018 年 2 月 6 日

AWS 用語集

AWS の最新の用語については、「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。