aws

デベロッパーガイド

Amazon MemoryDB

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

Amazon MemoryDB: デベロッパーガイド

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

Amazon の商標およびトレードドレスは Amazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

MemoryDB とは 1
MemoryDB の機能 1
MemoryDB コアコンポーネント 2
クラスター 3
ノード 4
シャード 5
パラメータグループ 5
サブネットグループ 5
アクセスコントロールリスト 6
Users 6
関連サービス 6
リージョンとアベイラビリティーゾーンの選択 7
ノードの配置 8
サポートされているリージョンおよびエンドポイント 9
MemoryDB にアクセスする 12
MemoryDB セキュリティ 13
MemoryDB の開始方法 15
ステップ 1: セットアップ 15
AWS アカウント へのサインアップ 15
管理アクセスを持つユーザーを作成する 16
プログラム的なアクセス権を付与する 17
アクセス許可を設定する (新規の MemoryDB ユーザーのみ) 19
AWS CLI のダウンロードと設定 20
ステップ 2: クラスターを作成する 21
MemoryDB クラスターの作成 21
認証のセットアップ 32
ステップ 3: クラスターへのアクセスの許可 33
ステップ 4: クラスターに接続する 35
クラスターエンドポイントを見つける 35
メモリ DB クラスターへの接続 (Linux) 35
ステップ 5: クラスターを削除する 37
次のステップ 39
ノードの管理 41
MemoryDB のノードとシャード 41

サポートされているノードの種類 43
リザーブドノード 45
リザーブドノードの概要 45
提供タイプ 46
サイズ柔軟なリザーブドノード 46
ノードを Redis OSS から Valkey にアップグレードする 48
リザーブドノードの削除 49
リザーブドノードの操作 49
ノードの置換 57
クラスターの管理 60
データ階層化 61
ベストプラクティス 62
データ階層化の制限 62
データ階層化の料金 63
データ階層化のモニタリング 63
データ階層化の使用 63
スナップショットからクラスターにデータを復元する 65
クラスターを準備する 66
要件の特定 67
クラスターの作成 70
クラスターの詳細を表示する 71
クラスターの変更 76
Redis OSS から Valkey へのクロスエンジンアップグレードをトリガーする方法 78
クラスターからのノードの追加/削除 80
クラスターへのアクセス 82
すべてのクラスターに対するアクセスを許可する 82
外部 AWS から MemoryDB にアクセスする 84
接続エンドポイントの検索 90
シャード 93
シャードの名前を見つける 94
MemoryDB の実装を管理する 98
エンジンバージョン 98
MemoryDB 7.3 99
Valkey 7.2.6 99
Redis OSS 7.0 (拡張) 100
Redis OSS 7.0 (拡張) 101

Redis OSS 6.2 (拡張) 102
エンジンバージョンのアップグレード 102
JSON の使用開始 105
JSON データ型の概要 106
サポートされているコマンド 118
MemoryDB リソースのタグ付け 159
タグによるコストのモニタリング 165
AWS CLI を使用したタグの管理 166
MemoryDB API を使用したタグの管理 170
メンテナンスの管理 172
ベストプラクティス 174
レジリエンス 175
ベストプラクティス: Pub/Sub および拡張 I/O マルチプレクシング 177
ベストプラクティス: オンラインクラスターのサイズ変更 177
MemoryDB レプリケーションを理解する 178
整合性 179
クラスター内のレプリケーション 179
マルチ AZ によるダウンタイムの最小化 180
レプリカの数の変更 188
スナップショットおよび復元 198
制約 199
コスト 199
自動スナップショットのスケジュール 200
手動スナップショットの作成 201
最終スナップショットの作成 204
スナップショットの説明 206
スナップショットのコピー 209
のスナップショットをエクスポートする 212
スナップショットからの復元 222
スナップショットのクラスターのシード 228
スナップショットのタグ付け 234
スナップショットの削除 235
Scaling (スケーリング) 236
MemoryDB クラスターのスケーリング 238
パラメータグループを使用したエンジンパラメータの設定 260
パラメータの管理 262

パラメータグループの階層 ... 263
パラメータグループを作成する ... 264
パラメータグループを名前別に一覧表示する ... 268
パラメータグループの値を一覧表示する ... 273
パラメータグループを変更する ... 274
パラメータグループを削除する ... 277
エンジン固有のパラメータ ... 279
制限されるコマンド ... 296
チュートリアル: Amazon VPC の MemoryDB にアクセスする Lambda 関数の設定 ... 297
ステップ 1: クラスターを作成する ... 297
ステップ 2: Lambda 関数を作成する ... 300
ステップ 3: Lambda 関数をテストする ... 304
ステップ 4: クリーンアップする (オプション) ... 304
ベクトル検索 ... 306
ベクトル検索の概要 ... 306
インデックスとキースペース ... 307
インデックスフィールドの型 ... 308
ベクトルインデックスアルゴリズム ... 309
ベクトル検索のクエリ式 ... 310
INFO コマンド ... 312
ベクトル検索のセキュリティ ... 315
ユースケース ... 316
取得拡張生成 (RAG) ... 316
耐久性のあるセマンティックキャッシュ ... 316
不正検出 ... 317
その他のユースケース ... 318
ベクター検索の機能と制限 ... 318
ベクトル検索が利用可能なリージョン ... 318
パラメトリック制限 ... 319
[Scaling limits] (スケーリング履歴) ... 319
オペレーションの制限 ... 320
スナップショットのインポート/エクスポートとライブ移行 ... 320
メモリ消費 ... 320
バックフィル中のメモリ不足 ... 324
トランザクション ... 324
ベクトル検索が有効になっているクラスターを作成する ... 324

の使用AWS マネジメントコンソール 324
の使用AWS Command Line Interface 325
ベクトル検索コマンド 326
FT.CREATE 326
FT.SEARCH 330
FT.AGGREGATE 333
FT.DROPINDEX 334
FT.INFO 335
FT._LIST 337
FT.ALIASADD 338
FT.ALIASDEL 338
FT.ALIASUPDATE 338
FT._ALIASLIST 339
FT.PROFILE 339
FT.EXPLAIN 339
FT.EXPLAINCLI 340
MemoryDB マルチリージョン 341
前提条件と制限 342
仕組み 344
整合性と競合の解決 345
CRDT と例 347
コンソールでの MemoryDB マルチリージョンの使用 350
MemoryDB マルチリージョンで新しいクラスターを作成する 350
スナップショットをマルチリージョンクラスター内の新規または既存のクラスターに復元する 352
MemoryDB マルチリージョンでクラスターを変更する 355
MemoryDB マルチリージョンでクラスターを削除する 358
CLI での MemoryDB マルチリージョンの使用 361
MemoryDBMulti リージョンでのクラスターの作成 361
マルチリージョンクラスターを更新する 362
MemoryDB クラスターのスケーリング 363
MemoryDB マルチリージョンでのクラスターの削除 362
MemoryDB マルチリージョンのモニタリング 363
MemoryDB マルチリージョンでのスケーリング 364
サポートされているコマンドとサポートされていないコマンド 366
セキュリティ 370

データ保護 ... 371
MemoryDB 内のデータのセキュリティ ... 372
保管時の暗号化 ... 373
転送時の暗号化 (TLS) ... 375
ACLs を使用したユーザーの認証 ... 376
IAM を使用した認証 ... 391
Identity and Access Management ... 398
対象者 ... 399
アイデンティティによる認証 ... 399
ポリシーを使用したアクセス権の管理 ... 401
MemoryDB と IAM の連携の仕組み ... 403
アイデンティティベースのポリシーの例 ... 411
トラブルシューティング ... 414
アクセスコントロール ... 416
アクセス管理の概要 ... 418
ログ記録とモニタリング ... 449
CloudWatch によるモニタリング ... 450
イベントのモニタリング ... 470
AWS CloudTrail で MemoryDB API 呼び出しをログに記録する ... 482
コンプライアンス検証 ... 489
インフラストラクチャセキュリティ ... 490
ネットワーク間のトラフィックのプライバシー ... 490
MemoryDB と Amazon VPC ... 491
サブネットおよびサブネットグループ ... 502
MemoryDB API とインターフェイス VPC エンドポイント (AWS PrivateLink) ... 518
セキュリティの脆弱性への対処 ... 521
サービスの更新 ... 523
サービスの更新の管理 ... 523
サービスの更新の適用 ... 529
の使用AWS CLI ... 530
参照資料 ... 532
MemoryDB API の使用 ... 533
クエリ API を使用する ... 533
利用可能なライブラリ ... 536
アプリケーションのトラブルシューティング ... 537
クォータ ... 539

ドキュメント履歴 541

MemoryDB とは

MemoryDB は、超高速パフォーマンスを実現する、耐久性に優れたインメモリデータベースサービスです。マイクロサービスアーキテクチャを備えた最新のアプリケーション専用に構築されています。

MemoryDB は、人気のオープンソースデータストアである Valkey および Redis OSS と互換であり、既に使用されているのと同じ柔軟で使いやすいデータ構造および API、コマンドを使用してアプリケーションを迅速に構築できます。MemoryDB では、すべてのデータがメモリに保存されるため、読み取り (マイクロ秒)、書き込みレイテンシー (数ミリ秒)、高いスループットを実現できます。また、MemoryDB はマルチ AZ トランザクションログを使用して複数のアベイラビリティーゾーン (AZ) にデータを永続的に保存し、迅速なフェイルオーバー、データベースリカバリ、ノード再起動を可能にします。

メモリ内のパフォーマンスとマルチ AZ の耐久性を兼ね備えた MemoryDB は、マイクロサービスアプリケーションの高性能プライマリデータベースとして使用できるため、キャッシュと耐久性の高いデータベースの両方を個別に管理する必要がありません。

トピック

- MemoryDB の機能
- MemoryDB コアコンポーネント
- 関連サービス
- リージョンとアベイラビリティーゾーンの選択
- MemoryDB にアクセスする
- MemoryDB セキュリティ

MemoryDB の機能

MemoryDB は、超高速パフォーマンスを実現する、耐久性に優れたインメモリデータベースサービスです。MemoryDB には以下の機能が含まれます。

- プライマリノードには強固な一貫性を、レプリカノードには最終的な一貫性を保証します。詳細については、「整合性」を参照してください。
- マイクロ秒単位の読み取りと1桁のミリ秒単位の書き込みレイテンシーで、クラスターあたり最大1億6,000万TPSです。

- 柔軟で使いやすい Valkey および Redis OSS データ構造と API。ほとんど変更を加えずに、新しいアプリケーションの構築や既存の Valkey ベースおよび Redis OSS ベースのアプリケーションの移行を簡単に行うことができます。
- マルチ AZ トランザクションログを使用したデータ耐久性により、データベースの復旧と再起動を迅速に行えます。
- 自動フェイルオーバー、ノード障害の検出と復旧によるマルチ AZ の可用性。
- ノードを追加および削除して、垂直方向にスケールするのは簡単です。ノードタイプを大きくおよび小さいノードタイプに移動して、垂直方向にスケールすることもできます。シャードを追加することで書き込みスループットをスケーリングでき、レプリカを追加することで読み取りスループットをスケーリングできます。
- プライマリノードでは書き込み後の読み取りの一貫性、レプリカノードでは最終的な整合性が保証されます。
- MemoryDB は、転送中の暗号化、保存時の暗号化、および アクセスコントロールリスト (ACL) によるユーザー認証 経由でのユーザー認証をサポートします。
- Amazon S3 での自動スナップショット。保存期間は最大 35 日間です。
- クラスターあたり最大 500 ノードと 100 TB を超えるストレージ (シャードあたり 1 レプリカ) Support。
- TLS による転送中の暗号化と AWS KMS キーによる保存時の暗号化。
- Valkey および Redis OSS アクセスコントロールリスト (ACL) によるユーザー認証 によるユーザー認証と認可。
- AWS Graviton2 インスタンスタイプSupport。
- CloudWatch、Amazon VPC、CloudTrail、Amazon SNS などの他の AWS サービスと統合して、モニタリング、セキュリティ、通知を行います。
- フルマネージド型のソフトウェアパッチ適用とアップグレード。
- AWS Identity and Access Management (IAM) の統合と管理 API のタグベースのアクセス制御。

MemoryDB コアコンポーネント

ここでは、MemoryDB のデプロイメントの主なコンポーネントの概要を確認できます。

トピック

- クラスター
- ノード

- シャード
- パラメータグループ
- サブネットグループ
- アクセスコントロールリスト
- Users

クラスター

クラスターは、単一のデータセットを提供する、1 つ以上のノードの集合です。MemoryDB データセットはシャードに分割され、各シャードには、プライマリノードと最大 5 個のリードノードが含まれます。プライマリノードは読み取りリクエストと書き込みリクエストを処理し、レプリカは読み取りリクエストのみを処理します。プライマリノードはレプリカノードにフェイルオーバーして、そのレプリカをそのシャードの新しいプライマリノードに昇格させることができます。MemoryDB はデータベースエンジンとして Valkey または Redis OSS を実行し、クラスターを作成するときにはクラスターのエンジンバージョンを指定します。クラスターは、AWS CLI、MemoryDB API、または AWS マネジメントコンソール 使用して作成、変更できます。

各 MemoryDB クラスターは、Valkey または Redis OSS エンジンバージョンを実行します。エンジンの各バージョンには、独自のサポート機能があります。さらに、エンジンの各バージョンは、パラメータグループに一連のパラメータを保有し、これにより管理するクラスターの動作を制御します。

クラスターの計算容量とメモリの容量は、クラスターのノードタイプによって決まります。お客様のニーズに最も合うノードの種類を選択できます。ニーズが時間の経過と共に変化する場合は、ノードの種類を変更できます。詳細については、「サポートされているノードの種類」を参照してください。

> **Note**
>
> MemoryDB ノードタイプの料金情報については、「MemoryDB 料金」を参照してください。

Amazon Virtual Private Cloud (Amazon VPC) サービスを使用して、仮想プライベートクラウド (VPC) 上のクラスターを実行できます。VPC を使用する場合、仮想ネットワーキング環境を制御できます。独自の IP アドレスの範囲を選択し、サブネットを作成してルーティングおよびアクセス制御リストを設定できます。MemoryDB は、スナップショット、ソフトウェアパッチ、自動的な障

害検出、および復旧を管理します。VPC でクラスターを実行するために、追加料金はかかりません。MemoryDB で Amazon VPC を使用する方法については、「MemoryDB と Amazon VPC」を参照してください。

クラスターを対象とした多くの MemoryDB オペレーションがあります。

- クラスターの作成
- クラスターの変更
- クラスターのスナップショットの作成
- クラスターの削除
- クラスターのエレメントの表示
- クラスター間で送受信されるコスト配分タグの追加または削除

詳細については、次の関連トピックを参照してください。

- クラスターの管理 および ノードの管理

  クラスター、ノードおよび関連オペレーションに関する情報。
- MemoryDB の耐障害性

  クラスターの耐障害性向上に関する情報。

ノード

ノードは MemoryDB デプロイメントの最小構成要素であり、Amazon EC2 インスタンスを使用して実行されます。各ノードは、クラスター作成時に選択したエンジンバージョンを実行します。ノードはクラスターに属するシャードに属します。

各ノードは、クラスター作成時に選択したバージョンでエンジンのインスタンスを実行します。必要に応じて、クラスター内のノードを別のタイプにスケールアップまたはスケールダウンできます。詳細については、「Scaling (スケーリング) 」を参照してください。

クラスター内の各ノードは同じノードタイプです。複数のタイプのキャッシュノードがサポートされており、それぞれ異なる量のメモリあります。サポートされているノードタイプについては、「サポートされているノードの種類」を参照してください。

ノードの詳細については、「ノードの管理」を参照してください。

シャード

シャードは 1~6 個のノードをグループ化したもので、1 つはプライマリ書き込みノードとして、残りの 5 つはリードレプリカとして機能します。MemoryDB クラスターには常に少なくとも 1 つのシャードがあります。

MemoryDB クラスターには、最大 500 個のシャードがあり、データはシャード間で分割されます。例えば、83 個のシャード (シャードごとに 1 つのプライマリと 5 レプリカ) と 500 個のシャード (プライマリのみでレプリカなし) の範囲で、500 個のノードクラスターを設定できます。増加に対応できる十分な IP アドレスがあることを確認してください。一般的な落とし穴として、サブネットグループ内のサブネットの CIDR 範囲が小さすぎる、またはサブネットが他のクラスターで共有され、頻繁に使用されていることが挙げられます。

複数ノードシャード では、1 つの読み書き可能プライマリノードと 1~5 個のレプリカノードを含めることで、レプリケーションを実装します。詳細については、「MemoryDB レプリケーションを理解する」を参照してください。

シャードの詳細については、「シャードの使用」を参照してください。

パラメータグループ

パラメータグループを使用すると、クラスター上のエンジンのランタイム設定を簡単に管理できます。パラメータは、メモリの使用状況、項目サイズなどを制御するために使用されます。MemoryDB パラメータグループはクラスターに適用可能なエンジン固有パラメータの名前付きコレクションであり、クラスター内のノードはすべてまったく同じ方法で設定されます。

MemoryDB パラメータグループの詳細については、「パラメータグループを使用したエンジンパラメータの設定」を参照してください。

サブネットグループ

サブネットグループ は、Amazon Virtual Private Cloud (VPC) 環境で実行しているクラスターに対して指定できるサブネット (通常はプライベート) の集合です。

Amazon VPC でクラスターを作成する場合、サブネットグループを指定するか、デフォルトで提供されるサブネットグループを使用できます。MemoryDB はそのキャッシュサブネットグループを使用して、そのサブネット内でノードに関連付けるサブネットおよび IP アドレスを選択します。

MemoryDB サブネットグループの詳細については、「サブネットおよびサブネットグループ」を参照してください。

アクセスコントロールリスト

アクセスコントロールリストは、1 人以上のユーザーのコレクションです。アクセス文字列は、「ACL ルール」に従い、Valkey または Redis OSS のコマンドとデータへのユーザーアクセスを許可します。

MemoryDB アクセスコントロールリストの詳細については、「アクセスコントロールリスト (ACL) によるユーザー認証」を参照してください。

Users

ユーザーにはユーザー名とパスワードがあり、MemoryDB クラスターのデータへのアクセスやコマンドの発行に使用されます。ユーザーはアクセスコントロールリスト (ACL) のメンバーであり、これを使用して MemoryDB クラスターでのそのユーザーの権限を決定できます。詳細については、アクセスコントロールリスト (ACL) によるユーザー認証 を参照してください。

関連サービス

ElastiCache

MemoryDB と ElastiCache のどちらを使用するかを決める際には、以下の比較を検討してください。

- MemoryDB は、超高速のプライマリデータベースを必要とするワークロード向けの、耐久性に優れたインメモリデータベースです。ワークロードが超高速のパフォーマンス (マイクロ秒の読み取りと 1 桁ミリ秒の書き込みレイテンシー) を提供する耐久性のあるデータベースを必要とする場合は、MemoryDB の使用を検討する必要があります。また、Valkey または Redis OSS のデータ構造と API を使用して、プライマリで耐久性の高いデータベースを使用してアプリケーションを構築したい場合も、MemoryDB が適している可能性があります。最後に、MemoryDB を使用してアプリケーションアーキテクチャを簡素化し、データベースの使用をキャッシュに置き換えて耐久性とパフォーマンスを向上させることで、コストを削減することを検討してください。
- ElastiCache は、Valkey および Redis OSS を使用して他のデータベースやデータストアからデータをキャッシュするために一般的に使用されるサービスです。既存のプライマリデータベースまたはデータストアによるデータアクセスを高速化 (マイクロ秒単位の読み取り/書き込みパフォーマンス) させるキャッシュワークロードには、ElastiCache を検討すべきです。また、Valkey または Redis OSS のデータ構造と API を使用してプライマリデータベースまたはデータストアに保存されているデータにアクセスする場合にも、ElastiCache を検討する必要があります。

リージョンとアベイラビリティーゾーンの選択

AWS クラウドコンピューティングリソースは、高可用性データセンター設備に収容されています。スケーラビリティと信頼性を向上させるために、これらのデータセンターの設備は物理的に異なる場所に配置されています。これらの場所は、リージョンとアベイラビリティーゾーンに分類されます。

AWS リージョンは大きく、広く分散した地理的な場所です。アベイラビリティーゾーンとは、別のアベイラビリティーゾーンで発生した障害から隔離するために作られた AWS リージョン内の場所です。アベイラビリティーゾーンは、同じ AWS リージョン内の他のアベイラビリティーゾーンに低価格かつ低レイテンシーのネットワーク接続を提供します。

> **Important**
>
> 各リージョンは完全に独立しています。お客様が開始した MemoryDB のアクティビティ (例えば、クラスターの作成) は、現在のデフォルトリージョンでのみ実行されます。

特定のリージョン内のクラスターを作成または操作するには、対応するリージョンのサービスエンドポイントを使用します。サービスエンドポイントについては、「MemoryDB マルチリージョン」を参照してください。

MemoryDB マルチリージョンにより、可用性と回復力の両方を向上できます。さらに、マルチリージョンアプリケーションの場合は、ローカルでの低レイテンシーの読み取りと書き込みというメリットも得られます。MemoryDB マルチリージョンの操作については、「サポートされているリージョンおよびエンドポイント」を参照してください。

ノードの配置

少なくとも 1 つのレプリカを持つクラスターは、AZ にまたがっている必要があります。単一の AZ 内のすべてを検索できる唯一の方法は、単一ノードのシャードで構成されるクラスターを使用することです。

ノードを異なる AZ に配置することで、MemoryDB は 1 つの AZ で停電などの障害が発生した場合に可用性が失われる可能性を排除します。

- MemoryDB クラスターの作成
- MemoryDB クラスターの変更

サポートされているリージョンおよびエンドポイント

MemoryDB は、複数の AWS リージョンで利用可能です。つまり、要件に合った場所で MemoryDB クラスターを起動できます。例えば、お客様の最寄りの AWS リージョンで起動するか、特定の AWS リージョンで起動して特定の法的要件を満たすことができます。さらに、MemoryDB が新しい AWS リージョンでも利用可能になると、そのリージョンでは、EmoryDB はその時点で最新の 2 つの `MAJOR.MINOR` バージョンをサポートします。バージョンの詳細については、「エンジンバージョン」を参照してください。

デフォルトでは、AWS SDK、AWS CLI、MemoryDB API、および MemoryDB コンソールは米国東部 (バージニア北部) リージョンを参照します。MemoryDB が新しいリージョンで利用できるようになるにつれ、これらのリージョン用の新しいエンドポイントも、HTTPリクエスト、AWS SDK、AWS CLI、コンソールで使用できるようになります。

各リージョンは、他のリージョンと完全に分離されるように設計されています。各リージョンには複数のアベイラビリティーゾーン (AZ) があります。別のアベイラビリティーゾーンのノードを起動して、最大限の耐障害性を実現できます。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、このトピックの最初の「リージョンとアベイラビリティーゾーンの選択」を参照してください。

MemoryDB がサポートされているリージョン

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) リージョン `us-east-2`	`memory-db.us-east-2.amazonaws.com`	HTTPS	
米国東部(バージニア州北部) リージョン `us-east-1`	`memory-db.us-east-1.amazonaws.com`	HTTPS	
US West (N. California) リージョン `us-west-1`	`memory-db.us-west-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	
米国西部 (オレゴン) リージョン us-west-2	memory-db.us-west-2.amazonaws.com	HTTPS	
カナダ (中部) リージョン ca-central-1	memory-db.ca-central-1.amazonaws.com	HTTPS	
アジアパシフィック (香港) リージョン ap-east-1	memory-db.ap-eastl-1.amazonaws.com	HTTPS	
アジアパシフィック (ムンバイ) リージョン ap-south-1	memory-db.ap-south-1.amazonaws.com	HTTPS	
アジアパシフィック (東京) リージョン ap-northeast-1	memory-db.ap-northeast-1.amazonaws.com	HTTPS	
Asia Pacific (Seoul) Region ap-northeast-2	memory-db.ap-northeast-2.amazonaws.com	HTTPS	
アジアパシフィック (シンガポール) リージョン ap-southeast-1	memory-db.ap-southeast-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	
アジアパシフィック (シドニー) リージョン ap-southeast-2	memory-db.ap-southeast-2.amazonaws.com	HTTPS	
欧州 (フランクフルト) リージョン eu-central-1	memory-db.eu-central-1.amazonaws.com	HTTPS	
欧州 (アイルランド) リージョン eu-west-1	memory-db.eu-west-1.amazonaws.com	HTTPS	
欧州 (ロンドン) リージョン eu-west-2	memory-db.eu-west-2.amazonaws.com	HTTPS	
欧州 (パリ) リージョン eu-west-3	memory-db.eu-west-3.amazonaws.com	HTTPS	
欧州 (ストックホルム) リージョン eu-north-1	memory-db.eu-north-1.amazonaws.com	HTTPS	
欧州 (ミラノ) リージョン eu-south-1	memory-db.eu-south-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	
欧州 (スペイン) リージョン eu-south-2	memory-db.eu-south-2.amazonaws.com	HTTPS	
南米 (サンパウロ) リージョン sa-east-1	memory-db.sa-east-1.amazonaws.com	HTTPS	
中国 (北京) リージョン cn-north-1	memory-db.cn-north-1.amazonaws.com.cn	HTTPS	
中国 (寧夏) リージョン cn-northwest-1	memory-db.cn-northwest-1.amazonaws.com.cn	HTTPS	

リージョンごとの AWS 製品およびサービスの表については、「リージョンごとの製品とサービス」を参照してください。

リージョン内でサポートされているアベイラビリティーゾーンのテーブルについては、「サブネットおよびサブネットグループ」を参照してください。

MemoryDB にアクセスする

MemoryDB クラスターの各エンドポイントには、アドレスとポートが含まれています。このクラスターエンドポイントは Valkey および Redis OSS クラスタープロトコルをサポートしているため、クライアントはクラスター内の各ノードの特定のロール、IP アドレス、スロットを検出できます。プライマリノードに障害が発生し、代わりにレプリカがプロモートされた場合、Valkey または Redis OSS クラスタープロトコルを使用してクラスターエンドポイントに接続し、新しいプライマリノードを検出できます。

cluster nodes または cluster slots コマンドを使用してノードエンドポイントを検出するには、クラスターエンドポイントに接続する必要があります。キーに適したノードが見つかったら、そのノードに直接接続して読み取り/書き込みリクエストを行うことができます。Valkey または Redis OSS クライアントはクラスターエンドポイントを使用して自動的に正しいノードに接続できます。

クラスター内の特定のノードをトラブルシューティングするには、ノード固有のエンドポイントを使用することもできますが、通常の使用では必要ありません。

クラスターのエンドポイントを見つけるには、以下を参照してください。

- (AWS CLI) MemoryDB クラスターのエンドポイントの検索
- MemoryDB クラスターのエンドポイントを検索する (MemoryDB API)

ノードまたはクラスターへの接続については、「redis-cli を使用して MemoryDB ノードに接続する」を参照してください。

MemoryDB セキュリティ

MemoryDB のセキュリティは次の 3 つのレベルで管理されます。

- MemoryDB クラスターと ノードに対する Amazon RDS 管理アクションを実行できるユーザーを管理するには、AWS Identity and Access Management (IAM)を使用します。IAM 認証情報を使用して AWS に接続するとき、AWS アカウントには、オペレーションの実行に必要なアクセス許可を付与する IAM ポリシーが必要です。詳細については、MemoryDB でのアイデンティティとアクセス権の管理を参照してください。
- クラスターへのアクセスレベルを制御するには、指定された権限を持つユーザーを作成し、アクセスコントロールリスト (ACL) に割り当てます。次に ACL は 1 つ以上のクラスターに関連付けられます。詳細については、「アクセスコントロールリスト (ACL) によるユーザー認証」を参照してください。
- MemoryDB クラスターは、Amazon VPC サービスに基づいて 仮想プライベートクラウド (VPC) で作成する必要があります。VPC 内の MemoryDB クラスター用のノードのエンドポイントとポートへの接続を開くことができるデバイスと Amazon EC2 インスタンスを制御するには、VPC セキュリティグループを使用します。これらのエンドポイントおよびポートの接続には TLS/SSL (Transport Layer Security/Secure Sockets Layer) を使用できます。さらに、ファイアウォールルールは、動作しているデバイスが MemoryDB クラスターへの接続を開くことができるかどうかを制御することができます。VPC の詳細については、「MemoryDB と Amazon VPC」を参照してください。

セキュリティ設定の詳細については、「MemoryDB 内のセキュリティ」を参照してください。

MemoryDB の開始方法

この演習では、MemoryDB マネジメントコンソールを使用して MemoryDB クラスターを作成、アクセス権の付与、接続、そして最後に削除する手順を順を追って説明します。

> **Note**
>
> この演習では、クラスターを作成する際に [簡易作成] オプションを使用し、MemoryDB の機能をさらに詳しく確認した後に、他の 2 つのオプションを検討することをお勧めします。

トピック

- ステップ 1: セットアップ
- ステップ 2: クラスターを作成する
- ステップ 3: クラスターへのアクセスの許可
- ステップ 4: クラスターに接続する
- ステップ 5: クラスターを削除する
- 次のステップ

ステップ 1: セットアップ

以下のトピックでは、MemoryDB の使用を開始する場合に 1 回のみ実行する必要がある操作を説明しています。

AWS アカウント へのサインアップ

AWS アカウント がない場合は、以下のステップを実行して作成します。

AWS アカウントにサインアップするには

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

   サインアップ手順の一環として、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

AWS アカウント にサインアップすると、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。

サインアップ処理が完了すると、AWS からユーザーに確認メールが送信されます。https://aws.amazon.com/ の [マイアカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理アクセスを持つユーザーを作成する

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

AWS アカウントのルートユーザーをセキュリティで保護する

1. [ルートユーザー] を選択し、AWS アカウント のメールアドレスを入力して、アカウント所有者として AWS マネジメントコンソール 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」で AWS アカウントのルートユーザーの仮想 MFA デバイスを有効にする方法 (コンソール) を確認してください。

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。

IAM アイデンティティセンターディレクトリ をアイデンティティソースとして使用するチュートリアルについては、「AWS IAM Identity Center ユーザーガイド」の「 Configure user access with the default IAM アイデンティティセンターディレクトリ」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 アイデンティティセンター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 User Guide」の「Signing in to the AWS access portal」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「Add groups」を参照してください。

プログラム的なアクセス権を付与する

AWS マネジメントコンソール の外部で AWS を操作するには、ユーザーはプログラムによるアクセスが必要です。プログラマチックアクセス権を付与する方法は、AWS にアクセスしているユーザーのタイプによって異なります。

ユーザーにプログラマチックアクセス権を付与するには、以下のいずれかのオプションを選択します。

プログラマチックアクセス権を必要とするユーザー	目的	方法
ワークフォースアイデンティティ (IAM アイデンティティセンターで管理されているユーザー)	一時的な認証情報を使用して、AWS CLI、AWS SDK、または AWS API へのプログラマチックリクエストに署名します。	使用するインターフェイスの指示に従ってください。 • AWS CLI については、AWS Command Line Interface ユーザーガイドの「AWS IAM Identity Center を使用するための AWS CLI の設定」を参照してください。 • AWS SDK、ツール、および AWS API については、AWS SDK とツールリファレンスガイドの「IAM Identity Center 認証」を参照してください。
IAM	一時的な認証情報を使用して、AWS CLI、AWS SDK、または AWS API へのプログラムによるリクエストに署名します。	IAM ユーザーガイドの「AWS リソースでの一時的な認証情報の使用」の指示に従ってください。
IAM	(非推奨) 長期的な認証情報を使用して、AWS CLI、AWS SDK、または AWS API へのプログラムによるリクエストに署名します。	使用するインターフェイスの指示に従ってください。 • AWS CLI については、AWS Command Line Interface ユーザーガイドの「IAM ユーザー認証情報を使用した認証」を参照してください。 • AWS SDK とツールについては、AWS SDK とツールリファレンスガイドの「長

プログラマチックアクセス権を必要とするユーザー	目的	方法
		期認証情報を使用して認証する」を参照してください。 • AWS API については、IAM ユーザーガイドの「IAM ユーザーのアクセスキーの管理」を参照してください。

関連トピック:

- IAM ユーザーガイドの IAM とは
- AWS 全般リファレンスの AWS セキュリティ認証情報

アクセス許可を設定する (新規の MemoryDB ユーザーのみ)

アクセス権限を付与するにはユーザー、グループ、またはロールにアクセス許可を追加します。

- AWS IAM Identity Center のユーザーとグループ:

  アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」の手順に従ってください。
- IAM 内で、ID プロバイダーによって管理されているユーザー:

  ID フェデレーションの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「サードパーティー ID プロバイダー (フェデレーション) 用のロールを作成する」を参照してください。
- IAM ユーザー:
  - ユーザーが担当できるロールを作成します。手順については「IAM ユーザーガイド」の「IAM ユーザーのロールの作成」を参照してください。
  - (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加します。詳細については「IAM ユーザーガイド」の「ユーザー (コンソール) へのアクセス権限の追加」を参照してください。

MemoryDB はサービスにリンクされたロールを作成し、それを使ってリソースをプロビジョニングして、お客様の代わりに AWS の他のリソースとサービスにアクセスします。MemoryDB がサービスにリンクされたロールを作成するには、AmazonMemoryDBFullAccess という名前の AWS マネージドポリシーを使用してください。 このロールには、サービスにリンクされたロールをサービスがユーザーに代わって作成するために必要なアクセス許可が事前に設定されています。

デフォルトのポリシーを使用せず、代わりにカスタムマネージドポリシーを使用することもできます。この場合、MemoryDB iam:createServiceLinkedRole を呼び出すアクセス許可を持っているか、自分が サービスにリンクされたロールを作成している必要があります。

詳細については次を参照してください:

- 新しいポリシーの作成 (IAM)
- MemoryDB の AWS 管理 (事前定義) ポリシー
- MemoryDB 用のサービスリンクロール

AWS CLI のダウンロードと設定

AWS CLI は、http://aws.amazon.com/cli で入手できます。Windows、MacOS、または Linux 上で実行できます。AWS CLI をダウンロードしたら、以下の手順に従って、インストールと設定を行います。

1. AWS コマンドラインインターフェイスのユーザーガイドに移動します。
2. 「AWS CLI のインストール」および「AWS CLI の設定」の手順に従います。

ステップ 2: クラスターを作成する

実稼働用のクラスターを作成する前に、ビジネスニーズに合わせてクラスターをどのように設定するかを検討する必要があります。これらの問題については、クラスターを準備する セクションで対応します。この「使用開始」の演習では、適用するデフォルトの設定値を受け入れます。

作成するクラスターはライブとなりますが、サンドボックスで実行されるわけではありません。インスタンスを削除するまで、MemoryDB の標準使用料が発生します。ここで説明する演習を一気に完了し、終了時にクラスターを削除すれば、使用料合計はごくわずかです (通常 1 ドル未満です)。MemoryDB の使用料の詳細については、「MemoryDB」を参照してください。

クラスターは、Amazon VPC サービスに基づいて Virtual Private Cloud (VPC) で起動されます

MemoryDB クラスターの作成

以下の例は、AWS マネジメントコンソール、AWS CLI および MemoryDB APIを使用してクラスターを作成する方法を示しています。

クラスターの作成 (コンソール)

コンソールを使用して MemoryDB クラスターを作成するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. ナビゲーションペインで、[クラスター] を選択し、[作成] を選択します。

Easy create

1. [設定] セクションに情報を入力します。これにより、クラスターのノードタイプとデフォルト設定が構成されます。次のオプションから、必要となる適切なメモリサイズとネットワークパフォーマンスを選択します:
   - 本番稼働用
   - 開発/テスト
   - デモ
2. [クラスター情報] セクションを完了します。
   a. 名前 に、クラスターの名前を入力します。

クラスターの命名に関する制約は次のとおりです。

- 1～40 個の英数字またはハイフンを使用する必要があります。
- 先頭は文字を使用する必要があります。
- 連続する 2 つのハイフンを含めることはできません。
- ハイフンで終わることはできません。

b. 説明 ボックスに、このクラスターの説明を入力します。

3. [サブネットグループ] セクションを完了します。

- [サブネットグループ] で、新しいサブネットグループを作成するか、このクラスターに適用する既存のサブネットグループを選択します。新しいものを作成する場合:
  - [名前] を入力する
  - [説明] を入力する
  - マルチ AZ を有効にした場合は、異なるアベイラビリティーゾーンに存在する少なくとも 2 つのサブネットをサブネットグループに含める必要があります。詳細については、「サブネットおよびサブネットグループ」を参照してください。
  - 新しいサブネットグループを作成していて、既存の VPC がない場合は、VPC を作成するように求められます。詳細については、「Amazon VPC ユーザーガイド」の「Amazon VPC とは」を参照してください。

4. [ベクトル検索] で、[ベクトル検索機能を有効にする] を選択して、ベクトル埋め込みを保存し、ベクトル検索を実行できます。これにより、エンジンバージョンの互換性、パラメータグループ、およびシャードの値が修正されることに留意してください。詳細については、「ベクトル検索」を参照してください。

5. [デフォルト設定を表示] を選択します。

   [簡易作成] を使用する場合、残りのクラスター設定はデフォルトで指定されます。これらの設定の一部は作成後に変更できることに留意してください ([作成後に編集可能] と示されています)。

6. [タグ] には、オプションでタグを適用してクラスターを検索およびフィルタリングしたり、AWS コストを追跡したりできます。

7. すべてのエントリと選択を確認し、必要な修正を行います。準備が完了したら、[作成] を選択してクラスターを起動するか、[キャンセル] を選択してオペレーションをキャンセルします。

クラスターのステータスが 使用可能 になり次第、EC2 にアクセス権を付与して接続し、使用を開始できます。詳細については、ステップ 3: クラスターへのアクセスの許可を参照してください。

> **Important**
>
> クラスターが使用可能になった直後から、クラスターがアクティブである間は (実際に使用していない場合でも)、時間に応じた料金が発生します。このクラスターに対する課金を中止するには、クラスターを削除する必要があります。「ステップ 5: クラスターを削除する」を参照してください。

Create new cluster

1. [クラスター情報] セクションを完了します。
   a. 名前 に、クラスターの名前を入力します。

      クラスターの命名に関する制約は次のとおりです。

      - 1 ~ 40 個の英数字またはハイフンを使用する必要があります。
      - 先頭は文字を使用する必要があります。
      - 連続する 2 つのハイフンを含めることはできません。
      - ハイフンで終わることはできません。

   b. 説明 ボックスに、このクラスターの説明を入力します。
2. [サブネットグループ] セクションを完了します。
   - [サブネットグループ] で、新しいサブネットグループを作成するか、このクラスターに適用する既存のサブネットグループを選択します。新しいものを作成する場合:
     - [名前] を入力する
     - [説明] を入力する
     - マルチ AZ を有効にした場合は、異なるアベイラビリティーゾーンに存在する少なくとも 2 つのサブネットをサブネットグループに含める必要があります。詳細については、「サブネットおよびサブネットグループ」を参照してください。

- 新しいサブネットグループを作成していて、既存の VPC がない場合は、VPC を作成するように求められます。詳細については、「Amazon VPC ユーザーガイド」の「Amazon VPC とは」を参照してください。

3. [クラスター設定] セクションに入力します。

   a. [ベクトル検索機能を有効にする] で、これを有効にしてベクトル埋め込みを保存し、ベクトル検索を実行できます。これにより、エンジンバージョンの互換性、パラメータグループ、およびシャードの値が修正されることに留意してください。詳細については、「ベクトル検索」を参照してください。

   b. エンジンバージョンとの互換性を保つため、デフォルトをそのまま使用してください。例えば、Valkey ではデフォルトは 7.2.6 で、Redis OSS ではデフォルトは 6.2 です。

   c. [ポート] には、デフォルトのポート 6379 をそのまま使用するか、別のポートを使用する理由がある場合はポート番号を入力します。

   d. [パラメータグループ] で、ベクトル検索を有効にしている場合は `default.memorydb-valkey7.search` を使用します。それ以外の場合は、Valkey は、`default.memorydb-valkey7` パラメータグループを受け入れます。

      パラメータグループはクラスターのランタイムパラメータを制御します。パラメータグループの詳細については、「エンジン固有のパラメータ」を参照してください。

   e. [ノードタイプ] では、必要なノードタイプの値 (および関連するメモリサイズ) を選択します。

      r6gd ファミリーからノードタイプを選択すると、データ階層化が自動的に有効になり、データストレージがメモリと SSD に分割されます。詳細については、「データ階層化」を参照してください。

   f. [シャード数] で、このクラスターに必要なシャード (パーティション/ノードグループ) の数を選択します。クラスターの可用性を高めるため、少なくとも 2 つのシャードを追加することをお勧めします。

      クラスター内のシャード数を動的に変更できます。詳細については、「MemoryDB クラスターのスケーリング」を参照してください。

   g. シャード当たりのレプリカ数 で、各シャードに必要なリードレプリカのノード数を選択します。

      には次の制限があります:

- マルチ AZ が有効になっている場合は、シャードごとに少なくとも 1 つのレプリカがあることを確認してください。
- コンソールを使用してクラスターを作成する場合、シャードごとのレプリカ数は同じになります。

h. 次へ を選択します。

i. [詳細設定] セクションを完了します。

i. セキュリティグループ で、このクラスターに必要なセキュリティグループを選択します。セキュリティグループは、クラスターへのネットワークアクセスを制御するためのファイアウォールとして機能します。VPC のデフォルトのセキュリティグループを使用するか、新しいセキュリティグループを作成できます。

VPC セキュリティグループの詳細については、Amazon VPC ユーザーガイドの「VPC のセキュリティグループ」を参照してください。

ii. データを暗号化するには、次のオプションがあります。

- 保管時の暗号化 – ディスクに保存されているデータの暗号化を有効にします。詳細については、「保管時の暗号化」を参照してください。

> **Note**
>
> 別の暗号化キーを指定するには、[カスタマー管理の AWS KMS キー] を選択し、キーを選択することもできます。

- 転送時の暗号化 – 転送中のデータの暗号化を有効にします。暗号化なしを選択すると、「オープンアクセス」というオープンアクセスコントロールリストがデフォルトユーザーで作成されます。詳細については、「アクセスコントロールリスト (ACL) によるユーザー認証」を参照してください。

iii. [スナップショット] には、オプションでスナップショットの保存期間とスナップショットウィンドウを指定します。デフォルトでは、[自動スナップショットを有効にする] があらかじめ選択されています。

iv. [メンテナンスウィンドウ] には、オプションでメンテナンスウィンドウを指定します。メンテナンスウィンドウは、MemoryDB がクラスターのシステムメンテナンスを毎週スケジュールする時間の長さ (通常は 1 時間単位) です。MemoryDB がメンテナンス期間の日時を選択することを許可するか ([指定なし])、自分で日時と期間を選択できます ([メンテナンス期間の指定])。リストから [メンテナンスウィンドウを

指定] を選択した場合は、メンテナンスウィンドウの [開始日]、[開始時間]、および [期間] (時間単位) を選択します。すべての時刻は協定世界時 (UCT) です。

詳細については、「メンテナンスの管理」を参照してください。

v. [通知] で、既存の Amazon Simple Notification Service (Amazon SNS) トピックを選択するか、手動 ARN 入力を選択してトピックの Amazon リソースネーム (ARN) を入力します。Amazon SNS では、インターネットに接続されたスマートデバイスに通知をプッシュすることができます。デフォルトでは、通知は無効になります。詳細については、https://aws.amazon.com/sns/ を参照してください。

vi. [タグ] には、オプションでタグを適用してクラスターを検索およびフィルタリングしたり、AWS コストを追跡したりできます。

j. すべてのエントリと選択を確認し、必要な修正を行います。準備が完了したら、[作成] を選択してクラスターを起動するか、[キャンセル] を選択してオペレーションをキャンセルします。

クラスターのステータスが 使用可能 になり次第、EC2 にアクセス権を付与して接続し、使用を開始できます。詳細については、ステップ 3: クラスターへのアクセスの許可を参照してください。

> **Important**
>
> クラスターが使用可能になった直後から、クラスターがアクティブである間は (実際に使用していない場合でも)、時間に応じた料金が発生します。このクラスターに対する課金を中止するには、クラスターを削除する必要があります。「ステップ 5: クラスターを削除する」を参照してください。

Restore from snapshots

[スナップショットのソース] で、データの移行元のソーススナップショットを選択します。詳細については、「スナップショットおよび復元 」を参照してください。

> **Note**
>
> 新しいクラスターでベクトル検索を有効にする場合は、ソーススナップショットでもベクトル検索を有効にする必要があります。

ターゲットクラスターは、ソースクラスターの設定にデフォルト設定されます。オプションで、ターゲットクラスターで次の設定を変更できます。

1. クラスター情報

   a. 名前 に、クラスターの名前を入力します。

      クラスターの命名に関する制約は次のとおりです。

      - 1~40 個の英数字またはハイフンを使用する必要があります。
      - 先頭は文字を使用する必要があります。
      - 連続する 2 つのハイフンを含めることはできません。
      - ハイフンで終わることはできません。

   b. 説明 ボックスに、このクラスターの説明を入力します。

2. サブネットグループ

   - [サブネットグループ] で、新しいサブネットグループを作成するか、このクラスターに適用する既存のサブネットグループを選択します。新しいものを作成する場合:

     - [名前] を入力する
     - [説明] を入力する
     - マルチ AZ を有効にした場合は、異なるアベイラビリティーゾーンに存在する少なくとも 2 つのサブネットをサブネットグループに含める必要があります。詳細については、「サブネットおよびサブネットグループ」を参照してください。
     - 新しいサブネットグループを作成していて、既存の VPC がない場合は、VPC を作成するように求められます。詳細については、「Amazon VPC ユーザーガイド」の「Amazon VPC とは」を参照してください。

3. クラスター設定

   a. [ベクトル検索機能を有効にする] で、これを有効にしてベクトル埋め込みを保存し、ベクトル検索を実行できます。これにより、エンジンバージョンの互換性、パラメータグループ、およびシャードの値が修正されることに留意してください。詳細については、「ベクトル検索」を参照してください。

   b. エンジンバージョンとの互換性を保つため、デフォルト 6.2 をそのまま使用してください。

   c. [ポート] には、デフォルトのポート 6379 をそのまま使用するか、別のポートを使用する理由がある場合はポート番号を入力します。

d. [パラメータグループ] で、ベクトル検索を有効にしている場合は `default.memorydb-redis7.search.preview` を使用します。それ以外の場合は、`default.memorydb-redis7` パラメータグループを受け入れます。

   パラメータグループはクラスターのランタイムパラメータを制御します。パラメータグループの詳細については、「エンジン固有のパラメータ」を参照してください。

e. [ノードタイプ] では、必要なノードタイプの値 (および関連するメモリサイズ) を選択します。

   r6gd ファミリーからノードタイプを選択すると、データ階層化が自動的に有効になり、データストレージがメモリと SSD に分割されます。詳細については、「データ階層化」を参照してください。

f. [シャード数] で、このクラスターに必要なシャード (パーティション/ノードグループ) の数を選択します。クラスターの可用性を高めるため、少なくとも 2 つのシャードを追加することをお勧めします。

   クラスター内のシャード数を動的に変更できます。詳細については、「MemoryDB クラスターのスケーリング」を参照してください。

g. シャード当たりのレプリカ数 で、各シャードに必要なリードレプリカのノード数を選択します。

   には次の制限があります:

   - マルチ AZ が有効になっている場合は、シャードごとに少なくとも 1 つのレプリカがあることを確認してください。
   - コンソールを使用してクラスターを作成する場合、シャードごとのレプリカ数は同じになります。

h. 次へ を選択します。

i. 詳細設定

   i. セキュリティグループ で、このクラスターに必要なセキュリティグループを選択します。セキュリティグループは、クラスターへのネットワークアクセスを制御するためのファイアウォールとして機能します。VPC のデフォルトのセキュリティグループを使用するか、新しいセキュリティグループを作成できます。

      VPC セキュリティグループの詳細については、Amazon VPC ユーザーガイドの「VPC のセキュリティグループ」を参照してください。

ii. データを暗号化するには、次のオプションがあります。

- 保管時の暗号化 – ディスクに保存されているデータの暗号化を有効にします。詳細については、「保管時の暗号化」を参照してください。

> **Note**
>
> 別の暗号化キーを指定するには、[カスタマー管理の AWS KMS キー] を選択し、キーを選択することもできます。

- 転送時の暗号化 – 転送中のデータの暗号化を有効にします。暗号化なしを選択すると、「オープンアクセス」というオープンアクセスコントロールリストがデフォルトユーザーで作成されます。詳細については、「アクセスコントロールリスト (ACL) によるユーザー認証」を参照してください。

iii. [スナップショット] には、オプションでスナップショットの保存期間とスナップショットウィンドウを指定します。デフォルトでは、[自動スナップショットを有効にする] があらかじめ選択されています。

iv. [メンテナンスウィンドウ] には、オプションでメンテナンスウィンドウを指定します。メンテナンスウィンドウは、MemoryDB がクラスターのシステムメンテナンスを毎週スケジュールする時間の長さ (通常は 1 時間単位) です。MemoryDB がメンテナンス期間の日時を選択することを許可するか ([指定なし])、自分で日時と期間を選択できます ([メンテナンス期間の指定])。リストから [メンテナンスウィンドウを指定] を選択した場合は、メンテナンスウィンドウの [開始日]、[開始時間]、および [期間] (時間単位) を選択します。すべての時刻は協定世界時 (UCT) です。

詳細については、「メンテナンスの管理」を参照してください。

v. [通知] で、既存の Amazon Simple Notification Service (Amazon SNS) トピックを選択するか、手動 ARN 入力を選択してトピックの Amazon リソースネーム (ARN) を入力します。Amazon SNS では、インターネットに接続されたスマートデバイスに通知をプッシュすることができます。デフォルトでは、通知は無効になります。詳細については、https://aws.amazon.com/sns/ を参照してください。

vi. [タグ] には、オプションでタグを適用してクラスターを検索およびフィルタリングしたり、AWS コストを追跡したりできます。

j. すべてのエントリと選択を確認し、必要な修正を行います。準備が完了したら、[作成] を選択してクラスターを起動するか、[キャンセル] を選択してオペレーションをキャンセルします。

クラスターのステータスが 使用可能 になり次第、EC2 にアクセス権を付与して接続し、使用を開始できます。詳細については、ステップ 3: クラスターへのアクセスの許可を参照してください。

> **Important**
>
> クラスターが使用可能になった直後から、クラスターがアクティブである間は (実際に使用していない場合でも)、時間に応じた料金が発生します。このクラスターに対する課金を中止するには、クラスターを削除する必要があります。「ステップ 5: クラスターを削除する」を参照してください。

クラスターの作成 (AWS CLI)

AWS CLI を使用してクラスターを作成するには、「create-cluster」を参照してください。以下に例を示します。

Linux、macOS、Unix の場合:

```
aws memorydb create-cluster \
    --cluster-name my-cluster \
    --node-type db.r6g.large \
    --acl-name my-acl \
    --engine valkey \
    --subnet-group my-sg
```

Windows の場合:

```
aws memorydb create-cluster ^
   --cluster-name my-cluster ^
   --node-type db.r6g.large ^
   --acl-name my-acl ^
   --engine valkey
   --subnet-group my-sg
```

以下のような JSON レスポンスが表示されます。

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "creating",
        "NumberOfShards": 1,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "7.2",
        "EnginePatchVersion": "7.2.6",
        "ParameterGroupName": "default.memorydb-valkey7",
        "Engine": "valkey"
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
```

```
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxxxx:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "ACLName": "my-acl",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

クラスターのステータスが available に変わったら、クラスターの使用を開始できます。

> **Important**
>
> クラスターが使用可能になった直後から、クラスターがアクティブである間は (実際に使用していない場合でも)、時間に応じた料金が発生します。このクラスターに対する課金を中止するには、クラスターを削除する必要があります。「ステップ 5: クラスターを削除する」を参照してください。

クラスターの作成 (MemoryDB API)

MemoryDB API を使用してクラスターを作成するには、「クラスターの作成」アクションを使用します。

> **Important**
>
> クラスターが使用可能になった直後から、そのクラスターがアクティブである間は (クラスターを使用していない場合でも)、時間に応じた料金が発生します。このクラスターに対する課金を中止するには、クラスターを削除する必要があります。「ステップ 5: クラスターを削除する」を参照してください。

認証のセットアップ

クラスターの 認証の設定の詳細については、「IAM を使用した認証」と「アクセスコントロールリスト (ACL) によるユーザー認証」を参照してください。

ステップ 3: クラスターへのアクセスの許可

このセクションでは、Amazon EC2 インスタンスの起動と接続に慣れていることを前提としています。詳細については、「Amazon EC2 入門ガイド」を参照してください。

すべての MemoryDB クラスターは Amazon EC2 インスタンスからアクセスするように設計されています。Amazon Elastic Container サービスまた AWS Lambda はで実行されているコンテナ化されたアプリケーションやサーバーレスアプリケーションからもアクセスできます。最も一般的なシナリオは、同じ Amazon Virtual Private Cloud (Amazon VPC) 内の Amazon EC2 インスタンスから MemoryDB クラスターにアクセスすることであり、それがこの演習でのケースとなります。

EC2 インスタンスからクラスターに接続するには、EC2 インスタンスにクラスターへのアクセスを許可する必要があります。

最も一般的ユースケースは、EC2 インスタンスにデプロイされたアプリケーションが同じ VPC のクラスターに接続する必要がある場合です。同じ VPC 内の EC2 インスタンスとクラスター間のアクセスを管理する方法として最も簡単なのは、次の方法です。

1. クラスターの VPC セキュリティグループを作成します。このセキュリティグループを使用して、クラスターへのアクセスを制限できます。例えば、クラスターを作成したときに割り当てたポートと、クラスターにアクセスするのに使用する IP アドレスを使用して TCP へのアクセスを許可する、このセキュリティグループのカスタムルールを作成できます。

   MemoryDB クラスターのデフォルトのポートは 6379 です。
2. EC2 インスタンス (ウェブサーバーとアプリケーションサーバー) 用の VPC セキュリティグループを作成します。このセキュリティグループは、必要に応じて VPC のルーティングテーブルを介してインターネットから EC2 インスタンスへのアクセスを許可できます。例えば、ポート 22 経由で EC2 インスタンスへの TCP アクセスを許可するルールをこのセキュリティグループに設定できます。
3. EC2 インスタンス用に作成したセキュリティグループからの接続を許可するクラスターのセキュリティグループで、カスタムルールを作成します。これは、セキュリティグループのメンバーにクラスターへのアクセスを許可します。

他のセキュリティグループからの接続を許可する VPC セキュリティグループでルールを作成するには

1. AWS マネジメントコンソールにサインインして、Amazon VPC コンソール (https://console.aws.amazon.com/vpc) を開きます。

2. 左のナビゲーションペインで セキュリティグループを選択します。
3. クラスターに使用するセキュリティグループを選択または作成します。インバウンドルールで、インバウンドルールの編集 を選択し、ルールの追加 を選択します。このセキュリティグループは、他のセキュリティグループのメンバーへのアクセスを許可します。
4. Type で Custom TCP Rule を選択します。

    a. Port Range ポートには、クラスター作成時に使用したポートを指定します。

        MemoryDB クラスターのデフォルトのポートは 6379 です。

    b. ソース ボックスに、セキュリティグループの ID の入力を開始します。リストから、Amazon EC2 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。
5. 終了したら、保存 を選択します。

アクセスを有効にしたら、次のセクションで説明するように、クラスターに接続する準備が整いました。

別の Amazon VPC、別の AWS リージョン、または企業ネットワークから MemoryDB クラスターにアクセスすることの詳細については、以下を参照してください。

- Amazon VPC の MemoryDB クラスターにアクセスするためのアクセスパターン
- 外部からのMemoryDBリソースへのアクセス AWS

ステップ 4: クラスターに接続する

続行する前に、「ステップ 3: クラスターへのアクセスの許可」を完了します。

このセクションでは、Amazon EC2 インスタンスが作成済みであり、このインスタンスに接続できることを前提としています。これを行う手順については、「Amazon EC2 入門ガイド」を参照してください。

Amazon EC2 インスタンスは、許可されている場合にのみクラスターに接続できます。

クラスターエンドポイントを見つける

クラスターが利用可能な状態であり、クラスターへのアクセスを許可されている場合は、Amazon EC2 インスタンスにログインしてクラスターに接続できます。そのためには、最初にエンドポイントを確認する必要があります。

エンドポイントを見つける方法の詳細については、以下を参照してください。

- (AWS マネジメントコンソール) MemoryDB クラスターのエンドポイントの検索
- (AWS CLI) MemoryDB クラスターのエンドポイントの検索
- MemoryDB クラスターのエンドポイントを検索する (MemoryDB API)

メモリ DB クラスターへの接続 (Linux)

これで、必要なエンドポイントが手に入ったので、EC2 インスタンスにログインし、クラスターに接続できます。次の例では、cli ユーティリティを使用して、Ubuntu 22 でクラスターに接続します。cli の最新バージョンでは、暗号化/認証が有効なクラスターを接続するための SSL/TLS もサポートしています。

redis-cli を使用して MemoryDB ノードに接続する

MemoryDB ノードの Secure Socket Layer (SSL) を使用するクライアントを使用します。Amazon Linux や Amazon Linux 2 で、TLS/SSL を使用して redis-cli を使用することもできます。

redis-cli を使用して、Amazon Linux 2 または Amazon Linux の MemoryDB クラスターに接続するには

1. redis-cli ユーティリティをダウンロードし、コンパイルします。このユーティリティは Redis OSS ソフトウェアディストリビューションに含まれています。

2. EC2 インスタンスのコマンドプロンプトで、使用している Linux のバージョンに適したコマンドを入力します。

   Amazon Linux 2023

   Amazon Linux 2023 を使用している場合は、次のように入力します。

   ```
   sudo yum install redis6 -y
   ```

   次のコマンドを入力します。この例に示されているクラスターのエンドポイントやポートを、使用するクラスターのエンドポイントやポートに置き換えてください。

   ```
   redis-cli -h Primary or Configuration Endpoint --tls -p 6379
   ```

   エンドポイントの検索の詳細については、「ノードのエンドポイントの検索」を参照してください。

   Amazon Linux 2

   Amazon Linux 2 を使用している場合は、次のように入力します。

   ```
   sudo yum -y install openssl-devel gcc
   wget https://download.redis.io/releases/redis-7.2.5.tar.gz
   tar xvzf redis-7.2.5.tar.gz
   cd redis-7.2.5
   make distclean
   make redis-cli BUILD_TLS=yes
   sudo install -m 755 src/redis-cli /usr/local/bin/
   ```

   Amazon Linux

   Amazon Linux を使用している場合は、次のように入力します。

   ```
   sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel clang wget
   wget https://download.redis.io/releases/redis-7.2.5.tar.gz
   tar xvzf redis-7.2.5.tar.gz
   cd redis-7.2.5
   make redis-cli CC=clang BUILD_TLS=yes
   sudo install -m 755 src/redis-cli /usr/local/bin/
   ```

Amazon Linux では、以下の追加ステップを実行する必要がある場合もあります。

```
sudo yum install clang
CC=clang make
sudo make install
```

3. redis-cli ユーティリティをダウンロードしてインストールしたら、オプションの make-test コマンドを実行することをお勧めします。
4. 暗号化と認証が有効になっているクラスターに接続するには、次のコマンドを入力します。

```
redis-cli -h Primary or Configuration Endpoint --tls -a 'your-password' -p 6379
```

> **Note**
>
> Amazon Linux 2023 に redis6 をインストールした場合、redis-cli の代わりに redis6-cli コマンドを使用できるようになりました。
>
> ```
> redis6-cli -h Primary or Configuration Endpoint --tls -p 6379
> ```

ステップ 5: クラスターを削除する

クラスターが使用可能な状態であれば、実際に使用しているかどうかに関係なく課金されます。課金を中止するには、クラスターを削除します。

> **Warning**
>
> - MemoryDB クラスターを削除しても、手動スナップショットは保持されます。クラスターを削除する前に最終スナップショットを作成することもできます。自動スナップショットは保持されません。詳細については、「スナップショットおよび復元」を参照してください。
> - 最終スナップショットを作成するには、CreateSnapshot アクセス許可が必要です。このアクセス許可がない場合、API コールは Access Denied 例外で失敗します。

の使用AWS マネジメントコンソール

次の手順では、デプロイから 1 つのクラスターを削除します。複数のクラスターを削除するには、削除するクラスターごとに同じ手順を繰り返してください。別のクラスターの削除手順を開始する前に、1 つのクラスターの削除が終了するのを待つ必要はありません。

クラスターを削除するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 削除するクラスターを選択するには、クラスターのリストでクラスター名の横にあるラジオボタンを選択します。この場合、ステップ 2: クラスターを作成する で作成したクラスターの名前です。
3. アクションとして、Delete (削除) を選択します。
4. まず、削除する前にクラスターのスナップショットを作成するかどうかを選択し、delete 確認ボックスに [削除] と入力してクラスターを削除するか、[キャンセル] を選択してクラスターを保持します。

   Delete を選択した場合は、クラスターのステータスが削除中に変わります。

クラスターがクラスターのリストに表示されなくなるとすぐに、課金が停止されます。

の使用AWS CLI

次のコードはクラスター my-cluster を削除します。この場合、my-cluster を、ステップ 2: クラスターを作成する で作成したクラスターの名前に置き換えます。

```
aws memorydb delete-cluster --cluster-name my-cluster
```

delete-cluster CLI オペレーションは 1 つのクラスターのみを削除します。複数のクラスターを削除する場合は、削除するクラスターごとに delete-cluster を呼び出します。1 つのクラスターの削除が終了するまで待たなくても次のクラスターを削除できます。

Linux、macOS、Unix の場合:

```
aws memorydb delete-cluster \
    --cluster-name my-cluster \
    --region us-east-1
```

Windows の場合:

```
aws memorydb delete-cluster ^
    --cluster-name my-cluster ^
    --region us-east-1
```

詳細については、 を参照してください。delete-cluster

MemoryDB API の使用

次のコードはクラスター my-cluster を削除します。この場合、my-cluster を、ステップ 2: クラスターを作成する で作成したクラスターの名前に置き換えます。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DeleteCluster
    &ClusterName=my-cluster
    &Region=us-east-1
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T220302Z
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210802T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210802T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

DeleteCluster API オペレーションは 1 つのクラスターのみを削除します。複数のクラスターを削除する場合は、削除するクラスターごとに DeleteCluster を呼び出します。1 つのクラスターの削除が終了するまで待たなくても次のクラスターを削除できます。

詳細については、「DeleteCluster」を参照してください。

次のステップ

これで入門演習は完了しました。MemoryDB と利用可能なツールについてさらに知識を深めるには、次の各セクションを参照してください。

- の開始方法AWS
- Amazon Web Services のツール

- AWS Command Line Interface
- 「MemoryDB API リファレンス」。

ノードの管理

ノードとは、MemoryDB のデプロイにおける最小の構成要素です。ノードはクラスターに属するシャードに属します。各ノードは、クラスターの作成時または最終変更時に選択されたエンジンバージョンを実行します。各ノードはそれぞれ Domain Name Service (DNS) 名とポートを持っています。複数のタイプの MemoryDB ノードがサポートされており、関連付けられたメモリ量と計算能力がそれぞれ異なります。

トピック

- MemoryDB のノードとシャード
- サポートされているノードの種類
- MemoryDB のリザーブドノード
- ノードの置換

ノードに関する重要な操作は次のとおりです。

- クラスターからのノードの追加/削除
- Scaling (スケーリング)
- 接続エンドポイントの検索

MemoryDB のノードとシャード

シャードは、それぞれクラスターにラップされたノードの階層的配列です。シャードはレプリケーションをサポートします。シャード内では、1 つのノードが読み取り/書き込みのプライマリノードとなります。シャード内の他のすべてのノードは、プライマリノードの読み取り専用のレプリカとなります。MemoryDB はクラスター内の複数のシャードをサポートします。このサポートにより、MemoryDB クラスター内でデータを分割できます。

MemoryDB はシャードによるレプリケーションをサポートします。API オペレーション「クラスターの詳細」は、シャードをメンバーノード、ノード名、エンドポイント、およびその他の情報とともに一覧表示します。

MemoryDB クラスターは作成された後、変更 (スケールインまたはスケールアウト) できます。詳細については、「Scaling (スケーリング)」および「ノードの置換」を参照してください。

新しいクラスターを作成するときに、古いクラスターからのデータをシードして、空から開始しないようにすることができます。これを行うことは、ノードタイプまたはエンジンバージョンの変更が必要な場合、または Amazon ElastiCache (Redis OSS) から移行する必要がある場合に役立ちます。詳細については、「手動スナップショットの作成」および「スナップショットからの復元」を参照してください。

サポートされているノードの種類

MemoryDB は次のノードタイプをサポートします。

メモリ最適化: | | | |

インスタンスタイプ	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)	拡張 I/O 多重化 (Valkey 7.2 および Redis OSS 7.0.4 以降)	エンジンの最小バージョン
db.r7g.large	0.937	12.5	なし	6.2
db.r7g.xlarge	1.876	12.5	なし	6.2
db.r7g.2xlarge	3.75	15	はい	6.2
db.r7g.4xlarge	7.5	15	はい	6.2
db.r7g.8xlarge	15	該当なし	はい	6.2
db.r7g.12xlarge	22.5	該当なし	はい	6.2
db.r7g.16xlarge	30	該当なし	はい	6.2
db.r6g.large	0.75	10.0	なし	6.2
db.r6g.xlarge	1.25	10.0	なし	6.2
db.r6g.2xlarge	2.5	10.0	はい	6.2
db.r6g.4xlarge	5.0	10.0	はい	6.2
db.r6g.8xlarge	12	該当なし	はい	6.2
db.r6g.12xlarge	20	該当なし	はい	6.2
db.r6g.16xlarge	25	該当なし	はい	6.2

データ階層化で最適化されたメモリ

インスタンスタイプ	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)	拡張 I/O 多重化 (Valkey 7.2 および Redis OSS 7.0.4 以降)	エンジンの最小バージョン
db.r6gd.xlarge	1.25	10	なし	6.2
db.r6gd.2xlarge	2.5	10	なし	6.2
db.r6gd.4xlarge	5.0	10	なし	6.2
db.r6gd.8xlarge	12	該当なし	なし	6.2

汎用ノード

インスタンスタイプ	ベースライン帯域幅 (Gbps)	バースト帯域幅 (Gbps)	拡張 I/O 多重化 (Valkey 7.2 および Redis OSS 7.0.4 以降)	エンジンの最小バージョン
db.t4g.small	0.128	5.0	なし	6.2
db.t4g.medium	0.256	5.0	なし	6.2

利用可能な AWS リージョンについては、「MemoryDB の料金」を参照してください

すべてのノードタイプは、仮想プライベートクラウド (VPC) で作成されます。

MemoryDB のリザーブドノード

オンデマンドノードの料金と比べて、リザーブドノードには大幅な割引が適用されます。リザーブドノードは物理ノードではなく、アカウント内のオンデマンドノードの使用に適用される割引です。リザーブドノードの割引は、ノードタイプと AWS リージョンによって異なります。

> **Note**
>
> 現在の MemoryDB リザーブドノードはすべて、Redis OSS エンジンを実行するノードの料金に基づいており、Redis OSS エンジンを実行するノードを対象としています。これらのリザーブドノードは、「サイズ柔軟なリザーブドノード」で説明されているように Valkey エンジンに適用できますが、Valkey 固有のリザーブドノードは使用できません。

リザーブドノードを使用する一般的なプロセスは次のとおりです。

- 利用可能なリザーブドノードサービスに関する情報を確認する
- AWS マネジメントコンソール、AWS Command Line Interface、または SDK を使用してリザーブドノードサービスを購入する
- 既存のリザーブドノードに関する情報を確認します

トピック

- リザーブドノードの概要
- 提供タイプ
- サイズ柔軟なリザーブドノード
- ノードを Redis OSS から Valkey にアップグレードする
- リザーブドノードの削除
- リザーブドノードの操作

リザーブドノードの概要

MemoryDB の予約ノードを購入すると、予約ノードの有効期間中、特定のノードタイプで割引料金で利用できることを約束することになります。MemoryDB のリザーブドノードを使用するには、オンデマンドノードの場合と同様に、新しいノードを作成します。新しく作成するノードは、リザーブドノードの仕様と完全に一致する必要があります。新しいノードの仕様がアカウント内の既存のリ

ザーブドノードと一致する場合は、リザーブドノードに適用される割引料金で請求されます。一致しない場合、ノードはオンデマンド料金で請求されます。AWS マネジメントコンソール、AWS CLI、または MemoryDB API を使用して、使用可能なリザーブドノードを一覧表示および購入できます。

MemoryDB は、メモリ最適化された R7g、R6g、および R6gd (データ階層化あり) ノード用のリザーブドノードを提供しています。料金に関する情報については、「MemoryDB の料金」を参照してください。

提供タイプ

リザーブドノードには、予想される使用量に基づいて、前払いなし、一部前払い、全前払いの 3 種類のオプションがあり、MemoryDB のコストを最適化できます。

前払いなし - このオプションは前払い料金なしでリザーブドノードへのアクセスを提供します。前払いなしのリザーブドノードでは、使用量にかかわらず、期間内の時間はすべて、割引された時間料金で請求されます。前払い料金は必要ありません。

一部前払い — このオプションでは、リザーブドノードの一部を前払いする必要があります。期間内の残りの時間は、使用量にかかわらず、割引された時間料金で請求されます。

すべて前払い - 期間のスタート時に全額を支払います。使用時間数に関係なく、残りの期間にそれ以外のコストは生じません。

3 つの提供タイプはすべて、1 年および 3 年の期間で利用できます。

サイズ柔軟なリザーブドノード

リザーブドノードを購入する際、指定する項目の 1 つはノードのタイプ db.r6g.xlarge などです。ノードタイプの詳細については、「MemoryDB の料金」を参照してください。

既存の ノードがあり、これをスケールして容量を増やす必要がある場合、リザーブド ノードはスケールした ノードに自動的に適用されます。つまり、リザーブドノードは、同じノードファミリーのあらゆるサイズの使用に自動的に適用されます。サイズ柔軟のリザーブドノードは、同じ AWS リージョンのノードで利用できます。サイズ柔軟なリザーブドノードは、そのノードファミリーでしかスケールできません。例えば、db.r6.large のリザーブドノードは db.r6.xlarge には適用できますが、db.r6g.large には適用できません。db.r6 と db.r6g は異なるインスタンスクラスタイプであるためです。

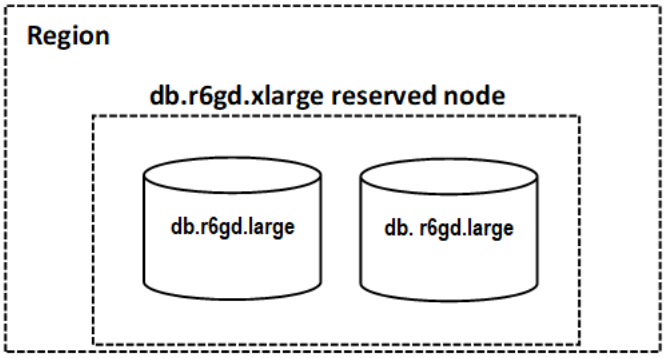
柔軟性とは、同じノードクラスタイプ内の設定間を自由に移動できることを意味します。例えば、追加費用なしで r6g.xlarge リザーブドノード (8 つの正規化された単位) から、同じ AWS リージョン内

の 2 つの r6g.large リザーブドノード (8 つの正規化された単位) (2*4 = 8 つの正規化された単位) に移動できます。

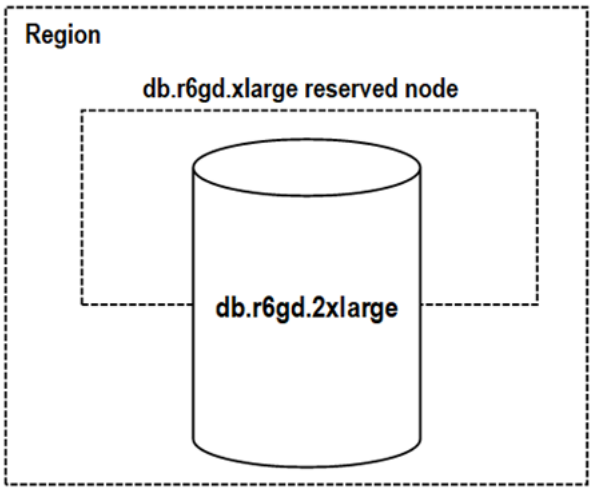
リザーブド ノードのサイズ別の使用は、正規化された単位を使用して比較できます。例えば、2 つの db.r6g.4xlarge ノードの 1 時間の使用量は、1 つの db.r6g.large ノードの 16 時間の使用量に相当します。次の表は、ノードのサイズ別の正規化された単位の数を示しています。

ノードサイズ	正規化された単位 (Redis OSS)	正規化された単位 (Valkey)
small	1	7.
medium	2	1.4
large	4	2.8
xlarge	8	5.6
2xlarge	16	11.2
4xlarge	32	22.4
6xlarge	48	33.6
8xlarge	64	44.8
10xlarge	80	56
12xlarge	96	67.2
16xlarge	128	89.6
24xlarge	192	134.4

例えば、db.r6gd.xlarge のリザーブドノードを購入し、同じ AWS リージョンのアカウントで 2 つの db.r6gd.large リザーブドノードを実行しているとします。この場合、料金上の利点は両方のノードに全面的に適用されます。



または、同じリージョンのアカウントで実行している 1 つの AWS インスタンスがある場合、料金上の利点はこの リザーブドノードの使用の 50 パーセント に適用されます。



ノードを Redis OSS から Valkey にアップグレードする

MemoryDB における Valkey の提供開始により、Redis OSS リザーブドノードの割引を Valkey エンジンに適用できるようになりました。既存の契約や予約のメリットを維持しながら、Redis OSS から Valkey にアップグレードすることができます。ノードファミリーとエンジン内で利点を適用できるだけでなく、さらに追加的な価値を得ることもできます。Valkey は Redis OSS と比較して 30% の割引価格になっており、リザーブドノードの柔軟性により、Redis OSS リザーブドノードを使用して、より多くの実行中の Valkey ノードをカバーできます。

割引料金を計算するために、各 MemoryDB ノードとエンジンの組み合わせには、単位で測定される正規化係数があります。リザーブドノードの単位は、特定のエンジン用にリザーブドノードのインスタンスファミリー内で実行されているノードに適用できます。Redis OSS リザーブドノードは、実行中の Valkey ノードをカバーするためにエンジン全体に追加で適用できます。Valkey は Redis OSS に比べ割引価格で提供されるため、特定のインスタンスタイプの単位がより低く、Redis OSS リザーブドノードでより多くの Valkey ノードをカバーできます。

例えば、Redis OSS エンジン (32 単位) 用の db.r7g.4xlarge 用にリザーブドノードを購入し、1 つの db.r7g.4xlarge Redis OSS ノード (32 単位) を実行しているとします。ノードを Valkey にアップグレードすると、実行中のノードの正規化係数が 22.4 単位に減少し、既存のリザーブドノードにより、リージョンの db.r7g ファミリー内の他の実行中の Valkey または Redis OSS ノードに対して追加で 9.6 単位を使用できるようになります。これを使用して、アカウントの別の db.r7g.4xlarge Valkey ノード (22.4 単位) の 42%、または db.r7g.xlarge Valkey ノード (5.6 単位) の 100% および db.r7g.large Valkey ノード (2.8 単位) の 100% をカバーできます。

リザーブドノードの削除

リザーブドノードには 1 年契約と 3 年契約があります。リザーブドノードはキャンセルできません。ただし、リザーブドノードの割引対象である ノードは削除できます。リザーブドノードの割引対象である ノードの削除プロセスは、他のノードの削除プロセスと同じです。

リザーブドノードの割引対象である ノードを削除した場合、互換性がある仕様の別の ノードを起動できます。この場合、予約期間 (1 年または 3 年) 中、割引料金を利用できます。

リザーブドノードの操作

AWS マネジメントコンソール、AWS Command Line Interface、および MemoryDB API を使用して、リザーブドノードを使用できます。

コンソール

リザーブドノード提供タイプの料金表と情報を取得するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. ナビゲーションペインで、[リザーブドノード] を選択します。
3. [リザーブドノードの購入] を選択します。
4. [ノードタイプ] で、デプロイするノードのタイプを選択します。
5. [数量] には、デプロイするノードの数を選択します。
6. [期間] で、データベースノードを予約する期間を選択します。
7. 提供タイプ で、提供タイプを選択します。

これらの選択を行うと、[予約の概要] に料金情報が表示されます。

> **Important**
>
> これらのノードを購入して料金が発生することを防ぐには、[キャンセル] を選択します。

リザーブドノード提供タイプに関する情報を取得したら、次の手順に従い、この情報を使用して提供タイプを購入できます。

リザーブドノードを購入するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. ナビゲーションペインで、[リザーブドノード] を選択します。
3. [リザーブドノードの購入] を選択します。
4. [ノードタイプ] で、デプロイするノードのタイプを選択します。
5. [数量] には、デプロイするノードの数を選択します。
6. [期間] で、データベースノードを予約する期間を選択します。
7. 提供タイプ で、提供タイプを選択します。
8. 「オプショナル」 購入したリザーブドノードに独自の識別子を割り当てると、インスタンスを追跡しやすくなります。[予約 ID]に、リザーブドノードの識別子を入力します。

   これらの選択を行うと、[予約の概要] に料金情報が表示されます。
9. [リザーブドノードの購入] を選択します。
10. リザーブドノードが購入され、[リザーブドノード] リストに表示されます。

AWS アカウントのリザーブドノードの情報を入手するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. ナビゲーションペインで、[リザーブドノード] を選択します。
3. アカウントのリザーブドノードが表示されます。特定のリザーブドノードに関する詳細な情報を確認するには、リストにあるそのノードを選択します。これによって、そのノードに関する詳細情報を表示できます。

AWS Command Line Interface

次の例 describe-reserved-nodes-offerings では、リザーブドノードサービスの詳細を返します。

```
aws memorydb describe-reserved-nodes-offerings
```

これによって、次のような出力が生成されます。

```
{
    "ReservedNodesOfferings": [
        {
            "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
            "NodeType": "db.xxx.large",
            "Duration": 94608000,
            "FixedPrice": $xxx.xx,
            "OfferingType": "Partial Upfront",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": $xx.xx,
                    "RecurringChargeFrequency": "Hourly"
                }
            ]
        }
    ]
}
```

次のパラメータを渡して、返される内容の範囲を制限することもできます。

- --reserved-nodes-offering-id - 購入する提供タイプの ID。
- --node-type – ノードタイプのフィルタ値。このパラメータを使用すると、指定されたノードタイプと一致する予約のみが表示されます。
- --duration – 期間フィルタ値は年または秒単位で指定します。このパラメータを使用すると、この期間の予約のみが表示されます。
- --offering-type – このパラメータを使用すると、指定した提供タイプと一致する利用可能なオファリングみが表示されます。

リザーブドノード提供タイプに関する情報を取得したら、この情報を使用して提供タイプを購入できます。

次の例 purchase-reserved-nodes-offering では、新しいリザーブドノードを購入します

Linux、macOS、Unix の場合:

```
aws memorydb purchase-reserved-nodes-offering \

    --reserved-nodes-offering-id 0193cc9d-7037-4d49-b332-d5e984f1d8ca \
    --reservation-id reservation \
    --node-count 2
```

Windows の場合:

```
aws memorydb purchase-reserved-nodes-offering ^
    --reserved-nodes-offering-id 0193cc9d-7037-4d49-b332-d5e984f1d8ca ^
    --reservation-id MyReservation
```

- --reserved-nodes-offering-id 購入を提案しているリザーブドノードの名前を表します。
- --reservation-id はこの予約を追跡するユーザー指定識別子です。

> **Note**
>
> 予約 ID は、この予約を追跡するユーザー指定の一意識別子です。このパラメータが指定されていない場合、MemoryDB により予約の識別子が自動的に生成されます。

- --node-count は予約するノードの数です。デフォルトは 1 です。

これによって、次のような出力が生成されます。

```
{
    "ReservedNode": {
        "ReservationId": "reservation",
        "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
        "NodeType": "db.xxx.large",
        "StartTime": 1671173133.982,
        "Duration": 94608000,
        "FixedPrice": $xxx.xx,
        "NodeCount": 2,
```

```
        "OfferingType": "Partial Upfront",
        "State": "payment-pending",
        "RecurringCharges": [
            {
                "RecurringChargeAmount": $xx.xx,
                "RecurringChargeFrequency": "Hourly"
            }
        ],
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxx:reservednode/reservation"
    }
}
```

リザーブドノードを購入したら、リザーブドノードに関する情報を取得できます。

describe-reserved-nodes 次の例では、このアカウントのリザーブドノードに関する情報を返します。

```
aws memorydb describe-reserved-nodes
```

これによって、次のような出力が生成されます。

```
{
    "ReservedNodes": [
        {
            "ReservationId": "ri-2022-12-16-00-28-40-600",
            "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
            "NodeType": "db.xxx.large",
            "StartTime": 1671150737.969,
            "Duration": 94608000,
            "FixedPrice": $xxx.xx,
            "NodeCount": 1,
            "OfferingType": "Partial Upfront",
            "State": "active",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": $xx.xx,
                    "RecurringChargeFrequency": "Hourly"
                }
            ],
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxx:reservednode/
ri-2022-12-16-00-28-40-600"
```

```
        }
    ]
}
```

次のパラメータを渡して、返される内容の範囲を制限することもできます。

- `--reservation-id` - 購入したリザーブドノードに独自の識別子を割り当てると、インスタンスを追跡しやすくなります。
- `--reserved-nodes-offering-id` – オファリング識別子のフィルタ値。このパラメータを使用すると、指定されたオファリングIDと一致する購入済み予約のみが表示されます。
- `--node-type` – ノードタイプのフィルタ値。このパラメータを使用すると、指定されたノードタイプと一致する予約のみが表示されます。
- `--duration` – 期間フィルタ値は年または秒単位で指定します。このパラメータを使用すると、この期間の予約のみが表示されます。
- `--offering-type` – このパラメータを使用すると、指定した提供タイプと一致する利用可能なオファリングみが表示されます。

MemoryDB API

次の例では、リザーブドノードに「MemoryDB クエリ API」を使用する方法を示します。

DescribeReservedNodesOfferings

リザーブドノードサービスの詳細を返します。

```
https://memorydb.us-west-2.amazonaws.com/
    ?Action=DescribeReservedNodesOfferings
    &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f
 &"Duration": 94608000,
    &NodeType="db.r6g.large"
    &OfferingType="Partial Upfront"
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

次のパラメータは、返される内容の範囲を制限します。

- ReservedNodesOfferingId 購入を提案しているリザーブドノードの名前を表します。
- Duration – 期間フィルタ値は年または秒単位で指定します。このパラメータを使用すると、この期間の予約のみが表示されます。
- NodeType – ノードタイプのフィルタ値。このパラメータを使用すると、指定されたノードタイプと一致するオファリングのみが表示されます。
- OfferingType – このパラメータを使用すると、指定した提供タイプと一致する利用可能なオファリングみが表示されます。

リザーブドノード提供タイプに関する情報を取得したら、この情報を使用して提供タイプを購入できます。

PurchaseReservedNodesOffering

リザーブドノードサービスの購入を許可します。

```
https://memorydb.us-west-2.amazonaws.com/
    ?Action=PurchasedReservedNodesOffering
    &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f
    &ReservationID=myreservationID
    &NodeCount=1
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

- ReservedNodesOfferingId 購入を提案しているリザーブドノードの名前を表します。
- ReservationID はこの予約を追跡するユーザー指定識別子です。

> **Note**
>
> 予約 ID は、この予約を追跡するユーザー指定の一意識別子です。このパラメータが指定されていない場合、MemoryDB により予約の識別子が自動的に生成されます。

- NodeCount は予約するノードの数です。デフォルトは 1 です。

リザーブドノードを購入したら、リザーブドノードに関する情報を取得できます。

DescribeReservedNodes

このアカウントのリザーブドノードに関する情報を返します。

```
https://memorydb.us-west-2.amazonaws.com/
 ?Action=DescribeReservedNodes
 &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f
 &ReservationID=myreservationID
 &NodeType="db.r6g.large"
 &Duration=94608000
 &OfferingType="Partial Upfront"
 &Version=2021-01-01
 &SignatureVersion=4
 &SignatureMethod=HmacSHA256
 &Timestamp=20141201T220302Z
 &X-Amz-Algorithm
 &X-Amz-SignedHeaders=Host
 &X-Amz-Expires=20141201T220302Z
 &X-Amz-Credential=<credential>
 &X-Amz-Signature=<signature>
```

次のパラメータは、返される内容の範囲を制限します。

- ReservedNodesOfferingId はリザーブドノードの名前を表します。
- ReservationID - 購入したリザーブドノードに独自の識別子を割り当てると、インスタンスを追跡しやすくなります。
- NodeType – ノードタイプのフィルタ値。このパラメータを使用すると、指定されたノードタイプと一致する予約のみが表示されます。
- Duration – 期間フィルタ値は年または秒単位で指定します。このパラメータを使用すると、この期間の予約のみが表示されます。
- OfferingType – このパラメータを使用すると、指定した提供タイプと一致する利用可能なオファリングみが表示されます。

リザーブドノードの請求を表示

リザーブドノードの請求は、AWS マネジメントコンソールの「請求ダッシュボード」 で表示できます。

リザーブドノードの請求を表示する

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. コンソール上部の 検索 ボタンから 請求 を選択します。
3. ダッシュボードの左側から [請求書] を選択します。
4. [AWS サービス料] で [MemoryDB] を展開します。
5. リザーブドノードがある AWS リージョン [米国東部 (バージニア北部)] などを拡張します。

リザーブドノードと当月の時間単位の料金は、[Amazon MemoryDB クラスターの作成リザーブドインスタンス] に表示されます。

Amazon MemoryDB CreateCluster Reserved Instances		
AmazonMemoryDB, db.r6g.large reserved instance applied	81.000 Hrs	
AmazonMemoryDB, db.r6g.4xlarge reserved instance applied	324.000 Hrs	
AmazonMemoryDB, db.r6g.4xlarge reserved instance applied	162.000 Hrs	
USD (hourly fee per AmazonMemoryDB, db.r6g.large instance	1,488.000 Hrs	
USD (hourly fee per AmazonMemoryDB, db.r6gd.2xlarge instance	744.000 Hrs	
USD 0 hourly fee per AmazonMemoryDB, db.r6g.4xlarge instance	744.000 Hrs	
USD (hourly fee per AmazonMemoryDB, db.r6gd.xlarge instance	744.000 Hrs	
USD 1 hourly fee per AmazonMemoryDB, db.r6gd.4xlarge instance	2,976.000 Hrs	

ノードの置換

MemoryDB は、通常はシームレスに、頻繁にフリートのアップグレードを行います。ただし、場合によっては基盤となるホスト OS の必須更新を適用するために MemoryDB ノードを再起動する必要があります。セキュリティ、信頼性、運用パフォーマンスを強化するアップグレードを適用するため、そのような置換が必要となります。

このような交換を、スケジュールされたノード交換ウィンドウより前に、任意のタイミングで独自に管理することもできます。交換を独自に管理する場合、インスタンスはノードの再起動時に OS の更新を受信し、スケジュールされたノードの交換はキャンセルされます。ノード交換が行われることを示すアラートを引き続き受け取ることがあります。すでにメンテナンスの必要を手動で軽減した場合には、これらのアラートを無視できます。

> **Note**
>
> MemoryDB によって自動生成される交換ノードは、異なる IP アドレスを持つ場合があります。ノードを適切な IP アドレスに関連付けるようにアプリケーションが設定されていることを必ず確認してください。

以下のリストは、MemoryDB ノードの 1 つの交換をスケジュールしている場合に取ることのできるアクションを示しています。

MemoryDB ノードの交換オプション

- 何もしない - 何もしない場合、MemoryDB はスケジュールどおりにノードを交換します。

  ノードがマルチ AZ クラスターのメンバーである場合、MemoryDB によって、パッチ適用、更新、その他のメンテナンス関連のノード交換時の可用性が向上します。

  交換は、クラスターが受信した書き込みリクエストを処理する間に完了します。
- メンテナンスウィンドウを変更する – スケジュールされたメンテナンスイベントの場合、MemoryDB から E メールまたは通知イベントを受け取ります。これらの場合、スケジュールされた交換時間より前にメンテナンスウィンドウを変更すると、ノードは新しい時間に交換されます。詳細については、「MemoryDB クラスターの変更」を参照してください。

  > **Note**
  >
  > メンテナンスウィンドウを移動して置換ウィンドウを変更する機能は、MemoryDB 通知にメンテナンスウィンドウが含まれている場合にのみ使用できます。通知にメンテナンスウィンドウが含まれていない場合、交換ウィンドウを変更することはできません。

  例えば、現在が 11 月 9 日の木曜日の 15:00 で、次のメンテナンスウィンドウが 11 月 10 日金曜日の 17:00 であるとします。以下は、3 つのシナリオとその結果です。
  - メンテナンスウィンドウを金曜日の 16:00 に変更します (現在の日時以降で、次の予定メンテナンスウィンドウより前)。ノードは、11 月 10 日の金曜日の 16:00 に交換されます。
  - メンテナンスウィンドウを土曜日の 16:00 に変更します (現在の日時以降で、次の予定メンテナンスウィンドウ以降)。ノードは、11 月 11 日の土曜日の 16:00 に交換されます。
  - メンテナンスウィンドウを水曜日の 16:00 に変更します 同じ週内で、現在の日時より前。ノードは、11 月 15 日の水曜日の 16:00 に交換されます。

手順については、「メンテナンスの管理」を参照してください。

クラスターの管理

MemoryDB の多くのオペレーションは、クラスターレベルで実行されます。クラスターは、特定数のキャッシュノードと、各ノードのプロパティを制御するパラメータグループを使用して設定できます。クラスター内のすべてのノードは、同じノードタイプで、同一のパラメーター設定およびセキュリティグループ設定となるように設計されています。

すべてのクラスターにはクラスター識別子が必要です。クラスター識別子は、お客様が指定するクラスターの名前です。この識別子によって、MemoryDB API と AWS CLI コマンドを使用して操作するときに、特定のクラスターを指定します。クラスター識別子は、AWS リージョン内でお客様別に一意である必要があります。

MemoryDB クラスターは Amazon EC2 インスタンスを使用してアクセスするように設計されています。MemoryDBクラスターは、Amazon VPC サービスに基づく 仮想プライベートクラウド (VPC) でのみ起動できますが、 外部 AWS からアクセスできます。 詳細については、「外部からのMemoryDBリソースへのアクセス AWS」を参照してください。

データ階層化

r6gd ファミリーのノードタイプを使用するクラスターでは、データがメモリとローカル SSD (ソリッドステートドライブ) ストレージの間で階層化されます。データ階層化は、データをメモリに保存するだけでなく、各クラスターノードで低コストのソリッドステートドライブ (SSD) を利用して、Valkey および Redis OSS ワークロードに新しいコストパフォーマンスの高いオプションを提供します。他のノードタイプと同様に、r6gd ノードに書き込まれたデータは、マルチ AZ トランザクションログに永続的に保存されます。データ階層化は、データセット全体の最大 20% に定期的にアクセスするワークロードや、SSD 上のデータにアクセスする際に増加するレイテンシーを許容できるアプリケーションに最適です。

データ階層化を行うクラスターでは、MemoryDB は保存するすべての項目の最終アクセス時間をモニタリングします。使用可能なメモリ (DRAM) が完全に消費されると、MemoryDB は Least Recently Used (LRU) アルゴリズムを使用して、アクセス頻度の低い項目をメモリから SSD に自動的に移動します。その後、SSD 上のデータにアクセスすると、MemoryDB はリクエストを処理する前に自動的かつ非同期的にデータをメモリに戻します。データのサブセットにのみ定期的にアクセスするワークロードがある場合、データ階層化は容量を優れたコスト効率でスケールするための最適な方法となります。

データ階層化を使用する場合、キー自体は常にメモリに残り、LRU によってメモリとディスクの値の配置が制御されます。一般に、データ階層化を使用する際は、キーサイズを値のサイズよりも小さくすることをお勧めします。

データ階層化は、アプリケーションワークロードへのパフォーマンスの影響を最小限に抑えるように設計されています。例えば、500 バイトの文字列値を想定した場合に、メモリ内のデータに対する読み取りリクエストと比較すると、SSD に保存されたデータに対する読み取りリクエストには、通常、平均で 450 マイクロ秒のレイテンシーが生じることが予想されます。

最も大きいデータ階層化ノードサイズ (db.r6gd.8xlarge) では、単一の 500 ノードクラスターに最大 500 TB まで保存できます (1 つのリードレプリカを使用する場合は 250 TB)。データ階層化では、MemoryDB はノードあたり (DRAM) メモリの 19% をデータ以外の用途用に確保しています。データ階層化は、MemoryDB でサポートされているすべての Valkey および Redis OSS コマンドおよびデータ構造と互換性があります。この機能を使用するためのクライアント側の変更は必要ありません。

トピック

- ベストプラクティス
- データ階層化の制限

- データ階層化の料金
- データ階層化のモニタリング
- データ階層化の使用
- スナップショットからクラスターにデータを復元する

ベストプラクティス

推奨されるベストプラクティスを以下に示します：

- データ階層化は、データセット全体の最大 20% に定期的にアクセスするワークロードや、SSD 上のデータにアクセスする際に増加するレイテンシーを許容できるアプリケーションに最適です。
- データ階層化ノードで利用可能な SSD 容量を使用する場合は、値のサイズをキーサイズよりも大きくすることをお勧めします。値のサイズは 128 MB を超えることはできません。128 MB を超えると、ディスクに移動されません。DRAM と SSD の間で項目を移動すると、キーは常にメモリに残り、値だけが SSD 階層に移動されます。

データ階層化の制限

データ階層化には以下の制限があります。

- 使用するノードタイプは、`us-east-2`、`us-east-1`、`us-west-2`、`us-west-1`、`eu-west-1`、`eu-west-3`、`eu-central-1`、`ap-northeast-1`、`ap-southeast-1`、`ap-southeast-2`、`ap-south-1`、`ca-central-1`、`sa-east-1` のリージョンで使用できる r6gd ファミリーのものである必要があります。
- r6gd クラスターは、r6gd を使用しない限りスナップショットを別のクラスターに復元できません。
- データ階層化クラスターのスナップショットを Amazon S3 にエクスポートすることはできません。
- 分岐なしの保存はサポートされていません。
- データ階層化クラスター (r6gd ノードタイプを使用するクラスターなど) からデータ階層化を使用しないクラスター (r6g ノードタイプを使用するクラスターなど) へのスケーリングはサポートされていません。
- データ階層化では、`volatile-lru`、`allkeys-lru` および `noeviction` の maxmemory ポリシーのみがサポートされます。
- 128 MiB を超える項目は SSD に移動されません。

データ階層化の料金

R6g ノード (メモリのみ) と比較すると、R6gd ノードは総容量 (メモリ + SSD) が 5 倍あり、最大使用率で実行すると 60 % 以上のコスト削減を実現できます。詳細については、「MemoryDB の料金」を参照してください。

データ階層化のモニタリング

MemoryDB は、データ階層化を使用するクラスターのパフォーマンスをモニタリングするために特別に設計されたメトリクスを提供します。SSD と比較した DRAM 内の項目の比率をモニタリングするには、MemoryDB のメトリック で `CurrItems` メトリクスを使用できます。パーセンテージは `(CurrItems with Dimension: Tier = Memory * 100) / (CurrItems with no dimension filter)` のように計算できます。

設定されたエビクションポリシーで許可されている場合、メモリ内の項目の割合が 5% を下回ると、MemoryDB は項目のエビクションを開始します。エビクションなしのポリシーで設定されたノードでは、書き込みオペレーションはメモリ不足エラーを受け取ります。

メモリ内の項目のパーセンテージが 5% を下回った場合は、MemoryDB クラスターのスケーリングを検討することをお勧めします。詳細については、「MemoryDB のメトリック」のデータ階層化を使用する MemoryDB クラスターのメトリクスを参照してください。

データ階層化の使用

AWS マネジメントコンソール を使用したデータ階層化の使用

クラスターを作成する場合は、r6gd ファミリーから db.r6gd.xlarge などのノードタイプを選択し、データ階層化を使用します。 ノードタイプを選択すると、データ階層化が自動的に有効になります。

クラスター作成の詳細については、ステップ 2: クラスターを作成する を参照してください。

AWS CLI を使用したデータ階層化の有効化

AWS CLI を使用してクラスターを作成する場合は、r6gd ファミリーから db.r6gd.xlarge などのノードタイプを選択し、`--data-tiering` パラメータを設定してデータ階層化を使用します。

r6gd ファミリーからノードタイプを選択する際に、データ階層化をオプトアウトすることはできません。`--no-data-tiering` パラメータを設定すると、オペレーションは失敗します。

Linux、macOS、Unix の場合:

```
aws memorydb create-cluster \
   --cluster-name my-cluster \
   --node-type db.r6gd.xlarge \
   --engine valkey  \
   --acl-name my-acl \
   --subnet-group my-sg \
   --data-tiering
```

Windows の場合:

```
aws memorydb create-cluster ^
   --cluster-name my-cluster ^
   --node-type db.r6gd.xlarge ^
   --engine valkey ^
   --acl-name my-acl ^
   --subnet-group my-sg
   --data-tiering
```

このオペレーションを実行すると、以下のようなレスポンスが表示されます。

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "creating",
        "NumberOfShards": 1,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Port": 6379
        },
        "NodeType": "db.r6gd.xlarge",
        "EngineVersion": "7.2",
        "EnginePatchVersion": "7.2.6",
        "Engine": "valkey"
        "ParameterGroupName": "default.memorydb-valkey7",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxxxx:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
```

```
        "SnapshotWindow": "04:30-05:30",
        "ACLName": "my-acl",
        "DataTiering":"true",
        "AutoMinorVersionUpgrade": true
    }
}
```

スナップショットからクラスターにデータを復元する

(コンソール)、(AWS CLI) または (MemoryDB API) を使用して、データ階層化を有効にした新しいクラスターにスナップショットを復元できます。r6gd ファミリーのノードタイプを使用してクラスターを作成すると、データ階層化が有効になります。

データ階層化を有効にして、スナップショットからクラスターにデータを復元する (コンソール)

データ階層化を有効にして新しいクラスターにスナップショットを復元するには (コンソール)「スナップショットからの復元 (コンソール)」の手順に従います。

データ階層化を有効にするには r6gd ファミリーからノードタイプを選択する必要があることに注意してください。

データ階層化を有効にして、バックアップからクラスターにデータを復元する AWS

AWS CLI を使用してクラスターを作成する場合、db.r6gd.xlarge などの r6gd ファミリーからノードタイプを選択し、`--data-tiering` パラメータを設定することで、デフォルトでデータ階層化が使用されます。

r6gd ファミリーからノードタイプを選択する際に、データ階層化をオプトアウトすることはできません。`--no-data-tiering` パラメータを設定すると、オペレーションは失敗します。

Linux、macOS、Unix の場合:

```
aws memorydb create-cluster \
   --cluster-name my-cluster \
   --node-type db.r6gd.xlarge \
   --engine valkey
   --acl-name my-acl \
   --subnet-group my-sg \
   --data-tiering \
   --snapshot-name my-snapshot
```

Windows の場合:

```
aws memorydb create-cluster ^
   --cluster-name my-cluster ^
   --node-type db.r6gd.xlarge ^
   --engine valkey ^
   --acl-name my-acl ^
   --subnet-group my-sg ^
   --data-tiering ^
   --snapshot-name my-snapshot
```

このオペレーションを実行すると、以下のようなレスポンスが表示されます。

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "creating",
        "NumberOfShards": 1,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Port": 6379
        },
        "NodeType": "db.r6gd.xlarge",
        "EngineVersion": "7.2",
        "EnginePatchVersion": "7.2.6",
        "Engine": "valkey"
        "ParameterGroupName": "default.memorydb-valkey7",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxxxx:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "ACLName": "my-acl",
        "DataTiering": "true"
}
```

クラスターを準備する

MemoryDB コンソール、AWS CLI、または MemoryDB API を使用してクラスターを作成する手順を以下に示します。

クラスターを作成するときは常に、すぐにアップグレードまたは変更が必要にならないように、何らかの準備作業をしておくことが推奨されます。

トピック

- 要件の特定

要件の特定

準備

以下の質問に対する回答を確認することで、クラスターの作成を円滑化できます。

- クラスターの作成を開始する前に、必ず同じ VPC にサブネットグループを作成してください。または、提供されているデフォルトのサブネットグループを使用できます。詳細については、「サブネットおよびサブネットグループ」を参照してください。

  MemoryDB は、Amazon EC2 を使用して AWS 内からアクセスからアクセスするためのものです。ただし、Amazon VPC に基づく VPC で起動すれば、AWS からアクセスを提供できます。 詳細については、「外部からのMemoryDBリソースへのアクセス AWS」を参照してください。
- パラメーター値をカスタマイズする必要がありますか。

  その場合、カスタムパラメータグループを作成します。詳細については、「パラメータグループを作成する」を参照してください。
- VPC セキュリティグループを作成する必要がありますか？

  詳細については、「VPC のセキュリティ」を参照してください。
- 耐障害性をどのようにして導入しますか。

  詳細については、「障害の軽減」を参照してください。

トピック

- メモリとプロセッサの要件
- MemoryDB クラスターの構成
- 強化された I/O マルチプレクシング
- スケーリングの要件
- アクセスの要件

- リージョンとアベイラビリティーゾーン

メモリとプロセッサの要件

MemoryDB の基本的な構成ブロックはノードです。ノードはシャード状に構成され、クラスターを形成します。クラスターに使用するノードタイプを決定するときは、クラスターのノード構成および保存する必要があるデータの量を考慮する必要があります。

MemoryDB クラスターの構成

MemoryDB クラスターは、1～500 個のシャードで構成されます。MemoryDB クラスター内のデータは、クラスターのシャード間に分割されます。アプリケーションは、エンドポイントと呼ばれるネットワークアドレスを使用して MemoryDB クラスターに接続します。ノードエンドポイントに加えて、MemoryDB クラスター自体にはクラスターエンドポイントと呼ばれるエンドポイントがあります。 アプリケーションはこのエンドポイントを使用してクラスターの読み取りまたは書き込みを行うことができ、どのノードに対して読み取りまたは書き込みを行うかの判断は MemoryDB に任せることができます。

強化された I/O マルチプレクシング

Valkey または Redis OSS バージョン 7.0 以降を実行している場合、拡張 I/O 多重化によってさらに高速化されます。この機能では、各専用ネットワーク IO スレッドが複数のクライアントからのコマンドをエンジンにパイプラインし、コマンドをバッチ処理する効率的な機能を活用します。詳細については、「超高速パフォーマンス」と「the section called “サポートされているノードの種類”」を参照してください。

スケーリングの要件

クラスターはすべて、より大きなノードタイプにスケールアップできます。MemoryDB クラスターをスケールアップする場合、クラスターを引き続き使用できるようにオンラインでスケールアップすることも、スナップショットから新しいクラスターをシードして、新しいクラスターが最初から空になるのを防ぐこともできます。

詳細については、このガイドの「Scaling (スケーリング) 」を参照してください。

アクセスの要件

設計上、MemoryDB クラスターは Amazon EC2 インスタンスからアクセスします。MemoryDB クラスターへのネットワークアクセスは、クラスターを作成したアカウントに制限されます。したがっ

て、Amazon EC2 インスタンスからクラスターにアクセスするには、クラスターへのアクセスを許可する必要があります。詳細な手順については、このガイドの「ステップ 3: クラスターへのアクセスの許可」を参照してください。

リージョンとアベイラビリティーゾーン

アプリケーションに近い リージョンに AWS MemoryDB クラスターを配置することで、レイテンシーを低減できます。クラスターに複数のノードがある場合、複数の異なるアベイラビリティーゾーンにノードを配置することで、クラスター上の障害の影響を低減できます。

詳細については次を参照してください:

- リージョンとアベイラビリティーゾーンの選択
- 障害の軽減

クラスターの作成

MemoryDB は、クラスターを作成するための 3 つの方法を提供します。詳細については、「ステップ 2: クラスターを作成する」を参照してください。

クラスターの詳細を表示する

MemoryDB コンソール、AWS CLI、または MemoryDB API を使用して、1 つ以上のクラスターについての詳細を表示できます。

MemoryDB クラスターの詳細の表示 (コンソール)

次の手順は、MemoryDB コンソールを使用して MemoryDB クラスターの詳細を表示する方法を示しています。

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. クラスターの詳細を表示するには、クラスター名の左側にあるラジオボタンを選択し、[詳細の表示] を選択します。クラスターを直接クリックして、クラスターの詳細ページを表示することもできます。

   [クラスターの詳細] ページには、クラスターエンドポイントを含むクラスターの詳細が表示されます。[クラスターの詳細] ページにある複数のタブを使用して、詳細を表示できます。
3. クラスター内のシャード数とシャードごとのノード数を一覧表示するには、[シャードとノード] タブを選択します。
4. ノードに関する特定の情報を表示するには、以下の表のシャードを展開してください。または、検索ボックスを使用してシャードを検索できます。

   これにより、アベイラビリティーゾーン、スロット/キースペース、ステータスなど、各ノードに関する情報が表示されます。
5. [メトリクス] タブを選択すると、[CPU 使用率] や [エンジン CPU 使用率] など、それぞれのプロセスを監視できます。詳細については、「MemoryDB のメトリック」を参照してください。
6. [ネットワークとセキュリティ] タブを選択すると、サブネットグループとセキュリティグループの詳細が表示されます。

   a. [サブネットグループ] には、サブネットグループの名前、サブネットが属する VPC へのリンク、サブネットグループの Amazon リソースネーム (ARN) が表示されます。

   b. [セキュリティグループ] では、セキュリティグループ ID、名前、説明を確認できます。
7. [メンテナンスとスナップショット] タブを選択すると、スナップショット設定の詳細が表示されます。

a. [スナップショット] では、自動スナップショットが有効かどうか、スナップショットの保持期間、スナップショットウィンドウを確認できます。

b. [スナップショット] には、スナップショットの名前、サイズ、シャード数、ステータスなど、このクラスターのすべてのスナップショットのリストが表示されます。

詳細については、「スナップショットおよび復元」を参照してください。

8. [メンテナンスとスナップショット] タブを選択すると、メンテナンスウィンドウの詳細と、保留中の ACL、リシャーディング、またはサービスのアップデートが表示されます。詳細については、「メンテナンスの管理」を参照してください。
9. [サービスの更新] タブを選択すると、このクラスターに適用されるすべてのサービスアップデートの詳細が表示されます。詳細については、「MemoryDB のサービスの更新」を参照してください。
10. [タグ] タブを選択すると、このクラスターに関連付けられているすべてのリソースまたはコスト配分タグの詳細が表示されます。詳細については、「スナップショットのタグ付け」を参照してください。

クラスターの詳細の表示 AWS

AWS CLI `describe-clusters` コマンドを使用してクラスターの詳細を表示できます。`--cluster-name` パラメーターを省略すると、最大で `--max-results` のクラスターの詳細が返されます。`--cluster-name` パラメータが含まれる場合は、指定したクラスターの詳細が返されます。`--max-results` パラメーターで返されるレコード数を制限できます。

次のコードは `my-cluster` の詳細を一覧します。

```
aws memorydb describe-clusters --cluster-name my-cluster
```

次のコードは最大で 25 のクラスターの詳細を一覧します。

```
aws memorydb describe-clusters --max-results 25
```

Example

Linux、macOS、Unix の場合:

```
aws memorydb describe-clusters \
```

```
    --cluster-name my-cluster \
    --show-shard-details
```

Windows の場合:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster ^
    --show-shard-details
```

次の JSON 出力は応答を示しています。

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Description": "my cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": 1629230643.961,
                            "Endpoint": {
                                "Address": "my-cluster-0001-001.my-
cluster.abcdef.memorydb.us-east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "CreateTime": 1629230644.025,
                            "Endpoint": {
                                "Address": "my-cluster-0001-002.my-
cluster.abcdef.memorydb.us-east-1.amazonaws.com",
                                "Port": 6379
```

```
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.abcdef.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "default",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:000000000:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "sat:06:30-sat:07:30",
            "SnapshotWindow": "04:00-05:00",
            "ACLName": "open-access",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true,
        }
```

詳細については、「AWS CLI for MemoryDB トピック describe-clusters」を参照してください。

クラスターの詳細を表示する（MemoryDB API）

MemoryDB API DescribeClusters アクションを使用してクラスターの詳細を表示できます。ClusterName パラメータが含まれる場合は、指定したクラスターの詳細が返されます。ClusterName パラメータを省略すると、最大で MaxResults (デフォルトは 100) のクラスターの詳細が返されます。MaxResults の値は 20 未満、または 100 を超えることはできません。

次のコードは my-cluster の詳細を一覧します。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeClusters
   &ClusterName=my-cluster
```

```
   &Version=2021-01-01
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

次のコードは最大で 25 のクラスターの詳細を一覧します。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeClusters
   &MaxResults=25
   &Version=2021-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

詳細については、「MemoryDB API リファレンストピック DescribeClusters」を参照してください。

MemoryDB クラスターの変更

クラスターへのノードの追加または削除以外にも、セキュリティグループの追加、メンテナンスウィンドウやパラメータグループの変更など、既存のクラスターに他の変更をかける必要がある場合があります。

メンテナンスウィンドウは使用率の最も低い時間帯に設定することをお勧めします。このため、場合によっては変更が必要になります。

クラスターのパラメータを変更すると、その変更は即座にクラスターに適用されます。これは、クラスターのパラメータグループ自体を変更するか、クラスターのパラメータグループ内のパラメータ値を変更するかに関係なく当てはまります。

クラスターのエンジンバージョンを更新することもできます。例えば、新しいエンジンのマイナーバージョンを選択すると、MemoryDB は直ちにクラスターの更新を開始します。

の使用AWS マネジメントコンソール

クラスターを変更するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 右上のリストから、変更するクラスターが存在する AWS リージョンを選択します。
3. 左側のナビゲーションから [クラスター] に移動します。[クラスターの詳細] から、ラジオボタンを使用してクラスターを選択し、[アクション]、[変更] の順に移動します。
4. [変更] ページが表示されます。
5. [クラスターの変更] ウィンドウで、必要な変更を行います。オプションには以下が含まれます。

   - 説明
   - サブネットグループ
   - VPC セキュリティグループ
   - ノードの種類

> **Note**
>
> クラスターが r6gd ファミリーのノードタイプを使用している場合は、そのファミリー内からのみ別のノードサイズを選択できます。r6gd ファミリーからノードタイプ

を選択すると、データ階層化が自動的に有効になります。詳細については、「データ階層化」を参照してください。

- Valkey または Redis OSS バージョンの互換性
- 自動スナップショットを有効にする
- スナップショットの保持期間
- スナップショットウィンドウ
- メンテナンスウィンドウ
- SNS 通知のトピック

6. Save changes (変更の保存) をクリックします。

[クラスターの詳細] ページに移動し、[変更] をクリックしてクラスターを変更することもできます。クラスターの特定のセクションを変更したい場合は、[クラスター詳細] ページの該当するタブに移動し、[変更]をクリックします。

の使用AWS CLI

AWS CLI `update-cluster` オペレーションを使用して既存のクラスターを変更できます。クラスターの設定値を変更するには、クラスターの ID、変更するパラメータ、パラメータの新しい値を指定します。次の例では、`my-cluster` という名前のクラスターのメンテナンスウィンドウを変更し、変更内容を直ちに適用します。

Linux、macOS、Unix の場合:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --preferred-maintenance-window sun:23:00-mon:02:00
```

Windows の場合:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --preferred-maintenance-window sun:23:00-mon:02:00
```

詳細については、「AWS CLI コマンドリファレンス」の「update-cluster」を参照してください。

MemoryDB API の使用

MemoryDB API UpdateCluster オペレーションを使用して既存のクラスターを変更できます。 クラスターの設定値を変更するには、クラスターの ID、変更するパラメータ、パラメータの新しい値を指定します。次の例では、my-cluster という名前のクラスターのメンテナンスウィンドウを変更し、変更内容を直ちに適用します。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &ClusterName=my-cluster
    &PreferredMaintenanceWindow=sun:23:00-mon:02:00
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210802T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Redis OSS から Valkey へのクロスエンジンアップグレードをトリガーする方法

コンソール、API、または CLI を使用して、既存の Redis OSS クラスターを Valkey エンジンにアップグレードできます。

デフォルトのパラメータグループを使用している既存の Redis OSS クラスターがある場合は、update-cluster API で新しいエンジンとエンジンバージョンを指定して、Valkey にアップグレードできます。

Linux、macOS、Unix の場合:

```
aws memorydb update-cluster \
   --cluster-name myCluster \
   --engine valkey \
   --engine-version 7.2
```

Windows の場合:

```
aws memorydb update-cluster ^
```

```
   --cluster-name myCluster ^
   --engine valkey ^
   --engine-version 7.2
```

アップグレードする既存の Redis OSS クラスターにカスタムパラメータグループが適用されている場合は、リクエストでカスタム Valkey パラメータグループも渡す必要があります。入力された Valkey カスタムパラメータグループは、既存の Redis OSS カスタムパラメータグループと同じ Redis OSS 静的パラメータ値を持っている必要があります。

Linux、macOS、Unix の場合:

```
aws memorydb update-cluster \
   --cluster-name myCluster \
   --engine valkey \
   --engine-version 7.2 \
   --parameter-group-name myParamGroup
```

Windows の場合:

```
aws memorydb update-cluster ^
   --cluster-name myCluster ^
   --engine valkey ^
   --engine-version 7.2 ^
   --parameter-group-name myParamGroup
```

クラスターからのノードの追加/削除

AWS マネジメントコンソール、AWS CLI、または MemoryDB API を使用して、クラスターにノードを追加または削除できます。

の使用AWS マネジメントコンソール

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. クラスターの一覧から、ノードを追加または削除するクラスターの名前を選択します。
3. [シャードとノード] タブで、[ノードの追加/削除] を選択します
4. [新しいノード数] に必要なノードの数を入力します。
5. 確認 を選択します。

> **Important**
>
> ノード数を 1 に設定すると、マルチ AZ は有効ではなくなります。[自動フェイルオーバー] を有効にすることもできます。

の使用AWS CLI

1. 削除するノードの名前を確認します。詳細については、「クラスターの詳細を表示する」を参照してください。
2. 次の例のように、削除するノードの一覧を使用して `update-cluster` CLI オペレーションを呼び出します。

   コマンドラインインターフェイスを使用してクラスターからノードを削除するには、以下のパラメーターを指定して `update-cluster` コマンドを使用します。

   - `--cluster-name` ノードを削除するクラスターの ID。
   - `--replica-configuration` – レプリカの数を設定できます。
     - `ReplicaCount` – レプリカノードの数を指定するには、このプロパティを設定します。
   - `--region` 複数のノードを削除するクラスターの AWS リージョンを指定します。

   Linux、macOS、Unix の場合:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --replica-configuration \
        ReplicaCount=1 \
    --region us-east-1
```

Windows の場合:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --replica-configuration ^
        ReplicaCount=1 ^
    --region us-east-1
```

詳細については、「AWS CLI のトピック update-cluster」を参照してください。

MemoryDB API の使用

MemoryDB API を使用してノードを削除するには、次のようにクラスター名と削除するノードの一覧を使用して UpdateCluster API オペレーションを呼び出します。

- ClusterName ノードを削除するクラスターの ID。
- ReplicaConfiguration – レプリカの数を設定できます。
  - ReplicaCount – レプリカノードの数を指定するには、このプロパティを設定します。
- Region 1 つのノードを削除するクラスターの AWS リージョンを指定します。

詳細については、「クラスターの更新」を参照してください。

クラスターへのアクセス

MemoryDB インスタンスは、Amazon EC2 インスタンスを介してアクセスするように設計されています。

同じ Amazon VPC 内の Amazon EC2 インスタンスから MemoryDB ノードにアクセスできます。または、VPC ピアリングを使用して、異なる Amazon VPC 内の Amazon EC2 から MemoryDB ノードにアクセスできます。

トピック

- すべてのクラスターに対するアクセスを許可する
- 外部からのMemoryDBリソースへのアクセス AWS

すべてのクラスターに対するアクセスを許可する

同じ Amazon VPC で実行されている Amazon EC2 インスタンスからのみ MemoryDB クラスターに接続できます。この場合、クラスターに対するネットワーク進入を許可する必要があります。

Amazon VPC セキュリティグループからクラスターへのネットワーク進入を許可するには

1. AWS マネジメントコンソール にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ネットワークとセキュリティ] の下にある [セキュリティグループ] を選択します。
3. セキュリティグループのリストから、Amazon VPC のセキュリティグループを選択します。MemoryDB 用のセキュリティグループを作成した場合を除き、このセキュリティグループは、デフォルトという名前になります。
4. Inbound タブを選択し、次の操作を行います。
    a. Edit (編集) を選択します。
    b. ルールの追加 を選択します。
    c. Type 列で Custom TCP rule を選択します。
    d. Port range ボックスに、クラスターノードのポート番号を入力します。この番号は、クラスターの起動時に指定した番号と同じ番号である必要があります。Valkey と Redis OSS の両方のデフォルトポートは **6379**です。

e. [ソース] ボックスで [任意の場所] を選択します。ポート範囲が 0.0.0.0/0 になるため、Amazon VPC 内で起動したすべての Amazon EC2 インスタンスを MemoryDB ノードに接続できます。

> **Important**
>
> MemoryDBクラスターを0.0.0.0/0にオープンしても、クラスターはパブリックIPアドレスを持たないためインターネットに公開されず、VPC外からアクセスすることはできません。ただし、お客様のアカウントの他の Amazon EC2 インスタンスにデフォルトのセキュリティグループが適用され、そのインスタンスにパブリック IP アドレスが付与される場合があります。それがデフォルトポートで何かを実行している場合、そのサービスが意図せず公開されることがあります。そのため、MemoryDB専用にVPCセキュリティグループを作成することをお勧めします。詳細については、「カスタムセキュリティグループ」を参照してください。

f. Save (保存) を選択します。

Amazon VPCにAmazon EC2インスタンスを起動すると、そのインスタンスはMemoryDBクラスターに接続できるようになります。

外部からのMemoryDBリソースへのアクセス AWS

MemoryDBは、VPC内部で使用するために設計されたサービスです。インターネットトラフィックの遅延やセキュリティ上の懸念により、外部アクセスは推奨されません。ただし、テストや開発目的でMemoryDBへの外部アクセスが必要な場合は、VPNを介してアクセスすることができます。

AWS クライアントVPNを使用することで、MemoryDBノードへの外部アクセスを許可することができます：

- 承認されたユーザーまたは認証キーへのアクセスの制限
- VPN クライアントと AWS VPN エンドポイント間のトラフィックの暗号化
- 特定のサブネットまたはノードへのアクセスの制限
- ユーザーまたは認証キーからのアクセスの容易な取り消し
- 接続の監査

次に、以下の方法について手順を示します。

トピック

- 認証局の作成
- AWS クライアント VPN コンポーネントの設定
- VPN クライアントの設定

認証局の作成

認証局 (CA) は、さまざまな手法やツールを使用して作成できます。ここでは、OpenVPN プロジェクトが提供する easy-rsa ユーティリティをお勧めします。選択するオプションにかかわらず、キーは安全に保管してください。次の手順では、easy-rsa スクリプトをダウンロードし、最初の VPN クライアントを認証するための認証局とキーを作成します。

- 初期証明書を作成するには、ターミナルを開き、次の操作を行います。
  - `git clone` https://github.com/OpenVPN/easy-rsa
  - `cd easy-rsa`
  - `./easyrsa3/easyrsa init-pki`
  - `./easyrsa3/easyrsa build-ca nopass`
  - `./easyrsa3/easyrsa build-server-full server nopass`

- `./easyrsa3/easyrsa build-client-full client1.domain.tld nopass`

  証明書を含む pki サブディレクトリが easy-rsa の下に作成されます。

- サーバー証明書を AWS Certificate Manager (ACM) に送信します。
  - ACM コンソールで、Certificate Manager を選択します。
  - 証明書のインポート を選択します。
  - `easy-rsa/pki/issued/server.crt` ファイルにあるパブリックキー証明書を 証明書本文 フィールドに入力します。
  - `easy-rsa/pki/private/server.key` にあるプライベートキーを 証明書のプライベートキー フィールドに貼り付けます。`BEGIN AND END PRIVATE KEY` 間にあるすべての行 (`BEGIN` 行と `END` 行を含む) を選択してください。
  - `easy-rsa/pki/ca.crt` ファイルにある CA パブリックキーを 証明書チェーン フィールドに貼り付けます。
  - レビューとインポート を選択します。
  - インポート を選択します。

  AWS CLI を使用してサーバーの証明書を ACM に送信するには、次のコマンドを実行します:
  `aws acm import-certificate --certificate fileb://easy-rsa/pki/issued/server.crt --private-key file://easy-rsa/pki/private/server.key --certificate-chain file://easy-rsa/pki/ca.crt --region region`

  後で使用するために証明書 ARN を書き留めます。

AWS クライアント VPN コンポーネントの設定

AWS コンソールを使用する

AWS コンソールで、サービス、VPC の順に選択します。

仮想プライベートネットワーク で、クライアント VPN エンドポイント を選択し、次の操作を行います。

AWS クライアント VPN コンポーネントの設定

- クライアント VPN エンドポイントの作成 を選択します。
- 以下のオプションを指定します。

- クライアント IPv4 CIDR: /22 以上の範囲のネットマスクを持つプライベートネットワークを使用します。選択したサブネットが VPC ネットワークのアドレスと競合していないことを確認します。例: 10.0.0.0/22。
- サーバー証明書 ARN で、以前にインポートした証明書の ARN を選択します。
- 相互認証の使用 を選択します。
- クライアント証明書 ARN で、以前にインポートした証明書の ARN を選択します。
- クライアント VPN エンドポイントの作成 を選択します。

の使用AWS CLI

次のコマンドを実行してください。

```
aws ec2 create-client-vpn-endpoint --client-cidr-block
"10.0.0.0/22" --server-certificate-arn arn:aws:acm:us-
east-1:012345678912:certificate/0123abcd-ab12-01a0-123a-123456abcdef --
authentication-options Type=certificate-
authentication,,MutualAuthentication={ClientRootCertificateChainArn=arn:aws:acm:
east-1:012345678912:certificate/123abcd-ab12-01a0-123a-123456abcdef} --
connection-log-options Enabled=false
```

出力例:

```
"ClientVpnEndpointId": "cvpn-endpoint-0123456789abcdefg",
"Status": { "Code": "pending-associate" }, "DnsName": "cvpn-
endpoint-0123456789abcdefg.prod.clientvpn.us-east-1.amazonaws.com" }
```

ターゲットネットワークと VPN エンドポイントの関連付け

- 新しい VPN エンドポイントを選択し、関連付け タブを選択します。
- 関連付け を選択し、以下のオプションを指定します。
  - [VPC]: メモリ DB クラスターの VPC を選択します。
  - MemoryDB クラスターのネットワークの 1 つを選択します。不確かな場合は、MemoryDB ダッシュボードの [サブネットグループ] でネットワークを確認します。
  - 関連付け を選択します。必要に応じて、残りのネットワークについても同じ手順を繰り返します。

の使用AWS CLI

次のコマンドを実行してください。

```
aws ec2 associate-client-vpn-target-network --client-vpn-endpoint-id cvpn-
endpoint-0123456789abcdefg --subnet-id subnet-0123456789abdcdef
```

出力例:

```
"Status": { "Code": "associating" }, "AssociationId": "cvpn-
assoc-0123456789abdcdef" }
```

VPN セキュリティグループの確認

VPN エンドポイントは、VPC のデフォルトのセキュリティグループを自動的に採用します。インバウンドルールとアウトバウンドルールを確認し、セキュリティグループがサービスポート (デフォルトではRedisの6379) でVPNネットワーク (VPNエンドポイント設定で定義) からMemoryDBネットワークへのトラフィックを許可しているかどうかを確認します。

VPN エンドポイントに割り当てられたセキュリティグループを変更する必要がある場合は、次の手順を実行します。

- 現在のセキュリティグループを選択します。
- セキュリティグループの適用 を選択します。
- 新しいセキュリティグループを選択します。

の使用AWS CLI

次のコマンドを実行してください。

```
aws ec2 apply-security-groups-to-client-vpn-target-network --
client-vpn-endpoint-id cvpn-endpoint-0123456789abcdefga  --vpc-id
vpc-0123456789abdcdef --security-group-ids sg-0123456789abdcdef
```

出力例:

```
"SecurityGroupIds": [ "sg-0123456789abdcdef" ] }
```

> **Note**
>
> MemoryDBセキュリティグループは、VPNクライアントからのトラフィックも許可する必要がある。クライアントのアドレスは、VPC ネットワークに従って VPN エンドポイントアド

レスでマスクされます。したがって、MemoryDBセキュリティグループの受信ルールを作成する際は、VPCネットワーク（VPNクライアントのネットワークではない）を考慮してください。

宛先ネットワークへの VPN アクセスの許可

[認証] タブで、[受信の承認]を選択し、以下を指定します。

- アクセスを許可する宛先ネットワーク： 0.0.0.0/0を使用してあらゆるネットワーク（インターネットを含む）へのアクセスを許可するか、MemoryDBのネットワーク/ホストを制限する。
- アクセスを付与する対象 で、すべてのユーザーにアクセスを許可する を選択します。
- [認可ルールの追加] を選択します。

の使用AWS CLI

次のコマンドを実行してください。

```
aws ec2 authorize-client-vpn-ingress --client-vpn-endpoint-id cvpn-endpoint-0123456789abcdefg --target-network-cidr 0.0.0.0/0 --authorize-all-groups
```

出力例:

```
{ "Status": { "Code": "authorizing" } }
```

VPN クライアントからインターネットへのアクセスの許可

VPN 経由でインターネットをブラウズする必要がある場合は、追加のルートを作成する必要があります。ルートテーブル タブを選択し、ルートの作成 を選択します。

- ルート送信先: 0.0.0.0/0
- ターゲット VPC サブネット ID: インターネットにアクセスできる、関連付けられたサブネットの 1 つを選択します。
- ルートの作成 を選択します。

の使用AWS CLI

次のコマンドを実行してください。

```
aws ec2 create-client-vpn-route --client-vpn-endpoint-id cvpn-
endpoint-0123456789abcdefg --destination-cidr-block 0.0.0.0/0 --target-vpc-
subnet-id subnet-0123456789abdcdef
```

出力例:

```
{ "Status": { "Code": "creating" } }
```

VPN クライアントの設定

AWS クライアント VPN ダッシュボードで、最近作成した VPN エンドポイントを選択し、クライアント設定のダウンロード を選択します。設定ファイル、easy-rsa/pki/issued/client1.domain.tld.crt ファイル、および easy-rsa/pki/private/client1.domain.tld.key ファイルをコピーします。設定ファイルを編集し、以下のパラメータを変更または追加します。

- cert: client1.domain.tld.crt ファイルを指すパラメータ cert を使用して新しい行を追加します。ファイルへの完全なパスを使用します。例:cert /home/user/.cert/client1.domain.tld.crt
- cert: key: client1.domain.tld.key ファイルを指すパラメータ key を使用して新しい行を追加します。ファイルへの完全なパスを使用します。例:key /home/user/.cert/client1.domain.tld.key

コマンド sudo openvpn --config downloaded-client-config.ovpnを使用して VPN 接続を確立します。

アクセスの取り消し

特定のクライアントキーからのアクセスを無効にする必要がある場合は、CA でキーを取り消します。次に、取り消しリストを AWS クライアント VPN に送信します。

easy-rsa でキーを取り消す方法は次のとおりです。

- cd easy-rsa
- ./easyrsa3/easyrsa revoke client1.domain.tld
- 続行するには、「yes」と入力します。中止するには、その他を入力します。

  ```
  Continue with revocation: `yes` ... * `./easyrsa3/easyrsa gen-crl
  ```
- 更新された CRL が作成されます。CRL ファイル: /home/user/easy-rsa/pki/crl.pem

AWS クライアント VPN への取り消しリストのインポート:

- AWS マネジメントコンソール で、サービス、VPC の順に選択します。
- クライアント VPN エンドポイント を選択します。
- クライアント VPN エンドポイントを選択し、アクション、クライアント証明書 CRL のインポート の順に選択します。
- crl.pem ファイルの内容を貼り付けます。

の使用AWS CLI

次のコマンドを実行してください。

```
aws ec2 import-client-vpn-client-certificate-revocation-list --certificate-revocation-list file://./easy-rsa/pki/crl.pem --client-vpn-endpoint-id cvpn-endpoint-0123456789abcdefg
```

出力例:

```
Example output: { "Return": true }
```

接続エンドポイントの検索

エンドポイントを使用してアプリケーションがクラスターに接続します。エンドポイントはクラスターの一意のアドレスです。クラスターのエンドポイントをすべてのオペレーションに使用します。

以下のセクションで、必要なエンドポイントの検索について説明します。

(AWS マネジメントコンソール) MemoryDB クラスターのエンドポイントの検索

MemoryDB クラスターのエンドポイントを検索するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. ナビゲーションペインで、クラスター を選択します。

   クラスターの一覧が表示されています。接続するクラスターを選択します。
3. クラスターのエンドポイントを検索するには、クラスターの名前を選択します。
4. [設定エンドポイント] は [クラスターの詳細] の下に表示されます。コピーするには、エンドポイントの左側にある (コピー) アイコンを選択します。

(AWS CLI) MemoryDB クラスターのエンドポイントの検索

`describe-clusters` コマンドを使用して、クラスターのエンドポイントを検出できます。 このコマンドは、クラスターのエンドポイントを返します。

次の操作は、クラスター `mycluster` のエンドポイント (この例では`####`です) を取得します。

以下の JSON コードを返します。

```
aws memorydb describe-clusters \
  --cluster-name mycluster
```

Windows の場合:

```
aws memorydb describe-clusters ^
   --cluster-name mycluster
```

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
```

```
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.4",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:zzzexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

詳細については、「describe-clusters」を参照してください。

MemoryDB クラスターのエンドポイントを検索する (MemoryDB API)

MemoryDB API を使用して、クラスターのエンドポイントを検出できます。

MemoryDB クラスターのエンドポイントを検索する (MemoryDB API)

MemoryDB API を使用して DescribeClusters アクションでクラスターのエンドポイントを検出することができます。アクションは、クラスターのエンドポイントを返します

次の操作は、クラスター mycluster のクラスターエンドポイントを取得します。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeClusters
    &ClusterName=mycluster
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &Version=2021-01-01
    &X-Amz-Credential=<credential>
```

オプションの詳細については、「DescribeClusters」を参照してください。

シャードの使用

シャードは 1~6 個のノードの集まりです。シャードの数が多くレプリカの数が少ないクラスターを作成できます。クラスターあたり最大 500 ノードです。このクラスター設定は、シャード 500 個およびレプリカ 0 個からシャード 100 個およびレプリカ 4 個 (許容されるレプリカの最大数) までです。クラスターのデータは、クラスターのシャード間で分割されます。シャードに複数のノードがある場合、1 つを読み書きのプライマリノード、その他を読み取り専用のレプリカノードとするレプリケーションが実装されます。

AWS マネジメントコンソール を使用して MemoryDB クラスターを作成する際は、クラスター内のシャード数とシャード内のノード数を指定します。詳細については、「MemoryDB クラスターの作成」を参照してください。

シャード内の各ノードのコンピューティング、ストレージ、メモリの仕様は同じです。MemoryDB API を使用すると、ノード数、セキュリティ設定、システムメンテナンス期間など、クラスター全体の属性を制御できます。

詳細については、「MemoryDB のオフラインリシャーディング」および「MemoryDB のオンラインリシャーディング」を参照してください。

シャードの名前を見つける

AWS マネジメントコンソール、AWS CLI または MemoryDB API を使用して、シャードの名前を見つけることができます。

の使用AWS マネジメントコンソール

以下の手順では、AWS マネジメントコンソールを使用して MemoryDB のクラスターのシャード名を検索します。

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 左のナビゲーションペインで [クラスター]を選択します。
3. [名前] で、検索するシャード名のクラスターを選択します。
4. [シャードとノード] タブの [名前] の下にシャードのリストが表示されます。各ノードを展開して詳細を表示することもできます。

の使用AWS CLI

MemoryDB クラスターのシャード (シャード) 名を検索するには、AWS CLI オペレーション `describe-clusters` に以下のオプションパラメータを指定します。

- **`--cluster-name`** - 使用すると、出力を指定されたクラスターの詳細に制限するオプションのパラメータ。このパラメータを省略すると、最大 100 個のクラスターの詳細が返されます。
- **`--show-shard-details`**— 名前を含むシャードの詳細を返します。

このコマンドは、`my-cluster` の詳細を返します。

Linux、macOS、Unix の場合:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Windows の場合:

```
aws memorydb describe-clusters ^
```

```
    --cluster-name my-cluster
    --show-shard-details
```

以下の JSON コードを返します。

改行は読みやすくするために追加しています。

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
```

```
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

MemoryDB API の使用

MemoryDB クラスターのシャード ID を検索するには、API オペレーション DescribeClusters に以下のオプションパラメータを指定します。

- **ClusterName** - 使用すると、出力を指定されたクラスターの詳細に制限するオプションのパラメータ。このパラメータを省略すると、最大 100 個のクラスターの詳細が返されます。
- **ShowShardDetails**— 名前を含むシャードの詳細を返します。

Example

このコマンドは、my-cluster の詳細を返します。

Linux、macOS、Unix の場合:

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeClusters
   &ClusterName=sample-cluster
   &ShowShardDetails=true
   &Version=2021-01-01
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

MemoryDB の実装を管理する

このセクションでは、MemoryDB の実装のさまざまなコンポーネントを管理する方法について詳しく説明します。

トピック

- エンジンバージョン
- JSON の使用開始
- MemoryDB リソースのタグ付け
- メンテナンスの管理
- ベストプラクティス
- MemoryDB レプリケーションを理解する
- スナップショットおよび復元
- Scaling (スケーリング)
- パラメータグループを使用したエンジンパラメータの設定
- 制限されるコマンド
- チュートリアル: Amazon VPC の MemoryDB にアクセスする Lambda 関数の設定

エンジンバージョン

このセクションでは、サポートされる Valkey および Redis OSS エンジンのバージョンについて説明します。

トピック

- MemoryDB バージョン 7.3
- MemoryDB バージョン 7.2.6
- MemoryDB バージョン 7.1 (拡張)
- MemoryDB バージョン 7.0 (拡張)
- Redis OSS バージョン 6.2 (拡張) を搭載した MemoryDB
- エンジンバージョンのアップグレード

MemoryDB バージョン 7.3

2024 年 12 月 1 日、MemoryDB 7.3 がリリースされました。MemoryDB バージョン 7.3 では、マルチリージョンクラスターがサポートされます。そのため、ユーザーは最大 99.999% の可用性を持ち、レイテンシーが非常に低いマルチリージョンアプリケーションを構築できます。MemoryDB マルチリージョンは現在、米国東部 (バージニア北部およびオハイオ)、米国西部 (オレゴン、北カリフォルニア)、欧州 (アイルランド、フランクフルト、ロンドン)、アジアパシフィック (東京、シドニー、ムンバイ、ソウル、シンガポール) の各 AWS リージョンでサポートされています。詳細については、「MemoryDB マルチリージョン」を参照してください。

MemoryDB バージョン 7.2.6

2024 年 10 月 8 日、Valkey 7.2.6 がリリースされました。Valkey 7.2.6 は、以前のバージョンの Redis OSS 7.2.5 と互換性が似ています。Valkey および Redis OSS 7.0 と 7.1 の主な違いは次のとおりです。

- ZRANK コマンドと ZREVRANK コマンドの新しい WITHSCORE オプション
- クライアントがキーの LRU/LFU に影響を与えることなくコマンドを実行するための CLIENT NO-TOUCH。
- レプリケーションに基づいてクラスターモードでノードを論理的にグループ化するためにノードのシャード ID を返す、新しいコマンド CLUSTER MYSHARDID。
- さまざまなデータタイプのパフォーマンスとメモリの最適化。

Valkey 7.2 と Redis OSS 7.1 (または 7.0) の間で発生する可能性のある動作の変更は次のとおりです。

- 同じチャネルにもサブスクライブされている RESP3 クライアントで PUBLISH を呼び出すと、順序が変更され、公開されたメッセージの前に返信が送信されます。
- スクリプトのクライアント側の追跡では、EVAL/FCALL の呼び出し元によって宣言されるキーではなく、スクリプトによって読み取られるキーを追跡するようになりました。
- フリーズ時間のサンプリングは、コマンドの実行中とスクリプトで行われます。
- ブロックされたコマンドがブロック解除されると、ACL、OOM などのチェックが再評価されます。
- ACL 障害エラーメッセージのテキストとエラーコードが統合されています。
- キーが存在しなくなったときに解放されるブロックされたストリームコマンドには、別のエラーコード (-UNBLOCKED ではなく -NOGROUP または -WRONGTYPE) が伴います。

- コマンド統計は、コマンドが実際に実行された場合にのみ、ブロックされたコマンドに対して更新されます。
- ACL ユーザーの内部ストレージは、冗長なコマンドおよびカテゴリルールを削除しなくなりました。これにより、これらのルールが ACL SAVE、ACL GETUSER、ACL LIST の一部として表示される方法が変更される可能性があります。
- TLS ベースのレプリケーション用に作成されたクライアント接続は、可能であれば SNI を使用します。
- XINFO STREAM: seen-time レスポンスフィールドは、最後に成功したインタラクションではなく、最後に試行されたインタラクションを示すようになりました。新しいアクティブタイムレスポンスフィールドは、最後に成功したインタラクションを示すようになりました。
- XREADGROUP と X[AUTO]CLAIM は、何らかの読み取り/クレームを実行できたかどうかに関係なく、コンシューマーを作成します。
- ACL LIST/GETUSER で新しく作成された ACL デフォルトユーザーセットのサニタイズペイロードフラグ。
- HELLO コマンドは、成功しない限りクライアントの状態には影響しません。
- NAN 応答は、inf の現在の動作と同様に、単一の nan タイプに正規化されます。

Valkey の詳細については、「Valkey」を参照してください。

Valkey 7.2 リリースの詳細については、GitHub の Valkey で 「Redis OSS 7.2.4 リリースノート」 (Valkey 7.2 には Redis OSS からバージョン 7.2.4 までのすべての変更が含まれています) と「Valkey 7.2 リリースノート」を参照してください。

MemoryDB バージョン 7.1 (拡張)

MemoryDB バージョン 7.1 では、すべてのリージョンでベクトル検索機能のサポートが追加され、重要なバグ修正とパフォーマンス強化が追加されました。

- ベクトル検索機能: ベクトル検索は、既存の MemoryDB 機能で使用できます。ベクトル検索を使用しないアプリケーションは、その存在の影響を受けません。ベクトル検索は、すべてのリージョンで MemoryDB バージョン 7.1 以降で利用できます。詳細については、こちらのドキュメントを参照してください。

> **Note**
>
> MemoryDB バージョン 7.1 は Redis OSS v7.0 と互換性があります。Redis OSS 7.0 のリリースの詳細については、GitHub の Redis OSS の「Redis OSS 7.0 リリースノート」を参照してください。

MemoryDB バージョン 7.0 (拡張)

MemoryDB 7.0 では、多くの改善と新機能のサポートが追加されています。

- 関数: MemoryDB 7 では、関数のサポートが追加され、管理されたエクスペリエンスが提供されるため、デベロッパーは MemoryDB クラスターに保存されたアプリケーションロジックを使用して LUA スクリプトを実行でき、クライアントは毎回の接続でスクリプトをサーバーに再送信する必要がありません。
- ACL の改善: MemoryDB 7 では、次期バージョンのアクセスコントロールリスト (ACL) のサポートが追加されました。MemoryDB OSS Valkey 7 または Redis OSS 7 では、クライアントは特定のキーまたはキースペースに対して複数の権限セットを指定できるようになりました。
- シャードされた Pub/Sub: MemoryDB 7 では、クラスターモード有効 (CME) で MemoryDB を実行する際に、Pub/Sub 機能をシャードされた方法で実行するサポートが追加されました。Pub/Sub 機能により、パブリッシャーはチャネル上の任意の数のサブスクライバーにメッセージを発行できます。Amazon MemoryDB Valkey 7 および Redis OSS 7 では、チャネルは MemoryDB クラスター内のシャードにバインドされるため、シャード間でチャネル情報を伝達する必要がありません。これにより、スケーラビリティが向上しました。
- 拡張 I/O 多重化: MemoryDB Valkey 7 および Redis OSS バージョン 7 では、拡張された I/O 多重化が導入されています。これにより、MemoryDB クラスターへの多数の同時クライアント接続がある高スループットワークロードのスループットが向上し、レイテンシーが短縮されます。例えば、r6g.4xlarge ノードのクラスターを使用し、5200 の同時クライアントを実行する場合、MemoryDB バージョン 6 と比較して、スループット (1 秒あたりの読み取りおよび書き込み操作) が最大 46% 向上し、P99 レイテンシーが最大 21% 減少します。

Valkey の詳細については、「Valkey」を参照してください。

Valkey 7.2 リリースの詳細については、GitHub の Valkey で 「Redis OSS 7.2.4 リリースノート」 (Valkey 7.2 には Redis OSS からバージョン 7.2.4 までのすべての変更が含まれています) と「Valkey 7.2 リリースノート」を参照してください。

Redis OSS バージョン 6.2 (拡張) を搭載した MemoryDB

MemoryDB では、Redis OSS エンジンの次のバージョンが導入されています。それには、アクセスコントロールリスト (ACL) によるユーザー認証、自動バージョンアップグレードのサポート、クライアント側のキャッシュ、および大幅な運用の改善などが含まれます。

Redis エンジンバージョン 6.2.6 では、ネイティブ JavaScript Object Notation (JSON) 形式のサポートも導入されています。これは、Redis OSS クラスター内の複雑なデータセットをエンコードするシンプルでスキーマレスな方法です。JSON サポートにより、JSON 上で動作するアプリケーションのパフォーマンスと Redis OSS API を活用できます。詳細については、「JSON の使用開始」を参照してください。また、このデータ型の使用状況を監視するために CloudWatch に組み込まれている JSON関連のメトリック`JsonBasedCmds`も含まれています。詳細については、「MemoryDB のメトリック」を参照してください。

MemoryDB では、Redis OSS 6 以降、複数のパッチバージョンを提供するのではなく、Redis OSS マイナーリリースごとに 1 つのバージョンが提供されます。これは、複数のマイナーバージョンから選択する必要がある場合の混乱とあいまいさを最小限に抑えるように設計されています。MemoryDBは、実行中のクラスターのマイナーバージョンとパッチバージョンを自動的に管理し、パフォーマンスの向上とセキュリティ強化を保証します。これは、サービス更新キャンペーンを通じて、標準的な顧客通知チャネルで処理されます。詳細については、「MemoryDB のサービスの更新」を参照してください。

作成時にエンジンバージョンを指定しない場合、MemoryDB は優先する Redis OSS バージョンを自動的に選択します。一方、エンジンバージョンを `6.2` で指定する場合、MemoryDB は利用可能な任意のパッチバージョンの Redis OSS 6.2 を自動的に呼び出します。

例えば、クラスターを作成するとき、`--engine-version` パラメータは `6.2` に設定されます。 クラスターは、作成時に、現在利用可能な優先パッチバージョンで起動されます。完全なエンジンバージョン値を持つリクエストは拒否され、例外がスローされ、プロセスは失敗します。

`DescribeEngineVersions` API の呼び出し時に、`EngineVersion` パラメータの値が 6.2に設定され、実際のフルエンジンバージョンは `EnginePatchVersion` フィールドに返されます。

Redis OSS 6.2 のリリースの詳細については、GitHub の Redis OSS の「Redis 6.2 リリースノート」を参照してください。

エンジンバージョンのアップグレード

MemoryDB はデフォルトで、サービスの更新を通じて実行中のクラスターのパッチバージョンを自動的に管理します。クラスターの `AutoMinorVersionUpgrade` プロパティを false に設定する

と、マイナーバージョンのauto アップグレードを追加でオプトアウトできます。ただし、auto パッチバージョンアップグレードをオプトアウトすることはできません。

自動アップグレードを開始する前に、クラスターを実現するプロトコルに準拠したソフトウェアを、MemoryDB がサポートする新しいバージョンにアップグレードするかどうかと、またいつアップグレードするかを管理します。このレベルのコントロールにより、特定のバージョンとの互換性を維持する、本稼働環境にデプロイする前にアプリケーションで新しいバージョンをテストする、および独自の条件とタイムラインでバージョンのアップグレードを実行することができます。

Redis OSS エンジンを搭載した既存の MemoryDB から Valkey エンジンにアップグレードすることもできます。

クラスターへのエンジンバージョンアップグレードは、以下の方法で開始できます。

- クラスターを更新し、新しいエンジンバージョンを指定する。詳細については、「MemoryDB クラスターの変更」を参照してください。
- 該当するエンジンバージョンのサービスアップデートを適用します。詳細については、「MemoryDB のサービスの更新」を参照してください。

次の点に注意してください。

- 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が、以前のエンジンバージョンにダウングレードすることはできません。以前のエンジンバージョンを使用する場合は、既存のクラスターを削除し、新たにそれを以前のエンジンバージョンで作成する必要があります。
- ほとんどの主要な改善は古いバージョンにバックポートされないため、最新のメジャーバージョンに定期的にアップグレードすることをお勧めします。MemoryDB が別の AWS リージョンでも利用可能になると、そのリージョンでは、その時点で最新の 2 つの `MAJOR.MINOR` バージョンの MemoryDB がサポートされます。例えば、新しい AWS リージョンが立ち上がり、`MAJOR.MINOR` MemoryDB のバージョンが 7.0 と 6.2 である場合、新しい AWS リージョンでは MemoryDB は、バージョン 7.0 と 6.2 をサポートします。MemoryDB の新しい `MAJOR.MINOR` バージョンがリリースされるたびに、MemoryDB は新しくリリースされた MemoryDB バージョンのサポートを継続的に追加します。MemoryDB のリージョンの選択について詳しくは、「サポートされているリージョンおよびエンドポイント」を参照してください。
- エンジンのバージョニングは、パッチの適用方法をできる限り制御できるように設計されています。ただし、システムまたはキャッシュソフトウェアに重大なセキュリティ脆弱性が発生した場合に、MemoryDBはお客様に代わってクラスターにパッチを適用するための権限を有します。

- MemoryDB では、複数のパッチバージョンを提供するのではなく、Valkey または Redis OSS マイナーリリースごとに 1 つのバージョンが提供されます。これは、複数のバージョンから選択する必要がある場合の混乱とあいまいさを最小限に抑えるように設計されています。MemoryDBは、実行中のクラスターのマイナーバージョンとパッチバージョンを自動的に管理し、パフォーマンスの向上とセキュリティ強化を保証します。これは、サービス更新キャンペーンを通じて、標準的な顧客通知チャネルで処理されます。詳細については、「MemoryDB のサービスの更新」を参照してください。
- 最小限のダウンタイムでクラスターバージョンをアップグレードできます。このクラスターは、アップグレード中のすべての読み取りと、数秒かかるフェイルオーバー操作中を除き、ほとんどすべてのアップグレード中の書き込みに対応します。
- エンジンのアップグレードは、受信書き込みトラフィックが少ない時間帯に行うことをお勧めします。

  複数のシャードを含むクラスターは、次のように処理され、パッチが適用されます。

  - アップグレード操作は、1 つのシャードにつき常に 1 回のみ実行されます。
  - 各シャードでは、プライマリが処理される前にすべてのレプリカが処理されます。シャードにレプリカが少ない場合、他のシャードのレプリカが処理を終了する前に、そのシャードのプライマリが処理されることがあります。
  - すべてのシャード間で、プライマリノードはシリーズで処理されます。一度にアップグレードできるプライマリノードは 1 つだけです。

トピック

- エンジンバージョンのアップグレード方法
- ブロックされた Redis OSS エンジンのアップグレードの解決

エンジンバージョンのアップグレード方法

クラスターのバージョンのアップグレードを開始するには、 MemoryDBコンソールAWS CLI、またはMemoryDB API を使用して、より新しいエンジンバージョンを指定します。詳細については、以下のトピックを参照してください。

- の使用AWS マネジメントコンソール
- の使用AWS CLI
- MemoryDB API の使用

ブロックされた Redis OSS エンジンのアップグレードの解決

以下の表に示すように、保留中のスケールアップオペレーションがある場合、Redis OSS エンジンのアップグレードオペレーションはブロックされます。

保留中のオペレーション	ブロックされたオペレーション
スケールアップ	即時のエンジンのアップグレード
エンジンのアップグレード	即時のスケールアップ
スケールアップとエンジンのアップグレード	即時のスケールアップ
	即時のエンジンのアップグレード

JSON の使用開始

MemoryDB は、Valkey または Redis OSS クラスター内の複雑なデータセットをエンコードするためのシンプルでスキーマレスな方法である JavaScript Object Notation (JSON) をサポートしています。クラスター内で JavaScript Object Notation (JSON) 形式を使用してデータをネイティブに保存およびアクセスし、それらのクラスターに保存されている JSON データを更新できます。カスタムコードを管理してシリアル化および逆シリアル化する必要はありません。

JSON 上で動作するアプリケーションに Valkey または Redis OSS API を使用することに加えて、オブジェクト全体を操作することなく、JSON ドキュメントの特定の部分を効率的に取得および更新できるようになりました。これによってパフォーマンスの向上とコストの削減が可能になります。また、Goessner-style `JSONPath` クエリを使用して、JSON ドキュメントの内容を検索することもできます。

サポートされているエンジンバージョンでクラスターを作成すると、JSON データタイプおよび関連するコマンドが自動的に使用可能になります。これは、RedisJSON モジュールのバージョン 2 と互換性のある API および RDB であるため、既存の JSON ベースの Valkey または Redis OSS アプリケーションを MemoryDB に簡単に移行できます。サポートされているコマンドの詳細については、「サポートされているコマンド」を参照してください。

JSON 関連のメトリクス `JsonBasedCmds` および は、このデータタイプの使用状況をモニタリングするために CloudWatch に組み込まれています。詳細については、「Metrics for MemoryDB」を参照してください。

> **Note**
>
> JSON を使用するには、Valkey 7.2 以降、Redis OSS エンジンバージョン 6.2.6 以降を実行している必要があります。

トピック

- JSON データ型の概要
- サポートされているコマンド

JSON データ型の概要

MemoryDB では、JSON データ型を操作するためのいくつかの Valkey および Redis OSS コマンドをサポートしています。以下に、JSON データ型の概要と、サポートされているコマンドの詳細なリストを示します。

用語

言葉	説明
JSON ドキュメント	JSON キーの値です
JSON 値	ドキュメント全体を表すルートを含む、JSON ドキュメントのサブセットです。値は、コンテナまたはコンテナ内のエントリにすることができます
JSON 要素	JSON 値と同じです

サポートされている JSON 標準

JSON 形式は、RFC 7159 および ECMA-404 JSON データ交換標準に準拠しています。JSON テキストの UTF-8 Unicode がサポートされています。

ルート要素

ルート要素は任意の JSON データ型にすることができます。以前の RFC 4627 では、オブジェクトまたは配列のみがルート値として許可されていたことに注意してください。RFC 7159 への更新以降、JSON ドキュメントのルートは任意の JSON データ型にすることができます。

ドキュメントサイズの制限

JSON ドキュメントは、迅速なアクセスおよび変更のために最適化された形式で内部的に格納されます。通常、この形式では、同じドキュメントのシリアル化された同等の表現よりもいくらか多くのメモリを消費することになります。単一の JSON ドキュメントによるメモリ消費量は 64 MB に制限されています。これは JSON 文字列ではなく、インメモリデータ構造のサイズです。`JSON.DEBUG MEMORY` コマンドを使用することで、JSON ドキュメントが消費するメモリの量を確認できます。

JSON ACLs

- JSON データ型は、Valkey および Redis OSS アクセスコントロールリスト (ACL) 機能に完全に統合されています。JSON コマンドおよびデータへのアクセスを簡単に管理するために、既存のデータ型ごとのカテゴリ (@string、@hash など) と同様の新しいカテゴリ @json が追加されました。他の既存の Valkey および Redis OSS コマンドは @json カテゴリのメンバーではありません。すべての JSON コマンドは、キースペースまたはコマンドの制限と権限を強制します。
- 次の 5 つの既存の ACL カテゴリが、新しい JSON コマンドを含むように更新されています: @read、@write、@fast、@slow、@admin。以下の表は、適切なカテゴリへの JSON コマンドのマッピングを示しています。

ACL

JSON コマンド	@read	@write	@fast	@slow	@admin
JSON.ARRAPPEND		y	y		
JSON.ARRINDEX	y		y		
JSON.ARRINSERT		y	y		

JSON コマンド	@read	@write	@fast	@slow	@admin
JSON.ARRLEN	y		y		
JSON.ARRPOP		y	y		
JSON.ARRTRIM		y	y		
JSON.CLEAR		y	y		
JSON.DEBUG	y			y	y
JSON.DEL		y	y		
JSON.FORGET		y	y		
JSON.GET	y		y		
JSON.MGET	y		y		
JSON.NUMINCRBY		y	y		
JSON.NUMMULTBY		y	y		
JSON.OBJKEYS	y		y		
JSON.OBJLEN	y		y		
JSON.RESP	y		y		

JSON コマンド	@read	@write	@fast	@slow	@admin
JSON.SET		y		y	
JSON.STRAPPEND		y	y		
JSON.STRLEN	y		y		
JSON.STRLEN	y		y		
JSON.TOGGLE		y	y		
JSON.TYPE	y		y		
JSON.NUMINCRBY		y	y		

ネスト深度の制限

JSON オブジェクトまたは配列に、それ自体が別の JSON オブジェクトまたは配列である要素がある場合、その内部オブジェクトまたは配列は外部オブジェクトまたは配列内で「ネスト」と呼ばれます。ネストの最大深度の制限は 128 です。128 より大きいネスト深度を含むドキュメントを作成しようとすると、エラーで拒否されます。

コマンド構文

ほとんどのコマンドでは、最初の引数として Valkey または Redis OSS キー名が必要です。一部のコマンドにはパス引数もあります。パス引数は、オプションで提供されない場合、デフォルトでルートになります。

表記法:

- 必須引数は山括弧 (例: <key>) で囲みます。
- オプションの引数は角括弧 (例: [path]) で囲みます。

- 追加のオプション引数は省略記号「…」(例: [json…]) で示されます。

パス構文

Valkey または Redis OSS の JSON では、次の 2 種類のパス構文をサポートしています。

- 拡張構文 – 以下の表に示すように、Goessner で説明されている JSONPath 構文に従います。わかりやすくするために、表の説明を並べ替え、一部変更しています。
- 制限構文 — クエリ機能が制限されます。

> **Note**
>
> 一部のコマンドの結果は、使用されるパス構文のタイプの影響を受けます。

クエリパスが「$」で始まる場合は、拡張構文が使用されます。その他の場合は、制限構文が使用されます。

拡張構文

記号/式	説明
$	ルート要素
. または	子演算子
..	再帰下降
*	ワイルドカード。オブジェクトまたは配列のすべての要素。
[]	配列の添字演算子。インデックスは 0 ベースです。
[,]	union 演算子
start:end:step	配列のスライス演算子

記号/式	説明
?()	フィルタ (スクリプト) 式を現在の配列またはオブジェクトに適用します
()	フィルタ式
@	処理中の現在のノードを参照するフィルタ式で使用されます
==	等しい。フィルタ式で使用されます。
!=	等しくない。フィルタ式で使用されます。
>	より大きい。フィルタ式で使用されます。
>=	以上。フィルタ式で使用されます。
<	より小さい。フィルタ式で使用されます。
<=	以下。フィルタ式で使用されます。
&&	論理 AND。複数のフィルタ式を組み合わせるために使用されます。
||	論理 OR。複数のフィルタ式を組み合わせるために使用されます。

例

以下の例は、Goessner のサンプル XML データに基づいて構築されています。フィールドを追加して一部変更しました。

```
{ "store": {
    "book": [
      { "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95,
        "in-stock": true,
        "sold": true
```

```
      },
      { "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99,
        "in-stock": false,
        "sold": true
      },
      { "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99,
        "in-stock": true,
        "sold": false
      },
      { "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99,
        "in-stock": false,
        "sold": false
      }
    ],
    "bicycle": {
      "color": "red",
      "price": 19.95,
      "in-stock": true,
      "sold": false
    }
  }
}
```

パス	説明
$.store.book*.author	この店のすべての本の著者です
$..author	すべての著者です
$.store.*	店のすべてのメンバー

パス	説明
$"store".*	店のすべてのメンバー
$.store..price	店のすべてのものの価格です
$..*	JSON 構造のすべての再帰的メンバーです
$..book*	すべての本です
$..book0	最初の本です
$..book-1	最後の本です
$..book0:2	最初の 2 冊の本です
$..book0,1	最初の 2 冊の本です
$..book0:4	インデックス 0 から 3 までの本です (終了インデックスは含みません)
$..book0:4:2	インデックス 0, 2 の本です
$..book?(@.isbn)	ISBN 番号があるすべての本です
$..book?(@.price<10)	10 ドルより安いすべての本
'$..book?(@.price < 10)'	10 ドルより安いすべての本。(パスに空白が含まれている場合は、引用符で囲む必要があります)
'$..book?(@"price"< 10)'	10 ドルより安いすべての本
'$..book?(@."price"< 10)'	10 ドルより安いすべての本
$..book?(@.price>=10&&@.price<=100)	10 ドルから 100 ドルの価格帯 (この値を含む) にあるすべての本です
'$..book?(@.price>=10 && @.price<=100)'	10 ドルから 100 ドルの価格帯 (この値を含む) にあるすべての本です。(パスに空白が含まれている場合は、引用符で囲む必要があります)

パス	説明
$..book?(@.sold==true||@.in-stock==false)	すべての本が売れたか、在庫切れです
'$..book?(@.sold == true || @.in-stock == false)'	すべての本が売れたか、在庫切れです。(パスに空白が含まれている場合は、引用符で囲む必要があります)
'$.store.book?(@."category" == "fiction")'	フィクションのカテゴリのすべての本です
'$.store.book?(@."category" != "fiction")'	ノンフィクションのカテゴリのすべての本です

フィルタ式の例:

```
127.0.0.1:6379> JSON.SET k1 . '{"books": [{"price":5,"sold":true,"in-
stock":true,"title":"foo"}, {"price":15,"sold":false,"title":"abc"}]}'
OK
127.0.0.1:6379> JSON.GET k1 $.books[?(@.price>1&&@.price<20&&@.in-stock)]
"[{\"price\":5,\"sold\":true,\"in-stock\":true,\"title\":\"foo\"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?(@.price>1 && @.price<20 && @.in-stock)]'
"[{\"price\":5,\"sold\":true,\"in-stock\":true,\"title\":\"foo\"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?((@.price>1 && @.price<20) && (@.sold==false))]'
"[{\"price\":15,\"sold\":false,\"title\":\"abc\"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?(@.title == "abc")]'
[{"price":15,"sold":false,"title":"abc"}]

127.0.0.1:6379> JSON.SET k2 . '[1,2,3,4,5]'
127.0.0.1:6379> JSON.GET k2 $.*.[?(@>2)]
"[3,4,5]"
127.0.0.1:6379> JSON.GET k2 '$.*.[?(@ > 2)]'
"[3,4,5]"

127.0.0.1:6379> JSON.SET k3 . '[true,false,true,false,null,1,2,3,4]'
OK
127.0.0.1:6379> JSON.GET k3 $.*.[?(@==true)]
"[true,true]"
127.0.0.1:6379> JSON.GET k3 '$.*.[?(@ == true)]'
"[true,true]"
127.0.0.1:6379> JSON.GET k3 $.*.[?(@>1)]
"[2,3,4]"
127.0.0.1:6379> JSON.GET k3 '$.*.[?(@ > 1)]'
```

```
"[2,3,4]"
```

制限構文

記号/式	説明
. または	子演算子
[]	配列の添字演算子。インデックスは 0 ベースです。

例

パス	説明
.store.book0.author	最初の本の著者です
.store.book-1.author	最後の本の著者です
.address.city	都市名です
"store""book"0"title"	最初の本のタイトルです
"store""book"-1"title"	最後の本のタイトルです

> **Note**
>
> このドキュメントで引用されているすべての Goessner コンテンツには、クリエイティブコモンズライセンスが適用されます。

一般的なエラープレフィックス

各エラーメッセージにはプレフィックスが付いています。以下は、一般的なエラープレフィックスのリストです:

プレフィックス	説明
ERR	一般的なエラーです
制限	サイズ制限超過エラー。例:ドキュメントのサイズ制限やネストの深さの制限を超えた
NONEXISTENT	キーまたはパスが存在しません
OUTOFBOUNDARIES	配列インデックスが範囲外です
SYNTAXERR	構文エラー
WRONGTYPE	値のタイプが間違っています

JSON 関連メトリクス

以下の JSON 情報メトリクスが提供されます。

Info	説明
json_total_memory_bytes	JSON オブジェクトに割り当てられたメモリの合計です
json_num_documents	Valkey または Redis OSS エンジン内のドキュメントの総数です

コアメトリクスのクエリを実行するには、以下のコマンドを実行します。

```
info json_core_metrics
```

MemoryDB が JSON とどのように相互作用するか

以下は、MemoryDB が JSON データ型とどのように相互作用するかを示しています。

演算子の優先順位

フィルタリングの条件式を評価するときは、ほとんどの言語と同様に、&& が最も優先され、次に || が評価されます。括弧内の操作が最初に実行されます。

最大パスネスト制限の動作

MemoryDB の最大パスネストの制限は 128 です。したがって、`$.a.b.c.d...` のような値は 128 レベルまでしか到達できません。

数値の処理

JSON では、整数と浮動小数点数で異なるデータ型を使用しません。それらはすべて数値と呼ばれます。

JSON 番号を受信すると、2 つのフォーマットのいずれかで保存されます。数値が 64 ビットの符号付き整数に収まる場合は、その形式に変換されます。それ以外の場合は、文字列として格納されます。2 つの JSON 数値 (JSON.NUMINCRBY と JSON.NUMMULTBY など) に対する算術演算では、可能な限り精度を保つように努めています。2 つのオペランドと結果の値が 64 ビットの符号付き整数に収まる場合は、整数演算が実行されます。それ以外の場合は、入力オペランドが 64 ビット IEEE 倍精度浮動小数点数に変換され、算術演算が実行されて結果が文字列に変換されます。

算術コマンド `NUMINCRBY` および `NUMMULTBY`:

- 両方の数値が整数で、結果が int64 の範囲外である場合は、自動的に倍精度浮動小数点数になります。
- 少なくとも 1 つの数値が浮動小数点の場合、結果は倍精度浮動小数点数になります。
- 結果が倍の範囲を超える場合は、`OVERFLOW` エラーが返されます。

> **Note**
>
> Redis OSS エンジンバージョン 6.2.6.R2 以前では、JSON 数値を入力で受け取ると、64 ビット符号付き整数または 64 ビット IEEE 倍精度浮動小数点の 2 つの内部バイナリ表現のいずれかに変換されます。元の文字列、およびそのすべての書式は保持されません。そのため、数値が JSON 応答の一部として出力されるときに、内部のバイナリ表現が、一般的な書式ルールが使用された印刷可能文字列に変換されます。これらのルールにより、受信した文字列とは異なる文字列が生成される場合があります。
>
> - 両方の数値が整数で、結果が `int64` の範囲外である場合は、自動的に 64 ビット IEEE 倍精度浮動小数点数になります。
> - 数字の少なくとも 1 つが浮動小数点の場合、結果は 64 ビット IEEE 倍精度浮動小数点数になります。
> - 結果が 64 ビット IEEE 倍精度の範囲を超える場合は、`OVERFLOW` エラーが返されます。

利用可能なコマンドの詳細なリストについては、「サポートされているコマンド」を参照してください。

厳密な構文評価

MemoryDB では、パスのサブセットに有効なパスが含まれていても、無効な構文の JSON パスは許可されません。これは、お客様のために正しい動作を維持することを目的とした処置です。

サポートされているコマンド

以下の JSON コマンドがサポートされています:

トピック

- JSON.ARRAPPEND
- JSON.ARRINDEX
- JSON.ARRINSERT
- JSON.ARRLEN
- JSON.ARRPOP
- JSON.ARRTRIM
- JSON.CLEAR
- JSON.DEBUG
- JSON.DEL
- JSON.FORGET
- JSON.GET
- JSON.MGET
- JSON.NUMINCRBY
- JSON.NUMMULTBY
- JSON.OBJLEN
- JSON.OBJKEYS
- JSON.RESP
- JSON.SET
- JSON.STRAPPEND
- JSON.STRLEN
- JSON.TOGGLE

- JSON.TYPE

JSON.ARRAPPEND

パスの配列値に 1 つ以上の値を追加します。

構文

```
JSON.ARRAPPEND <key> <path> <json> [json ...]
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス (必須) - JSON パス
- json (必須) - 配列に追加される JSON 値

戻る

パスが拡張構文の場合:

- 各パスの配列の新しい長さを表す整数の配列。
- 値が配列でない場合、対応する戻り値は null です。
- SYNTAXERR入力 json 引数のいずれかが有効な JSON 文字列でない場合は、 エラーになります。
- NONEXISTENTパスが存在しない場合は、 エラーになります。

パスが制限構文の場合:

- 整数、配列の新しい長さ。
- 複数の配列値が選択されている場合、コマンドは最後に更新された配列の新しい長さを返します。
- パスの値が配列でない場合は、WRONGTYPE エラーになります。
- 入力 json 引数のいずれかが有効な JSON 文字列でない場合は、SYNTAXERR エラーになります。
- パスが存在しない場合は、NONEXISTENT エラーになります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
```

```
OK
127.0.0.1:6379> JSON.ARRAPPEND  k1 $[*] '"c"'
1) (integer) 1
2) (integer) 2
3) (integer) 3
127.0.0.1:6379> JSON.GET k1
"[[\"c\"],[\"a\",\"c\"],[\"a\",\"b\",\"c\"]]"
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRAPPEND  k1 [-1] '"c"'
(integer) 3
127.0.0.1:6379> JSON.GET k1
"[[],[\"a\"],[\"a\",\"b\",\"c\"]]"
```

JSON.ARRINDEX

パスの配列で最初に出現するスカラー JSON 値を検索します。

- 範囲外のエラーは、インデックスを配列の開始と終了に丸めることによって処理されます。
- start > end の場合は、-1 (見つからない) を返します。

構文

```
JSON.ARRINDEX <key> <path> <json-scalar> [start [end]]
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス (必須) - JSON パス
- json-scalar (必須) — 検索するスカラー値。JSON スカラーはオブジェクトでも配列でもない値を指します。つまり、文字列、数値、ブール値、null はスカラー値となります。
- 開始「オプショナル」– 開始インデックス、インクルーシブ。指定しない場合、デフォルトで 0 になります。
- 終了「オプショナル」– 終了インデックス、エクスクルーシブ。指定しない場合、デフォルトで 0 になります。したがって、最後の要素が含まれます。0 または -1 は、最後の要素が含まれることを意味します。

戻る

パスが拡張構文の場合:

- 整数の配列。各値は、パスの配列の一致する要素のインデックスです。見つからない場合の値は -1 です。
- 値が配列でない場合、対応する戻り値は null です。

パスが制限構文の場合:

- 整数、一致する要素のインデックス。見つからない場合は -1。
- パスの値が配列でない場合は、WRONGTYPE エラーになります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRINDEX k1 $[*] '"b"'
1) (integer) -1
2) (integer) -1
3) (integer) 1
4) (integer) 1
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.ARRINDEX k1 .children '"Tom"'
(integer) 2
```

JSON.ARRINSERT

そのインデックスの前のパスの配列値に 1 つ以上の値を挿入します。

構文

```
JSON.ARRINSERT <key> <path> <index> <json> [json ...]
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス (必須) - JSON パス
- index (必須) – 値が挿入される前の配列インデックス。
- json (必須) - 配列に追加される JSON 値

戻る

パスが拡張構文の場合:

- 各パスの配列の新しい長さを表す整数の配列。
- 値が空の配列の場合、対応する戻り値は null です。
- 値が配列でない場合、対応する戻り値は null です。
- index 引数が範囲外である場合は、OUTOFBOUNDARIES エラーになります。

パスが制限構文の場合:

- 整数、配列の新しい長さ。
- パスの値が配列でない場合は、WRONGTYPE エラーになります。
- index 引数が範囲外である場合は、OUTOFBOUNDARIES エラーになります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRINSERT k1 $[*] 0 '"c"'
1) (integer) 1
2) (integer) 2
3) (integer) 3
127.0.0.1:6379> JSON.GET k1
"[[\"c\"],[\"c\",\"a\"],[\"c\",\"a\",\"b\"]]"
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRINSERT k1 . 0 '"c"'
(integer) 4
127.0.0.1:6379> JSON.GET k1
"[\"c\",[],[\"a\"],[\"a\",\"b\"]]"
```

JSON.ARRLEN

パスの配列値の長さを取得します。

構文

```
JSON.ARRLEN <key> [path]
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス「オプショナル」– JSON パス。指定しない場合、デフォルトでルートになります

戻る

パスが拡張構文の場合:

- 各パスの配列の長さを表す整数の配列。
- 値が配列でない場合、対応する戻り値は null です。
- ドキュメントキーが存在しない場合は、null になります。

パスが制限構文の場合:

- 一括文字列の配列。各要素はオブジェクトのキー名です。
- 整数、配列の長さ。
- 複数のオブジェクトが選択されている場合、このコマンドは最初の配列の長さを返します。
- パスの値が配列でない場合は、WRONGTYPE 工ラーになります。
- パスが存在しない場合は、WRONGTYPE 工ラーになります。

- ドキュメントキーが存在しない場合は、null になります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[[], [\"a\"], [\"a\", \"b\"], [\"a\", \"b\", \"c\"]]'
(error) SYNTAXERR Failed to parse JSON string due to syntax error
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRLEN k1 $[*]
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 3

127.0.0.1:6379> JSON.SET k2 . '[[], "a", ["a", "b"], ["a", "b", "c"], 4]'
OK
127.0.0.1:6379> JSON.ARRLEN k2 $[*]
1) (integer) 0
2) (nil)
3) (integer) 2
4) (integer) 3
5) (nil)
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRLEN k1 [*]
(integer) 0
127.0.0.1:6379> JSON.ARRLEN k1 $[3]
1) (integer) 3

127.0.0.1:6379> JSON.SET k2 . '[[], "a", ["a", "b"], ["a", "b", "c"], 4]'
OK
127.0.0.1:6379> JSON.ARRLEN k2 [*]
(integer) 0
127.0.0.1:6379> JSON.ARRLEN k2 $[1]
1) (nil)
127.0.0.1:6379> JSON.ARRLEN k2 $[2]
```

```
1) (integer) 2
```

JSON.ARRPOP

配列からそのインデックスの要素を削除し、返します。空の配列をポップすると null が返されます。

構文

```
JSON.ARRPOP <key> [path [index]]
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス「オプショナル」– JSON パス。指定しない場合、デフォルトでルートになります
- index (オプション) — ポップを開始する配列内の位置。
  - 指定しない場合、デフォルトで -1 になります。これは最後の要素を意味します。
  - 負の値は、最後の要素からの位置を意味します。
  - 境界外インデックスは、それぞれの配列境界に丸められます。

戻る

パスが拡張構文の場合:

- 各パスのポップされた値を表す一括文字列の配列。
- 値が空の配列の場合、対応する戻り値は null です。
- 値が配列でない場合、対応する戻り値は null です。

パスが制限構文の場合:

- 一括文字列。ポップされた JSON 値を表します
- 配列が空の場合は null になります。
- パスの値が配列でない場合は、WRONGTYPE 工ラーになります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRPOP k1 $[*]
1) (nil)
2) "\"a\""
3) "\"b\""
127.0.0.1:6379> JSON.GET k1
"[[],[],[\"a\"]]"
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRPOP k1
"[\"a\",\"b\"]"
127.0.0.1:6379> JSON.GET k1
"[[],[\"a\"]]"

127.0.0.1:6379> JSON.SET k2 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRPOP k2 . 0
"[]"
127.0.0.1:6379> JSON.GET k2
"[[\"a\"],[\"a\",\"b\"]]"
```

JSON.ARRTRIM

部分配列 start, end となるようにパスの配列をトリムします (どちらもこの値を含みます)。

- 配列が空の場合は、何もしないで 0 を返します。
- start < 0 の場合は、0 として扱います。
- end >= サイズ (配列のサイズ) の場合は、サイズ-1 として扱います。
- start >= サイズまたは start > end の場合は、配列を空にして 0 を返します。

構文

```
JSON.ARRINSERT <key> <path> <start> <end>
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス (必須) - JSON パス
- 開始 (必須) — 開始インデックス (この値を含みます)。
- 終了 (必須) — 終了インデックス (この値を含みます)。

戻る

パスが拡張構文の場合:

- 各パスの配列の新しい長さを表す整数の配列。
- 値が空の配列の場合、対応する戻り値は null です。
- 値が配列でない場合、対応する戻り値は null です。
- index 引数が範囲外である場合は、OUTOFBOUNDARIES エラーになります。

パスが制限構文の場合:

- 整数、配列の新しい長さ。
- 配列が空の場合は null になります。
- パスの値が配列でない場合は、WRONGTYPE エラーになります。
- index 引数が範囲外である場合は、OUTOFBOUNDARIES エラーになります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRTRIM k1 $[*] 0 1
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 2
   127.0.0.1:6379> JSON.GET k1
   "[[],[\"a\"],[\"a\",\"b\"],[\"a\",\"b\"]]"
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.ARRTRIM k1 .children 0 1
(integer) 2
127.0.0.1:6379> JSON.GET k1 .children
"[\"John\",\"Jack\"]"
```

JSON.CLEAR

パスの配列またはオブジェクトをクリアします。

構文

```
JSON.CLEAR <key> [path]
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス「オプショナル」– JSON パス。指定しない場合、デフォルトでルートになります

戻る

- 整数、クリアされたコンテナの数。
- 空の配列またはオブジェクトをクリアすると、0 つのコンテナがクリアされます。

> **Note**
>
> Redis OSS バージョン 6.2.6.R2 以前では、空の配列またはオブジェクトをクリアすると、1 つのコンテナがクリアされます。

- コンテナ以外の値をクリアすると 0 が返されます。
- パスに配列やオブジェクト値が見つからない場合、コマンドは 0 を返します。

例

```
127.0.0.1:6379> JSON.SET k1 . '[[], [0], [0,1], [0,1,2], 1, true, null, "d"]'
OK
127.0.0.1:6379>  JSON.CLEAR k1  $[*]
(integer) 6
127.0.0.1:6379> JSON.CLEAR k1  $[*]
```

```
(integer) 0
127.0.0.1:6379> JSON.SET k2 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.CLEAR k2 .children
(integer) 1
127.0.0.1:6379> JSON.GET k2 .children
"[]"
```

JSON.DEBUG

レポート情報。サポートされるサブコマンドは以下のとおりです。

- MEMORY <key> [path] – メモリの使用状況を JSON 値のバイト数でレポートします。パスが指定されていない場合、デフォルトはルートになります。
- DEPTH <key> [path] – JSON ドキュメントの最大パス深度を報告します。

> **Note**
>
> このサブコマンドは、Valkey 7.2 以降または Redis OSS エンジンバージョン 6.2.6.R2 以降でのみ使用できます。

- FIELDS <key> [path] – 指定されたドキュメントパスのフィールド数をレポートします。パスが指定されていない場合、デフォルトはルートになります。コンテナ以外の JSON 値はそれぞれ 1 つのフィールドとしてカウントされます。オブジェクトと配列は、それらを含む JSON 値ごとに 1 つのフィールドを再帰的にカウントします。ルートコンテナを除く各コンテナ値は、1 つの追加フィールドとしてカウントされます。
- HELP – コマンドに関するヘルプメッセージを出力します。

構文

```
JSON.DEBUG <subcommand & arguments>
```

サブコマンドによって異なります。

MEMORY

- パスが拡張構文の場合:
  - 各パスの JSON 値のフィールド数を表す整数の配列を返します。

  - キーが存在しない場合は、空の配列を返します。
- パスが制限構文の場合:
  - 整数のメモリサイズ、および JSON 値 (バイト単位) を返します。
  - キーが存在しない場合は、null を返します。

DEPTH

- JSON ドキュメントの最大パス深度を表す整数を返します。
- キーが存在しない場合は、null を返します。

FIELDS

- パスが拡張構文の場合:
  - 各パスにおける JSON 値のフィールド数を表す整数の配列を返します。
  - キーが存在しない場合は、空の配列を返します。
- パスが制限構文の場合:
  - JSON 値のフィールド数を整数で返します。
  - キーが存在しない場合は、null を返します。

HELP - ヘルプメッセージの配列を返します。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[1, 2.3, "foo", true, null, {}, [], {"a":1, "b":2},
 [1,2,3]]'
OK
127.0.0.1:6379> JSON.DEBUG MEMORY k1 $[*]
1) (integer) 16
2) (integer) 16
3) (integer) 19
4) (integer) 16
5) (integer) 16
6) (integer) 16
7) (integer) 16
8) (integer) 50
```

```
9) (integer) 64
127.0.0.1:6379> JSON.DEBUG FIELDS k1 $[*]
1) (integer) 1
2) (integer) 1
3) (integer) 1
4) (integer) 1
5) (integer) 1
6) (integer) 0
7) (integer) 0
8) (integer) 2
9) (integer) 3
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.DEBUG MEMORY k1
(integer) 632
127.0.0.1:6379> JSON.DEBUG MEMORY k1 .phoneNumbers
(integer) 166

127.0.0.1:6379> JSON.DEBUG FIELDS k1
(integer) 19
127.0.0.1:6379> JSON.DEBUG FIELDS k1 .address
(integer) 4

127.0.0.1:6379> JSON.DEBUG HELP
1) JSON.DEBUG MEMORY <key> [path] - report memory size (bytes) of the JSON element.
 Path defaults to root if not provided.
2) JSON.DEBUG FIELDS <key> [path] - report number of fields in the JSON element. Path
 defaults to root if not provided.
3) JSON.DEBUG HELP - print help message.
```

JSON.DEL

ドキュメントキーのパスにある JSON 値を削除します。パスがルートの場合、Valkey または Redis OSS からキーを削除することと同じです。

構文

```
JSON.DEL <key> [path]
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス「オプショナル」– JSON パス。指定しない場合、デフォルトでルートになります

戻る

- 削除された要素の数。
- キーが存在しない場合は、0 になります。
- JSON パスが無効であるか、存在しない場合は、0 になります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.DEL k1 $.d.*
(integer) 3
127.0.0.1:6379> JSOn.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.DEL k1 $.e[*]
(integer) 5
127.0.0.1:6379> JSOn.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[]}"
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}, "e": [1,2,3,4,5]}'
```

```
OK
127.0.0.1:6379> JSON.DEL k1 .d.*
(integer) 3
127.0.0.1:6379> JSON.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.DEL k1 .e[*]
(integer) 5
127.0.0.1:6379> JSON.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[]}"
```

JSON.FORGET

JSON.DEL のエイリアス

JSON.GET

1 つ以上のパスにあるシリアル化された JSON を返します。

構文

```
JSON.GET <key>
[INDENT indentation-string]
[NEWLINE newline-string]
[SPACE space-string]
[NOESCAPE]
[path ...]
```

- キー (必須) - JSON ドキュメントタイプのキー
- INDENT/NEWLINE/SPACE (オプション) – 返される JSON 文字列の形式、すなわち「整形出力」を制御します。それぞれのデフォルト値は空の文字列です。任意の組み合わせでオーバーライドすることが可能です。これらは任意の順序で指定できます。
- NOESCAPE - オプション。レガシーの互換性のために存在しており、他の効果はありません。
- パス (オプション) – ゼロ以上の JSON パス。何も指定されていない場合は、デフォルトでルートになります。パス引数は末尾に置く必要があります。

戻る

拡張パス構文:

パスが 1 つ指定されている場合:

- 値の配列のシリアル化された文字列を返します。
- 値が選択されなかった場合は、空の配列を返します。

複数のパスが指定されている場合:

- 各パスがキーである、文字列化された JSON オブジェクトを返します。
- 拡張パス構文と制限パス構文が混在している場合、結果は拡張構文に準拠します。
- パスが存在しない場合、対応する値は空の配列です。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.GET k1 $.address.*
"[\"21 2nd Street\",\"New York\",\"NY\",\"10021-3100\"]"
127.0.0.1:6379> JSON.GET k1 indent "\t" space " " NEWLINE "\n" $.address.*
"[\n\t\"21 2nd Street\",\n\t\"New York\",\n\t\"NY\",\n\t\"10021-3100\"\n]"
127.0.0.1:6379> JSON.GET k1 $.firstName $.lastName $.age
"{\"$.firstName\":[\"John\"],\"$.lastName\":[\"Smith\"],\"$.age\":[27]}"
127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}}'
OK
127.0.0.1:6379> json.get k2 $..*
"[{},{\"a\":1},{\"a\":1,\"b\":2},1,1,2]"
```

制限パス構文:

```
 127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
```

```
OK
127.0.0.1:6379> JSON.GET k1 .address
"{\"street\":\"21 2nd Street\",\"city\":\"New York\",\"state\":\"NY\",\"zipcode\":
\"10021-3100\"}"
127.0.0.1:6379> JSON.GET k1 indent "\t" space " " NEWLINE "\n" .address
"{\n\t\"street\": \"21 2nd Street\",\n\t\"city\": \"New York\",\n\t\"state\": \"NY\",\n
\t\"zipcode\": \"10021-3100\"\n}"
127.0.0.1:6379> JSON.GET k1 .firstName .lastName .age
"{\".firstName\":\"John\",\".lastName\":\"Smith\",\".age\":27}"
```

JSON.MGET

複数のドキュメントキーからのパスでシリアル化された JSON を取得します。存在しないキーまたは JSON パスの場合は null を返します。

構文

```
JSON.MGET <key> [key ...] <path>
```

- key (必須) - ドキュメントタイプの 1 つ以上のキー。
- パス (必須) - JSON パス

戻る

- 一括文字列の配列。配列のサイズは、コマンド内のキーの数と等しくなります。配列の各要素には、(a) パスによって配置されたシリアル化された JSON、または (b) キーが存在しない場合、パスがドキュメント内に存在しない場合、パスが無効な場合 (構文エラー) は null が入力されます。
- 指定されたキーのいずれかが存在し、JSON キーではない場合、コマンドは WRONGTYPE エラーを返します。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"address":{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021"}}'
OK
127.0.0.1:6379> JSON.SET k2 . '{"address":{"street":"5 main
 Street","city":"Boston","state":"MA","zipcode":"02101"}}'
```

```
OK
127.0.0.1:6379> JSON.SET k3 . '{"address":{"street":"100 Park
 Ave","city":"Seattle","state":"WA","zipcode":"98102"}}'
OK
127.0.0.1:6379> JSON.MGET k1 k2 k3 $.address.city
1) "[\"New York\"]"
2) "[\"Boston\"]"
3) "[\"Seattle\"]"
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"address":{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021"}}'
OK
127.0.0.1:6379> JSON.SET k2 . '{"address":{"street":"5 main
 Street","city":"Boston","state":"MA","zipcode":"02101"}}'
OK
127.0.0.1:6379> JSON.SET k3 . '{"address":{"street":"100 Park
 Ave","city":"Seattle","state":"WA","zipcode":"98102"}}'
OK

127.0.0.1:6379> JSON.MGET k1 k2 k3 .address.city
1) "\"New York\""
2) "\"Seattle\""
3) "\"Seattle\""
```

JSON.NUMINCRBY

指定された数だけパスの数値を増分します。

構文

```
JSON.NUMINCRBY <key> <path> <number>
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス (必須) - JSON パス
- 番号 (必須) — 数値

戻る

パスが拡張構文の場合:

- 各パスの結果値を表す一括文字列の配列。
- 値が数値でない場合、対応する戻り値は null です。
- 番号を解析できない場合は、WRONGTYPE エラーになります。
- 結果が 64 ビット IEEE 倍精度の範囲外の場合は、OVERFLOW エラーになります。
- ドキュメントキーが存在しない場合は、NONEXISTENT エラーになります。

パスが制限構文の場合:

- 結果の値を表す一括文字列。
- 複数の値を選択した場合、コマンドは最後に更新された値の結果を返します。
- パスの値が数値でない場合は、WRONGTYPE エラーになります。
- 番号を解析できない場合は、WRONGTYPE エラーになります。
- 結果が 64 ビット IEEE 倍精度の範囲外の場合は、OVERFLOW エラーになります。
- ドキュメントキーが存在しない場合は、NONEXISTENT エラーになります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 $.d[*] 10
"[11,12,13]"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[11,12,13]}"

127.0.0.1:6379> JSON.SET k1 $ '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 $.a[*] 1
"[]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.b[*] 1
"[2]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.c[*] 1
"[2,3]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.d[*] 1
"[2,3,4]"
```

```
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[2,3,4]}"

127.0.0.1:6379> JSON.SET k2 $ '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k2 $.a.* 1
"[]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.b.* 1
"[2]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.c.* 1
"[2,3]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.d.* 1
"[2,3,4]"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":2,\"b\":3,\"c\":4}}"

127.0.0.1:6379> JSON.SET k3 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k3 $.a.* 1
"[null]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.b.* 1
"[null,2]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.c.* 1
"[null,null]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.d.* 1
"[2,null,4]"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\",\"b\":2},\"c\":{\"a\":\"a\",\"b\":\"b\"},\"d
\":{\"a\":2,\"b\":\"b\",\"c\":4}}"
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 .d[1] 10
"12"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[1,12,3]}"

127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
```

```
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 .a[*] 1
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMINCRBY k1 .b[*] 1
"2"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[1,2],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMINCRBY k1 .c[*] 1
"3"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMINCRBY k1 .d[*] 1
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[2,3,4]}"

127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k2 .a.* 1
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMINCRBY k2 .b.* 1
"2"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":1,\"b\":2},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMINCRBY k2 .c.* 1
"3"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMINCRBY k2 .d.* 1
"4"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":2,\"b\":3,\"c\":4}}"

127.0.0.1:6379> JSON.SET k3 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k3 .a.* 1
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .b.* 1
"2"
127.0.0.1:6379> JSON.NUMINCRBY k3 .c.* 1
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .d.* 1
```

```
"4"
```

JSON.NUMMULTBY

指定された数だけパスの数値を乗算します。

構文

```
JSON.NUMMULTBY <key> <path> <number>
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス (必須) - JSON パス
- 番号 (必須) — 数値

戻る

パスが拡張構文の場合:

- 各パスの結果値を表す一括文字列の配列。
- 値が数値でない場合、対応する戻り値は null です。
- 番号を解析できない場合は、WRONGTYPE エラーになります。
- 結果が 64 ビット IEEE 倍精度の範囲外の場合は、OVERFLOW エラーになります。
- ドキュメントキーが存在しない場合は、NONEXISTENT エラーになります。

パスが制限構文の場合:

- 結果の値を表す一括文字列。
- 複数の値を選択した場合、コマンドは最後に更新された値の結果を返します。
- パスの値が数値でない場合は、WRONGTYPE エラーになります。
- 番号を解析できない場合は、WRONGTYPE エラーになります。
- 結果が 64 ビット IEEE 倍精度の範囲外の場合は、OVERFLOW エラーになります。
- ドキュメントキーが存在しない場合は、NONEXISTENT エラーになります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 $.d[*] 2
"[2,4,6]"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[2,4,6]}"

127.0.0.1:6379> JSON.SET k1 $ '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 $.a[*] 2
"[]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.b[*] 2
"[2]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.c[*] 2
"[2,4]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.d[*] 2
"[2,4,6]"

127.0.0.1:6379> JSON.SET k2 $ '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k2 $.a.* 2
"[]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.b.* 2
"[2]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.c.* 2
"[2,4]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.d.* 2
"[2,4,6]"

127.0.0.1:6379> JSON.SET k3 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k3 $.a.* 2
"[null]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.b.* 2
"[null,2]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.c.* 2
"[null,null]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.d.* 2
"[2,null,6]"
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 .d[1] 2
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[1,4,3]}"

127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 .a[*] 2
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMMULTBY k1 .b[*] 2
"2"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[1,2],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMMULTBY k1 .c[*] 2
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,4],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMMULTBY k1 .d[*] 2
"6"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,4],\"d\":[2,4,6]}"

127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k2 .a.* 2
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMMULTBY k2 .b.* 2
"2"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":1,\"b\":2},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k2 .c.* 2
"4"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":4},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k2 .d.* 2
"6"
```

```
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":4},\"d\":{\"a\":2,\"b\":4,\"c\":6}}"

127.0.0.1:6379> JSON.SET k3 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k3 .a.* 2
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMMULTBY k3 .b.* 2
"2"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\",\"b\":2},\"c\":{\"a\":\"a\",\"b\":\"b\"},\"d
\":{\"a\":1,\"b\":\"b\",\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k3 .c.* 2
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMMULTBY k3 .d.* 2
"6"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\",\"b\":2},\"c\":{\"a\":\"a\",\"b\":\"b\"},\"d
\":{\"a\":2,\"b\":\"b\",\"c\":6}}"
```

JSON.OBJLEN

パスにあるオブジェクト値のキーの数を取得します。

構文

```
JSON.OBJLEN <key> [path]
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス「オプショナル」– JSON パス。指定しない場合、デフォルトでルートになります

戻る

パスが拡張構文の場合:

- 各パスのオブジェクトの長さを表す整数の配列。
- 値がオブジェクトでない場合、対応する戻り値は null です。
- ドキュメントキーが存在しない場合は、null になります。

パスが制限構文の場合:

- 整数、オブジェクト内のキーの数。
- 複数のオブジェクトが選択されている場合、このコマンドは最初のオブジェクトの長さを返します。
- パスの値がオブジェクトでない場合は、WRONGTYPE エラーになります。
- パスが存在しない場合は、WRONGTYPE エラーになります。
- ドキュメントキーが存在しない場合は、null になります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJLEN k1 $.a
1) (integer) 0
127.0.0.1:6379> JSON.OBJLEN k1 $.a.*
(empty array)
127.0.0.1:6379> JSON.OBJLEN k1 $.b
1) (integer) 1
127.0.0.1:6379> JSON.OBJLEN k1 $.b.*
1) (nil)
127.0.0.1:6379> JSON.OBJLEN k1 $.c
1) (integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 $.c.*
1) (nil)
2) (nil)
127.0.0.1:6379> JSON.OBJLEN k1 $.d
1) (integer) 3
127.0.0.1:6379> JSON.OBJLEN k1 $.d.*
1) (nil)
2) (nil)
3) (integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 $.*
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 3
5) (nil)
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJLEN k1 .a
(integer) 0
127.0.0.1:6379> JSON.OBJLEN k1 .a.*
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.OBJLEN k1 .b
(integer) 1
127.0.0.1:6379> JSON.OBJLEN k1 .b.*
(error) WRONGTYPE JSON element is not an object
127.0.0.1:6379> JSON.OBJLEN k1 .c
(integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 .c.*
(error) WRONGTYPE JSON element is not an object
127.0.0.1:6379> JSON.OBJLEN k1 .d
(integer) 3
127.0.0.1:6379> JSON.OBJLEN k1 .d.*
(integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 .*
(integer) 0
```

JSON.OBJKEYS

パスにあるオブジェクト値のキー名を取得します。

構文

```
JSON.OBJKEYS <key> [path]
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス「オプショナル」– JSON パス。指定しない場合、デフォルトでルートになります

戻る

パスが拡張構文の場合:

- 一括文字列の配列の配列。各要素は、一致するオブジェクト内のキーの配列です。
- 値がオブジェクトでない場合、対応する戻り値は空の値です。
- ドキュメントキーが存在しない場合は、null になります。

パスが制限構文の場合:

- 一括文字列の配列。各要素はオブジェクトのキー名です。
- 複数のオブジェクトが選択されている場合、このコマンドは最初のオブジェクトのキーを返します。
- パスの値がオブジェクトでない場合は、WRONGTYPE エラーになります。
- パスが存在しない場合は、WRONGTYPE エラーになります。
- ドキュメントキーが存在しない場合は、null になります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJKEYS k1 $.*
1) (empty array)
2) 1) "a"
3) 1) "a"
   2) "b"
4) 1) "a"
   2) "b"
   3) "c"
5) (empty array)
127.0.0.1:6379> JSON.OBJKEYS k1 $.d
1) 1) "a"
   2) "b"
   3) "c"
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
```

```
OK
127.0.0.1:6379> JSON.OBJKEYS k1 .*
1) "a"
127.0.0.1:6379> JSON.OBJKEYS k1 .d
1) "a"
2) "b"
3) "c"
```

JSON.RESP

Valkey または Redis OSS の Serialization Protocol (RESP) で指定されたパスの JSON 値を返します。値がコンテナの場合、応答は RESP 配列またはネストされた配列になります。

- JSON null は、RESP Null 一括文字列にマップされます。
- JSON ブール値は、それぞれの RESP 単純文字列にマッピングされます。
- 整数は RESP 整数にマップされます。
- 64 ビット IEEE 倍精度浮動小数点数は、RESP 一括文字列にマッピングされます。
- JSON 文字列は、RESP 一括文字列にマッピングされます。
- JSON 配列は RESP 配列として表されます。最初の要素は単純な文字列で、その後に配列の要素が続きます。
- JSON オブジェクトは RESP 配列として表されます。最初の要素は単純な文字列 { で、その後にキーと値のペアが続きます。それぞれが RESP 一括文字列です。

構文

```
JSON.RESP <key> [path]
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス「オプショナル」– JSON パス。指定しない場合、デフォルトでルートになります

戻る

パスが拡張構文の場合:

- 配列の配列。各配列要素は、1 つのパスにおける値の RESP 形式を表します。
- ドキュメントキーが存在しない場合は、空の配列になります。

パスが制限構文の場合:

- パスの値の RESP 形式を表す配列。
- ドキュメントキーが存在しない場合は、null になります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK

127.0.0.1:6379> JSON.RESP k1 $.address
1) 1) {
   2) 1) "street"
      2) "21 2nd Street"
   3) 1) "city"
      2) "New York"
   4) 1) "state"
      2) "NY"
   5) 1) "zipcode"
      2) "10021-3100"

127.0.0.1:6379> JSON.RESP k1 $.address.*
1) "21 2nd Street"
2) "New York"
3) "NY"
4) "10021-3100"

127.0.0.1:6379> JSON.RESP k1 $.phoneNumbers
1) 1) [
   2) 1) {
      2) 1) "type"
         2) "home"
      3) 1) "number"
         2) "555 555-1234"
   3) 1) {
```

```
      2) 1) "type"
         2) "office"
      3) 1) "number"
         2) "555 555-4567"

127.0.0.1:6379> JSON.RESP k1 $.phoneNumbers[*]
1) 1) {
   2) 1) "type"
      2) "home"
   3) 1) "number"
      2) "212 555-1234"
2) 1) {
   2) 1) "type"
      2) "office"
   3) 1) "number"
      2) "555 555-4567"
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK

127.0.0.1:6379> JSON.RESP k1 .address
1) {
2) 1) "street"
   2) "21 2nd Street"
3) 1) "city"
   2) "New York"
4) 1) "state"
   2) "NY"
5) 1) "zipcode"
   2) "10021-3100"

127.0.0.1:6379> JSON.RESP k1
 1) {
 2) 1) "firstName"
    2) "John"
```

```
 3) 1) "lastName"
    2) "Smith"
 4) 1) "age"
    2) (integer) 27
 5) 1) "weight"
    2) "135.25"
 6) 1) "isAlive"
    2) true
 7) 1) "address"
    2) 1) {
       2) 1) "street"
          2) "21 2nd Street"
       3) 1) "city"
          2) "New York"
       4) 1) "state"
          2) "NY"
       5) 1) "zipcode"
          2) "10021-3100"
 8) 1) "phoneNumbers"
    2) 1) [
       2) 1) {
          2) 1) "type"
             2) "home"
          3) 1) "number"
             2) "212 555-1234"
       3) 1) {
          2) 1) "type"
             2) "office"
          3) 1) "number"
             2) "555 555-4567"
 9) 1) "children"
    2) 1) [
10) 1) "spouse"
    2) (nil)
```

JSON.SET

パスに JSON 値を設定します。

パスがオブジェクトメンバーを要求する場合:

- 親要素が存在しない場合、このコマンドは NONEXISTENT エラーを返します。

- 親要素は存在するがオブジェクトではない場合、このコマンドは ERROR を返します。
- 親要素が存在し、オブジェクトである場合:
  - メンバーが存在しない場合、親オブジェクトがパスの最後の子である場合にのみ、新しいメンバーが親オブジェクトに追加されます。それ以外の場合、このコマンドは NONEXISTENT エラーを返します。
  - メンバーが存在する場合、その値は JSON 値に置き換えられます。

パスが配列インデックスを要求する場合:

- 親要素が存在しない場合、このコマンドは NONEXISTENT エラーを返します。
- 親要素は存在するが配列ではない場合、このコマンドは ERROR を返します。
- 親要素は存在するが、インデックスが範囲外である場合、このコマンドは OUTOFBOUNDARIES エラーを返します。
- 親要素が存在し、インデックスが有効な場合、要素は新しい JSON 値に置き換えられます。

パスがオブジェクトまたは配列を要求する場合、値 (オブジェクトまたは配列) は新しい JSON 値に置き換えられます。

構文

```
JSON.SET <key> <path> <json> [NX | XX]
```

NX | XX ここで、NX | XX の識別子を 0 個または 1 個持つことができます

- キー (必須) - JSON ドキュメントタイプのキー
- パス (必須) - JSON パス。新しいキーの場合、JSON パスはルート「.」でなければなりません。
- NX (オプション) - パスがルートである場合は、キーが存在しない場合にのみ値を設定します。つまり、新しいドキュメントを挿入します。パスがルートではない場合は、パスが存在しない場合にのみ値を設定します。つまり、ドキュメントに値を挿入します。
- XX (オプション) - パスがルートの場合は、キーが存在する場合にのみ値を設定します。つまり、既存のドキュメントを置き換えます。パスがルートではない場合は、パスが存在する場合にのみ値を設定します。つまり、既存の値を更新します。

戻る

- 成功した場合は、シンプルな文字列「OK」が返されます。
- NX または XX 条件が満たされない場合は、null が返されます。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{"a":1, "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.SET k1 $.a.* '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"a\":{\"a\":0,\"b\":0,\"c\":0}}"

127.0.0.1:6379> JSON.SET k2 . '{"a": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.SET k2 $.a[*] '0'
OK
127.0.0.1:6379> JSON.GET k2
"{\"a\":[0,0,0,0,0]}"
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"c":{"a":1, "b":2}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.SET k1 .c.a '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"c\":{\"a\":0,\"b\":2},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.SET k1 .e[-1] '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"c\":{\"a\":0,\"b\":2},\"e\":[1,2,3,4,0]}"
127.0.0.1:6379> JSON.SET k1 .e[5] '0'
(error) OUTOFBOUNDARIES Array index is out of bounds
```

JSON.STRAPPEND

パスの JSON 文字列に文字列を追加します。

構文

```
JSON.STRAPPEND <key> [path] <json_string>
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス「オプショナル」– JSON パス。指定しない場合、デフォルトでルートになります
- json_string (必須) — 文字列の JSON 表現。JSON 文字列は引用符で囲む ("foo") 必要があることに注意してください。

戻る

パスが拡張構文の場合:

- 各パスの文字列の新しい長さを表す整数の配列。
- パスの値が文字列でない場合、対応する戻り値は null です。
- 入力された json 引数が有効な JSON 文字列でない場合は、SYNTAXERR エラーになります。
- パスが存在しない場合は、NONEXISTENT エラーになります。

パスが制限構文の場合:

- 整数、文字列の新しい長さ。
- 複数の文字列値が選択されている場合、このコマンドは最後に更新された文字列の新しい長さを返します。
- パスの値が文字列でない場合は、WRONGTYPE エラーになります。
- 入力された json 引数が有効な JSON 文字列でない場合は、WRONGTYPE エラーになります。
- NONEXISTENTパスが存在しない場合は、 エラーになります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRAPPEND k1 $.a.a '"a"'
1) (integer) 2
```

```
127.0.0.1:6379> JSON.STRAPPEND k1 $.a.* '"a"'
1) (integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 $.b.* '"a"'
1) (integer) 2
2) (nil)
127.0.0.1:6379> JSON.STRAPPEND k1 $.c.* '"a"'
1) (integer) 2
2) (integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 $.c.b '"a"'
1) (integer) 4
127.0.0.1:6379> JSON.STRAPPEND k1 $.d.* '"a"'
1) (nil)
2) (integer) 2
3) (nil)
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRAPPEND k1 .a.a '"a"'
(integer) 2
127.0.0.1:6379> JSON.STRAPPEND k1 .a.* '"a"'
(integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 .b.* '"a"'
(integer) 2
127.0.0.1:6379> JSON.STRAPPEND k1 .c.* '"a"'
(integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 .c.b '"a"'
(integer) 4
127.0.0.1:6379> JSON.STRAPPEND k1 .d.* '"a"'
(integer) 2
```

JSON.STRLEN

パスの JSON 文字列値の長さを取得します。

構文

```
JSON.STRLEN <key> [path]
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス「オプショナル」– JSON パス。指定しない場合、デフォルトでルートになります

戻る

パスが拡張構文の場合:

- 各パスの文字列値の長さを表す整数の配列。
- 値が文字列でない場合、対応する戻り値は null です。
- ドキュメントキーが存在しない場合は、null になります。

パスが制限構文の場合:

- 整数、文字列の長さ。
- 複数の文字列値が選択されている場合、このコマンドは最初の文字列の長さを返します。
- パスの値が文字列でない場合は、WRONGTYPE エラーになります。
- パスが存在しない場合は、NONEXISTENT エラーになります。
- ドキュメントキーが存在しない場合は、null になります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRLEN k1 $.a.a
1) (integer) 1
127.0.0.1:6379> JSON.STRLEN k1 $.a.*
1) (integer) 1
127.0.0.1:6379> JSON.STRLEN k1 $.c.*
1) (integer) 1
2) (integer) 2
127.0.0.1:6379> JSON.STRLEN k1 $.c.b
1) (integer) 2
127.0.0.1:6379> JSON.STRLEN k1 $.d.*
1) (nil)
```

```
2) (integer) 1
3) (nil)
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRLEN k1 .a.a
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .a.*
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .c.*
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .c.b
(integer) 2
127.0.0.1:6379> JSON.STRLEN k1 .d.*
(integer) 1
```

JSON.TOGGLE

パスのブール値を true と false の間で切り替えます。

構文

```
JSON.TOGGLE <key> [path]
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス「オプショナル」– JSON パス。指定しない場合、デフォルトでルートになります

戻る

パスが拡張構文の場合:

- 各パスの結果のブール値を表す整数 (0 - false、1 - true) の配列。
- 値がブール値でない場合は、対応する戻り値は null です。
- ドキュメントキーが存在しない場合は、NONEXISTENT エラーになります。

パスが制限構文の場合:

- 結果のブール値を表す文字列 (「true」/「false」)。
- ドキュメントキーが存在しない場合は、NONEXISTENT エラーになります。
- パスの値がブール値でない場合は、WRONGTYPE エラーになります。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":true, "b":false, "c":1, "d":null, "e":"foo", "f":
[], "g":{}}'
OK
127.0.0.1:6379> JSON.TOGGLE k1 $.*
1) (integer) 0
2) (integer) 1
3) (nil)
4) (nil)
5) (nil)
6) (nil)
7) (nil)
127.0.0.1:6379> JSON.TOGGLE k1 $.*
1) (integer) 1
2) (integer) 0
3) (nil)
4) (nil)
5) (nil)
6) (nil)
7) (nil)
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 . true
OK
127.0.0.1:6379> JSON.TOGGLE k1
"false"
127.0.0.1:6379> JSON.TOGGLE k1
"true"

127.0.0.1:6379> JSON.SET k2 . '{"isAvailable": false}'
```

```
OK
127.0.0.1:6379> JSON.TOGGLE k2 .isAvailable
"true"
127.0.0.1:6379> JSON.TOGGLE k2 .isAvailable
"false"
```

JSON.TYPE

指定されたパスの値の型を報告します。

構文

```
JSON.TYPE <key> [path]
```

- キー (必須) - JSON ドキュメントタイプのキー
- パス「オプショナル」– JSON パス。指定しない場合、デフォルトでルートになります

戻る

パスが拡張構文の場合:

- 各パスの値の型を表す文字列の配列。型は、{「null」、「boolean」、「string」、「number」、「integer」、「object」、および「array」}のいずれかです。
- パスが存在しない場合、対応する戻り値は null です。
- ドキュメントキーが存在しない場合は、空の配列になります。

パスが制限構文の場合:

- 文字列、値の型
- ドキュメントキーが存在しない場合は、null になります。
- JSON パスが無効であるか、存在しない場合は null です。

例

拡張パス構文:

```
127.0.0.1:6379> JSON.SET k1 . '[1, 2.3, "foo", true, null, {}, []]'
```

```
OK
127.0.0.1:6379> JSON.TYPE k1 $[*]
1) integer
2) number
3) string
4) boolean
5) null
6) object
7) array
```

制限パス構文:

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.TYPE k1
object
127.0.0.1:6379> JSON.TYPE k1 .children
array
127.0.0.1:6379> JSON.TYPE k1 .firstName
string
127.0.0.1:6379> JSON.TYPE k1 .age
integer
127.0.0.1:6379> JSON.TYPE k1 .weight
number
127.0.0.1:6379> JSON.TYPE k1 .isAlive
boolean
127.0.0.1:6379> JSON.TYPE k1 .spouse
null
```

MemoryDB リソースのタグ付け

クラスターと他の MemoryDB リソースを管理しやすくするために、タグ形式で各リソースに独自のメタデータを割り当てることができます。タグを使用すると、例えば用途別、所有者別、環境別などのさまざまな方法で AWS リソースを分類できます。これは同じタイプのリソースが多数ある場合に

役立ちます。割り当てたタグに基づいて、特定のリソースをすばやく識別できます。ここではタグとその作成方法について説明します。

> ⚠ Warning
>
> ベストプラクティスとして、機密データをタグに含めないようお勧めします。

タグの基本

タグとはAWS リソースに割り当てるラベルです。タグはそれぞれ、1 つのキーとオプションの 1 つの値で設定されており、どちらもお客様側が定義します。タグを使用すると、AWS リソースを用途、所有者などのさまざまな方法で分類できます。例えば、各インスタンスの所有者とユーザーグループを追跡しやすくするため、アカウントの MemoryDB クラスターに対して一連のタグを定義できます。

各リソースタイプのニーズを満たす一連のタグキーを考案することをお勧めします。一貫性のある一連のタグキーを使用することで、リソースの管理が容易になります。追加したタグに基づいてリソースを検索およびフィルタリングできます。効果的なリソースのタグ付け戦略を実装する方法の詳細については、「AWS ホワイトペーパーのタグ付けのベストプラクティス」を参照してください。

タグには、MemoryDB に関連する意味はなく、完全に文字列として解釈されます。また、タグは自動的にリソースに割り当てられます。タグのキーと値は編集でき、タグはリソースからいつでも削除できます。タグの値は `null` に設定できます。特定のリソースについて既存のタグと同じキーを持つタグを追加した場合、以前の値は新しい値によって上書きされます。リソースを削除すると、リソースのタグも削除されます。

AWS マネジメントコンソール、AWS CLI、および MemoryDB API を使用してタグを操作できます。

IAM を使用している場合は、タグを作成、編集、削除する許可を持つ AWS アカウントのユーザーを制御できます。詳細については、「リソースレベルのアクセス許可」を参照してください。

タグを付けることができるリソース

アカウントに既に存在するほとんどの MemoryDB リソースにタグ付けできます。以下の表に、タグ付けをサポートするリソースを示します。AWS マネジメントコンソール を使用している場合、リソースにタグを適用するには、タグエディタ を使用します。一部のリソースの画面では、リソースの作成時にリソースのタグを指定できます。例えば、Name のキーと指定した値をタグ付けしま

す。ほとんどの場合、リソースの作成後すぐに (リソースの作成時ではなく) コンソールによりタグが適用されます。コンソールではリソースを [名前] タグに応じて整理できますが、このタグには MemoryDB サービスに対する意味論的意味はありません。

さらに、リソース作成アクションによってはリソースの作成時にリソースのタグを指定できます。リソースの作成時にタグを適用できない場合はリソース作成プロセスがロールバックされます。これにより、リソースがタグ付きで作成されるか、まったく作成されないようになるため、タグ付けされていないリソースが存在することがなくなります。作成時にリソースにタグ付けすることで、リソース作成後にカスタムタグ付けスクリプティングを実行する必要がなくなります。

Amazon MemoryDB API、AWS CLI、または AWS SDK を使用している場合は、関連する MemoryDB API アクションの `Tags` パラメータを使用して、タグを適用できます。具体的には次の 2 つです。

- `CreateCluster`
- `CopySnapshot`
- `CreateParameterGroup`
- `CreateSubnetGroup`
- `CreateSnapshot`
- `CreateACL`
- `CreateUser`
- `CreateMultiRegionCluster`

次の表では、タグ付け可能な MemoryDB リソースと、MemoryDB API、AWS CLI、または AWS SDK を使用して作成時にタグ付け可能なリソースについて説明します。

MemoryDB リソースのタグ付けのサポート

タグをサポート	作成時のタグ付けをサポート
はい	はい
はい	はい

タグをサポート	作成時のタグ付けをサポート
はい	はい
はい	はい
はい	はい
はい	はい
はい	はい

作成時のタグ付けをサポートする MemoryDB API アクションに、IAM ポリシーのタグベースでリソースレベルの許可を適用し、作成時のリソースタグ付けができるユーザーとグループを細かくコントロールできます。リソースは、作成時から適切に保護されます。タグはリソースに即座に適用されます。したがって、リソースの使用を制御するタグベースのリソースレベルの許可は、ただちに有効になります。リソースはより正確に追跡および報告されます。新しいリソースにタグ付けの使用を適用し、リソースで設定されるタグキーと値をコントロールできます。

詳細については、「リソースのタグ付けの例」を参照してください。

請求用のリソースへのタグ付けの詳細については、「コスト配分タグによるコストのモニタリング」を参照してください。

クラスターとスナップショット、およびマルチリージョンクラスターのタグ付け

リクエストオペレーションの一部としてタグ付けには、次のルールが適用されます。

- CreateCluste:
  - `--cluster-name` が供給された場合:

    リクエストにタグが含まれている場合、クラスターにはタグ付けされます。
  - `--snapshot-name` が供給された場合:

    タグがリクエストに含まれている場合、クラスターはそれらのタグのみでタグ付けされます。タグがリクエストに含まれていない場合、スナップショットタグはクラスターに追加されます。
- CreateSnapshot:
  - `--cluster-name` が供給された場合:

    タグがリクエストに含まれている場合、リクエストタグのみがスナップショットに追加されます。タグがリクエストに含まれていない場合、クラスタータグがスナップショットに追加されます。
  - 自動スナップショットでは:

    タグは、クラスタータグから伝播されます。
- CopySnapshot:

  タグがリクエストに含まれている場合、リクエストタグのみがスナップショットに追加されます。タグがリクエストに含まれていない場合、コピー元のスナップショットタグがコピーされたスナップショットに追加されます。
- TagResource、UntagResource :

  タグはリソースに追加されたり、リソースから削除されたりします。

マルチリージョンクラスターのタグ付け

MemoryDB マルチリージョンクラスターはグローバルリソースです。そのため、MemoryDB マルチリージョンがサポートされている任意のリージョンで関連する API を呼び出すことで、マルチリージョンクラスターでタグを指定、変更、または一覧表示できます。リージョンサポートの詳細については、「前提条件と制限」を参照してください。

マルチリージョンクラスターのタグは、リージョンクラスターのタグから独立しています。マルチリージョンクラスターには、リージョンクラスターのタグを含むさまざまなタグセットを指定できます。これらのタグ間に階層的なつながりはなく、これらのリソースタイプ間の階層を通じてコピーされません。

TagResource および UntagResource API を使用してタグを追加または削除した場合、ListTags API のレスポンスに最新の有効なタグがすぐに表示されない可能性があります。これは、特にマルチリージョンクラスターの場合に、タグの整合性が最終的に確保されるためです。

タグの制限

タグには以下のような基本制限があります。

- リソースあたりのタグの最大数 - 50 件
- タグキーはリソースごとにそれぞれ一意である必要があります。また、各タグキーに設定できる値は 1 つのみです。
- キーの最大長 – 128 Unicode 文字 (UTF-8)
- 値の最大長 - 256 Unicode 文字 (UTF-8)。
- MemoryDB ではタグ内に任意の文字を使用できますが、他のサービスでは制限がある場合があります。すべてのサービスで使用できる文字は、UTF-8 で表現できる文字、数字、およびスペースに加えて、 + - = . _ : / @ です。
- タグのキーと値は大文字と小文字が区別されます。
- aws: プレフィックスは AWS 用に限定されています。タグにこのプレフィックスが付いたタグキーがある場合、タグのキーまたは値を編集、削除することはできません。aws: プレフィックスを持つタグはリソースあたりのタグ数の制限時には計算されません。

タグのみに基づいてリソースを終了、停止、終了することはできません。リソース識別子を指定する必要があります。例えば、DeleteMe というタグキーを使用してタグ付けしたスナップショットを削除するにはDeleteSnapshot のようなスナップショットのリソース識別子を指定して snap-1234567890abcdef0 アクションを使用する必要があります。

タグ付けできる MemoryDB リソースの詳細については、「タグを付けることができるリソース」を参照してください。

リソースのタグ付けの例

- 新しいクラスターにタグを追加する。

```
aws memorydb tag-resource \
--resource-arn arn:aws:memorydb:us-east-1:111111222233:cluster/my-cluster \
--tags Key="project",Value="XYZ" Key="memorydb",Value="Service"
```

- タグを使用したクラスターの作成。

```
aws memorydb create-cluster \
--cluster-name testing-tags \
--description cluster-test \
--subnet-group-name test \
--node-type db.r6g.large \
--acl-name open-access \
--tags Key="project",Value="XYZ" Key="memorydb",Value="Service"
```

- タグ付きのスナップショットを作成します。

  この場合、リクエストでタグを追加すると、クラスターにタグが含まれている場合でも、スナップショットはリクエストタグのみを受け取ります。

```
aws memorydb create-snapshot \
--cluster-name testing-tags \
--snapshot-name bkp-testing-tags-mycluster \
--tags Key="work",Value="foo"
```

コスト配分タグによるコストのモニタリング

MemoryDB でリソースにコスト配分タグを追加する場合、リソースのタグ値に基づいて請求書の費用をグループ化してコストを追跡できます。

MemoryDB コスト配分タグは、MemoryDB リソースを定義してそのリソースに関連付けるキーと値のペアです。キーと値は大文字と小文字が区別されます。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の項目にすることができます。例えば、「CostCenter」というタグキーと「10010」というタグ値を定義して、リソースがコストセンター 10010 に割り当てられていることを示すことができます。また、Environment などのキーと、test や production などの値を使用して、リソースがテスト用なのか本稼働用なのかを示すこともできます。リソースに関連付けられているコストの追跡が簡単になるように、一貫した一連のタグキーを使用することをお勧めします。

コスト配分タグを使用して AWS の請求書を分類し、自分のコスト構造を反映させます。そのためには、サインアップして、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必要があります。次に、結合したリソースのコストを見るには、同じタグキー値のリソースに従って請求書情報を整理します。例えば、複数のリソースに特定のアプリケーション名のタグを付け、請求情報を整理することで、複数のサービスを利用しているアプリケーションの合計コストを確認することができます。

タグを組み合わせてさらに細かくコストを追跡することもできます。例えば、リージョンごとのサービスのコストを追跡するために、Service と Region というタグキーを使用できます。1 つのリソースでは値を MemoryDB と Asia Pacific (Singapore) にし、別のリソースでは値を MemoryDB と Europe (Frankfurt) にします。これによって、MemoryDB の合計コストをリージョンごとに表示できます。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」(Use Cost Allocation Tags) を参照してください。

Memcached クラスターに MemoryDB コスト配分タグを追加できます。タグの追加やリスト、変更、削除を行った場合、そのオペレーションは、指定したクラスターにのみ適用されます。

MemoryDB コスト配分タグの特徴

- コスト配分タグは、ARN として CLI および API オペレーションで指定された MemoryDB リソースに適用されます。resource-type は "cluster" です。

  ARN 形式: `arn:aws:memorydb:<region>:<customer-id>:<resource-type>/<resource-name>`

  サンプル ARN: `arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`
- タグキーは、必須のタグ名です。キーの文字列値は、長さが 1 ~ 128 文字の Unicode 文字です。aws: をプレフィックスとして使用することはできません。文字列には、一連の Unicode 文字、数字、空白、下線 (_)、ピリオド (.)、コロン (:)、バックスラッシュ (\ )、等号 (=)、プラス記号 (+)、ハイフン (-)、またはアットマーク (@) を含めることができます。
- タグ値は、オプションのタグの値です。値の文字列値は、長さが 1 ~ 256 文字の Unicode 文字です。aws: をプレフィックスとして使用することはできません。文字列には、一連の Unicode 文字、数字、空白、下線 (_)、ピリオド (.)、コロン (:)、バックスラッシュ (\ )、等号 (=)、プラス記号 (+)、ハイフン (-)、またはアットマーク (@) を含めることができます。
- MemoryDB リソースには、最大 50 個のタグを設定できます。
- 値はタグセット内で一意である必要はありません。例えば、タグセット内に Service と Application というキーがあり、両方の値として MemoryDB を指定できます。

AWS はタグに意味論的意味を適用しません。タグは厳密に文字列として解釈されます。AWS は MemoryDB リソースにタグを自動的に設定しません。

AWS CLI を使用したコスト配分タグの管理

AWS CLI を使用して、コスト配分タグを追加、変更、または削除できます。

サンプル arn: `arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`

トピック

- AWS CLI を使用したタグの一覧表示
- AWS CLI を使用したタグの追加
- AWS CLI を使用したタグの変更
- AWS CLI を使用したタグの削除

AWS CLI を使用したタグの一覧表示

list-tags オペレーションを使用すると、AWS CLI を使用して既存のMemoryDB リソースのタグを一覧表示できます。

次のコードは、AWS CLI を使用して、us-east-1 リージョンの MemoryDB クラスター `my-cluster` のタグをリスト表示します。

Linux、macOS、Unix の場合:

```
aws memorydb list-tags \
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
```

Windows の場合:

```
aws memorydb list-tags ^
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
```

このオペレーションの出力は、リソースのすべてのタグを示した次のリストのようになります。

```
{
   "TagList": [
      {
         "Value": "10110",
         "Key": "CostCenter"
      },
      {
         "Value": "EC2",
         "Key": "Service"
      }
   ]
```

```
}
```

リソースにタグが見つからない場合は、空の TagList が出力されます。

```
{
   "TagList": []
}
```

詳細については、MemoryDB の AWS CLI の list-tags を参照してください。

AWS CLI を使用したタグの追加

tag-resource CLI オペレーションを行い、AWS CLI を使用して、既存の MemoryDB リソースにタグを追加できます。タグキーがリソースに存在しない場合は、キーと値がリソースに追加されます。キーが既にリソースに存在する場合、キーに関連付けられた値は新しい値に更新されます。

次のコードは、AWS CLI を使用して、それぞれ `memorydb` と `us-east-1` の値を持つキー `Service` と `Region` をus-east-1 リージョン内のクラスター `my-cluster` に追加します。

Linux、macOS、Unix の場合:

```
aws memorydb tag-resource \
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster \
 --tags Key=Service,Value=memorydb \
        Key=Region,Value=us-east-1
```

Windows の場合:

```
aws memorydb tag-resource ^
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster ^
 --tags Key=Service,Value=memorydb ^
        Key=Region,Value=us-east-1
```

このオペレーションの出力は、次のオペレーションのリソースのすべてのタグを示した以下のリストのようになります。

```
{
   "TagList": [
      {
         "Value": "memorydb",
```

```
         "Key": "Service"
      },
      {
         "Value": "us-east-1",
         "Key": "Region"
      }
   ]
}
```

詳細については、「MemoryDB tag-resource の AWS CLI」を参照してください。

オペレーション create-cluster を使用して新しいクラスターを作成するときに、AWS CLI を使用してクラスターにタグを追加することもできます。

AWS CLI を使用したタグの変更

AWS CLI を使用して、MemoryDB クラスターのタグを変更できます。

タグを変更するには:

- tag-resource を使用して、新しいタグと値を追加するか、既存のタグに関連付けられている値を変更します。
- 指定したタグをリソースから削除するには、untag-resource を使用します。

どちらのオペレーションでも、指定のクラスターのタグとその値を示すリストが出力されます。

AWS CLI を使用したタグの削除

AWS CLI を使用すると、untag-resource オペレーションを使用して MemoryDB クラスターからタグを削除できます。

次のコードは、AWS CLI を使用して、us-east-1 リージョンのクラスター `my-cluster` からキー `Service` と `Region` を含むタグを削除します。

Linux、macOS、Unix の場合:

```
aws memorydb untag-resource \
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster \
 --tag-keys Region Service
```

Windows の場合:

```
aws memorydb untag-resource ^
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster ^
 --tag-keys Region Service
```

このオペレーションの出力は、次のオペレーションのリソースのすべてのタグを示した以下のリストのようになります。

```
{
   "TagList": []
}
```

詳細については、MemoryDB の AWS CLI の untag-resource を参照してください。

MemoryDB API を使用したコスト配分タグの管理

MemoryDB API を使用して、コスト配分タグを追加、変更、または削除できます。

コスト配分タグは、MemoryDB 用 クラスターに適用されます。タグ付けされるクラスターは、ARN (Amazon リソースネーム) を使用して指定されます。

サンプル arn: `arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`

トピック

- MemoryDB API を使用したタグの一覧表示
- MemoryDB API を使用したタグの追加
- MemoryDB API を使用したタグの変更
- MemoryDB API を使用したタグの削除

MemoryDB API を使用したタグの一覧表示

ListTags オペレーションを使用すると、MemoryDB API を使用して既存のリソースのタグを一覧表示できます。

次のコードは、MemoryDB API を使用して、us-east-1 リージョンのリソース `my-cluster` のタグをリスト表示します。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=ListTags
   &ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
```

```
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Version=2021-01-01
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

MemoryDB API を使用したタグの追加

TagResource オペレーションを使用すると、MemoryDB API を使用して、既存の MemoryDB クラスターにタグを追加できます。タグキーがリソースに存在しない場合は、キーと値がリソースに追加されます。キーが既にリソースに存在する場合、キーに関連付けられた値は新しい値に更新されます。

次のコードは、MemoryDB API を使用して、us-east-1 リージョンのリソース `my-cluster` に、それぞれ値 `memorydb` と `us-east-1` を持つキー `Service` と `Region` を追加します。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=TagResource
   &ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Tags.member.1.Key=Service
   &Tags.member.1.Value=memorydb
   &Tags.member.2.Key=Region
   &Tags.member.2.Value=us-east-1
   &Version=2021-01-01
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

詳細については、「TagResource」を参照してください。

MemoryDB API を使用したタグの変更

MemoryDB API を使用して、MemoryDB クラスターのタグを変更できます。

タグの値を変更するには:

- TagResource オペレーションを使用して新しいタグと値を追加するか、既存のタグの値を変更します。
- リソースからタグを削除するには、 UntagResource を使用します。

どちらのオペレーションでも、指定のリソースのタグとその値を示すリストが出力されます。

MemoryDB API を使用したタグの削除

MemoryDB API を使用すると、UntagResource オペレーションを使用して既存の MemoryDB クラスターからタグを削除できます。

次のコードは、MemoryDB API を使用して、リージョン us-east-1 のクラスター `my-cluster` からキー `Service` と `Region` を持つタグを削除します。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=UntagResource
   &ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &TagKeys.member.1=Service
   &TagKeys.member.2=Region
   &Version=2021-01-01
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

メンテナンスの管理

すべてのクラスターには、週ごとのメンテナンス時間があります。その時間内にシステムの変更が適用されます。クラスターの作成または変更時に、希望するメンテナンスウィンドウを指定しない場合、MemoryDB では、ランダムに選択された曜日に対してリージョン内で 60 分のメンテナンスウィンドウを割り当てます。

60 分のメンテナンス時間は、リージョンごとに決められた 8 時間の中でランダムに選択されます。次の表に、デフォルトでメンテナンス時間が割り当てられる各リージョンの時間ブロックを示します。リージョンのメンテナンス時間外で、希望するメンテナンス時間を選択できます。

リージョンコード	リージョン名	リージョンメンテナンスウィンドウ
ap-northeast-1	アジアパシフィック (東京) リージョン	13:00 ~ 21:00 UTC
ap-northeast-2	アジアパシフィック (ソウル) リージョン	12:00 ~ 20:00 UTC

リージョンコード	リージョン名	リージョンメンテナンスウィンドウ
ap-south-1	アジアパシフィック (ムンバイ) リージョン	17:30 ~ 1:30 UTC
ap-southeast-1	アジアパシフィック (シンガポール) リージョン	14:00 ~ 22:00 UTC
ap-east-1	アジアパシフィック (香港) リージョン	13:00 ~ 21:00 UTC
ap-southeast-2	アジアパシフィック (シドニー) リージョン	12:00 ~ 20:00 UTC
cn-north-1	中国 (北京) リージョン	14:00 ~ 22:00 UTC
cn-northwest-1	中国 (寧夏) リージョン	14:00 ~ 22:00 UTC
eu-west-3	EU (パリ) リージョン	23:59 ~ 07:29 UTC
eu-central-1	欧州 (フランクフルト) リージョン	23:00 ~ 07:00 UTC
eu-west-1	欧州 (アイルランド) リージョン	22:00 ~ 06:00 UTC
eu-west-2	欧州 (ロンドン) リージョン	23:00 ~ 07:00 UTC
sa-east-1	南米 (サンパウロ) リージョン	01:00 ~ 09:00 UTC
ca-central-1	カナダ (中部) リージョン	03:00 ~ 11:00 UTC
us-east-1	米国東部 (バージニア北部) リージョン	03:00 ~ 11:00 UTC
us-east-1	米国東部 (オハイオ) リージョン	04:00 ~ 12:00 UTC
us-west-1	US West (N. California) リージョン	06:00 ~ 14:00 UTC
us-west-2	米国西部 (オレゴン) リージョン	06:00 ~ 14:00 UTC

クラスターのメンテナンスウィンドウの変更

メンテナンスウィンドウは使用率の最も低い時間帯に設定する必要があります。このため、場合によっては変更が必要になります。クラスターを変更して、リクエストしたメンテナンス作業が発生するまでの時間範囲 (最大 24 時間) を指定することができます。お客様がリクエストした延期または保留中のクラスターの変更は、この時間に行われます。

詳細情報

メンテナンスウィンドウとノード交換の詳細については、以下を参照してください。

- ノードの置換—ノード交換の管理
- MemoryDB クラスターの変更—クラスターのメンテナンスウィンドウの変更

ベストプラクティス

以下は、MemoryDB の推奨されるベストプラクティスです。これらに従うと、クラスターのパフォーマンスと信頼性が向上します。

トピック

- MemoryDB の耐障害性
- ベストプラクティス: Pub/Sub および拡張 I/O マルチプレクシング
- ベストプラクティス: オンラインクラスターのサイズ変更

MemoryDB 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として構築されています。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、およびスケーラビリティが優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、「AWS グローバルインフラストラクチャ」を参照してください。

AWS グローバルインフラストラクチャに加えて、MemoryDB は、データの耐障害性とスナップショットのニーズに対応できるように複数の機能を提供しています。

トピック

- 障害の軽減

障害の軽減

MemoryDB の実装を計画した場合、障害がアプリケーションやデータに及ぼす影響を最小限にとどめるように計画する必要があります。このセクションのトピックでは、アプリケーションおよびデータを障害から保護するために実行できるアプローチについて説明します。

障害の軽減:MemoryDB クラスター

MemoryDB クラスターは、アプリケーションの読み取りと書き込みが可能な単一のプライマリノードと、0～5 個の読み取り専用のレプリカノードで構成されます。ただし、高可用性を実現するには、少なくとも 1 つのレプリカを使用することを強くお勧めします。データがプライマリノードに書き込まれるときは、そのデータはトランザクションログに永続的に保存され、レプリカノードでデータが非同期的に更新されます。

リードレプリカが失敗した場合

1. MemoryDB が、障害の発生したレプリカを検出します。
2. MemoryDB が、障害のあるノードをオフラインにします。
3. MemoryDB が、同じ AZ の代替のノードを起動し、プロビジョニングします。

4. 新しいノードがトランザクションログと同期されます。

この間、アプリケーションは他のノードを使用して読み書きを続行できます。

MemoryDB マルチ AZ

マルチ AZ が MemoryDB クラスターでアクティブになっている場合、障害が発生したプライマリが検出され、自動的に置き換えられます。

1. MemoryDB がプライマリノードの失敗を検出します。
2. MemoryDB は、障害が発生したプライマリとの整合性を確認した後、レプリカにフェイルオーバーします。
3. MemoryDBは、障害が発生したプライマリの AZ のレプリカをスピンアップします。
4. 新しいノードがトランザクションログと同期されます。

レプリカノードへのフェイルオーバーは、通常、新しいプライマリノードを作成してプロビジョニングするより高速です。つまり、アプリケーションはプライマリノードへの書き込みをすばやく再開できることを意味します。

詳細については、「マルチ AZ による MemoryDB のダウンタイムの最小化」を参照してください。

ベストプラクティス: Pub/Sub および拡張 I/O マルチプレクシング

Valkey または Redis OSS バージョン 7 以降を使用する場合は、シャードされた Pub/Sub を使用することをお勧めします。また、拡張 I/O 多重化によってスループットとレイテンシーが向上します。これは、Valkey または Redis OSS バージョン 7 以降を使用すると自動的に利用可能になり、クライアント側の変更を必要としません。これは、多くの場合、複数のクライアント接続でスループットが制限される Pub/Sub ワークロードに最適です。

ベストプラクティス: オンラインクラスターのサイズ変更

リシャーディングには、クラスターへのシャードまたはノードの追加と削除、およびキースペースの再分散が含まれます。したがって、クラスターの負荷、メモリ使用率、データ全体のサイズなど、シャーディングオペレーションには複数のものが影響します。最適なエクスペリエンスを得るには、均一なワークロードパターンディストリビューションのクラスターベストプラクティス全体に従うことをお勧めします。さらに、次のステップを実行することをお勧めします。

リシャーディングを開始する前に、次のことをお勧めします:

- アプリケーションをテストする – 可能であれば、ステージング環境でリシャーディング中にアプリケーションの動作をテストします。
- スケーリング問題の早期通知の取得 – リシャーディングは計算処理能力を集中的に使用するオペレーションです。このため、リシャーディング中は CPU 使用率をマルチコアインスタンスで 80% 未満、シングルコアインスタンスで 50% 未満にすることをお勧めします。MemoryDB メトリックスをモニタリングして、アプリケーションでスケーリングの問題が発生する前にリシャーディングを開始します。追跡すると有用なメトリックスは、`CPUUtilization`、`NetworkBytesIn`、`NetworkBytesOut`、`CurrConnections`、`NewConnecti` です。
- スケーリングする前に、空きメモリが十分に確保されていることを確認する – スケーリングする場合、保持するシャードの空きメモリが、削除するシャードに使用されているメモリの 1.5 倍以上であることを確認します。
- オフピーク時にリシャーディングを開始する – このプラクティスは、リシャーディングオペレーションがクライアントのレイテンシーとスループットに与える影響を軽減するのに役立ちます。また、スロット再分散に多くのリソースを使用できるため、リシャーディングをより迅速に完了できます。
- クライアントのタイムアウト動作を確認する – オンラインクラスターのサイズ変更中に、一部のクライアントでレイテンシーが長くなる場合があります。より大きなタイムアウトでクライアントライブラリを設定すると、サーバーがより高い負荷条件でもシステムが接続する時間を与えるこ

とができます。場合によっては、サーバーへの接続を多数開く必要があります。この場合、エクスポネンシャルバックオフを追加してロジックを再接続することを検討してください。こうすると、サーバーに対して大量の新しい接続が同時に行われるのを防ぐことができます。

リシャーディング中に、次のことをお勧めします:

- コストの高いコマンドを避ける – KEYS や SMEMBERS コマンドのような、計算コストが高いオペレーションや入出力量の多いオペレーションを避けてください。これらのオペレーションでは、クラスターへの負荷が増えてクラスターのパフォーマンスに影響するため、これらを避けるアプローチをお勧めします。代わりに、SCAN コマンドおよび SSCAN コマンドを使用します。
- Lua のベストプラクティスに従う – 長時間実行する Lua スクリプトを避け、常に Lua スクリプトで使用されているキーを前に宣言します。Lua スクリプトがクロススロットコマンドを使用していないことを確認するために、この方法をお勧めします。Lua スクリプトで使用されるキーが同じスロットに属していることを確認します。

リシャーディング後は、以下の点に注意してください:

- ターゲットのシャードで十分なメモリが利用できない場合、スケールインが部分的に成功している可能性があります。そのような結果が生じた場合、必要に応じて使用可能なメモリを確認し、オペレーションを再試行してください。
- 大きなアイテムのスロットは移行されません。特に、シリアル化後に 256 MB を超えるアイテムを持つスロットは移行されません。
- FLUSHALL および FLUSHDB コマンドは、リシャーディング操作中の Lua スクリプト内ではサポートされません。

MemoryDB レプリケーションを理解する

MemoryDBは、最大500のシャードに分割されたデータでレプリケーションを実装しています。

クラスター内の各シャードには、単一の読み取り/書き込みプライマリノードと、最大 5 個の読み取り専用レプリカノードがあります。各プライマリノードは最大 100 MB/秒を維持できます。シャードの数が多くレプリカの数が少ないクラスターを作成できます。クラスターあたり最大 500 ノードです。このクラスター設定は、シャード 500 個およびレプリカ 0 個からシャード 100 個およびレプリカ 4 個 (許容されるレプリカの最大数) までです。

整合性

MemoryDBでは、プライマリノードは強い一貫性を持っています。成功した書き込み操作は、クライアントに返される前に、分散されたマルチ AZ トランザクションログに永続的に保存されます。プライマリでの読み取り操作では、それまでに成功したすべての書き込み操作の影響を反映した最新のデータが常に返されます。このような強固な一貫性は、プライマリのフェイルオーバー後も維持されます。

MemoryDB では、レプリカノードは結果整合性です。レプリカからの読み取り操作 (READONLY コマンドを使用) は、遅延メトリクスが CloudWatch に発行したため、直近に成功した書き込み操作の影響を常に反映するとは限りません。ただし、1 つのレプリカからの読み取りオペレーションはシーケンシャルに一貫性があります。書き込み操作が成功すると、プライマリで実行されたのと同じ順序で各レプリカで有効になります。

クラスター内のレプリケーション

シャード内の各リードレプリカは、シャードのプライマリノードからのデータのコピーを維持します。トランザクションログを使用した非同期レプリケーション機能は、リードレプリカとプライマリの同期を維持するのに使用されます。アプリケーションは、クラスター内のどのノードからでも読み取ることができます。アプリケーションは、そのプライマリノードにのみ書き込むことができます。リードレプリカは読み取りのスケーラビリティを高めます。MemoryDB はデータを耐久性のあるトランザクションログに保存するので、データが失われるリスクはありません。データは MemoryDB クラスター内のシャード間で分割されます。

アプリケーションは、MemoryDB クラスターのクラスターエンドポイントを使用してクラスターのノードに接続します。詳細については、「接続エンドポイントの検索」を参照してください。

MemoryDB クラスターはリージョナルで、1 つのリージョンのノードのみを含めることができます。耐障害性を向上させるために、そのリージョン内の複数のアベイラビリティーゾーンにプライマリとリードレプリカの両方をプロビジョニングできます。

マルチ AZ を提供するレプリケーションの使用は、すべての MemoryDB クラスターで強く推奨されます。詳細については、「マルチ AZ による MemoryDB のダウンタイムの最小化」を参照してください。

マルチ AZ による MemoryDB のダウンタイムの最小化

MemoryDB がプライマリノードを置き換える必要があるインスタンスが多数あります。これには、特定のタイプの計画されたメンテナンスや、プライマリノードまたはアベイラビリティーゾーンの予期できない障害などが含まれます。

ノード障害への対応は、どのノードに障害が発生したかによって異なります。ただし、いずれの場合も、MemoryDB はノードの交換やフェイルオーバー中にデータが失われないようにします。例えば、レプリカに障害が発生した場合、障害が発生したノードは交換され、データがトランザクションログから同期されます。プライマリノードに障害が発生すると、整合性のとれたレプリカへのフェイルオーバーがトリガーされ、フェイルオーバー中にデータが失われることはありません。これで、書き込みは新しいプライマリノードから処理されます。その後、古いプライマリノードが置き換えられ、トランザクションログから同期されます。

1 つのノードシャード (レプリカなし) でプライマリノードに障害が発生した場合、MemoryDB は、プライマリノードが交換されてトランザクションログと同期されるまで、書き込みを受け付けなくなります。

ノードの交換により、クラスターのダウンタイムが発生しますが、マルチ AZ が有効になっている場合、ダウンタイムは最小限に抑えられます。プライマリノードのロールは、リードレプリカの 1 つに自動的にフェイルオーバーされます。MemoryDB ではこれを透過的に処理するため、新しいプライマリノードを作成してプロビジョニングする必要はありません。このフェイルオーバーとレプリカの昇格により、昇格が完了したらすぐに新しいプライマリへの書き込みを再開できます。

メンテナンス更新またはサービス更新により計画されたノード置換が開始された場合、クラスターが着信した書き込みリクエストを処理している間に、計画されたノード置換が完了することに注意してください。

MemoryDB クラスターのマルチ AZ を使用すると、耐障害性が向上します。これは特に、クラスターのプライマリノードが到達できなくなった場合、または何らかの理由で障害が発生した場合に当てはまります。MemoryDB クラスターのマルチ AZ では、各シャードに複数のノードが必要で、自動的に有効になります。

トピック

- 障害シナリオとマルチ AZ のレスポンス
- 自動フェイルオーバーのテスト

障害シナリオとマルチ AZ のレスポンス

マルチ AZ がアクティブな場合、障害が発生したプライマリノードは使用可能なレプリカにフェイルオーバーします。レプリカは自動的にトランザクションログと同期され、プライマリノードになります。これは、新しいプライマリノードを作成して再プロビジョニングするよりもはるかに高速です。通常は数秒で、クラスターへの書き込みが再び可能になります。

マルチ AZ を有効にすると、MemoryDB はプライマリノードの状態を継続的にモニタリングします。プライマリノードが失敗すると、失敗のタイプに応じて次のいずれかのアクションが実行されます。

トピック

- プライマリノードのみが失敗した場合の障害シナリオ
- プライマリノードと一部のリードレプリカが失敗した場合の障害シナリオ
- クラスター全体が失敗した場合の障害シナリオ

プライマリノードのみが失敗した場合の障害シナリオ

プライマリノードのみが失敗した場合、レプリカは自動的にプライマリノードになります。次に、障害の発生したプライマリと同じアベイラビリティーゾーンに代替のリードレプリカが作成されてプロビジョニングされます。

プライマリノードのみが失敗した場合、MemoryDB のマルチ AZ は次の処理を行います。

1. 失敗したプライマリノードがオフラインになります。
2. 最新のレプリカが自動的にプライマリになります。

   書き込みは、フェイルオーバープロセスが完了すると通常は数秒で再開できます。
3. 代替のリードレプリカが起動され、プロビジョニングされます。

   代替のレプリカは、障害が発生したプライマリノードが属していたアベイラビリティーゾーンで起動され、ノードの分散が維持されます。
4. レプリカはトランザクションログと同期します。

クラスターのエンドポイントの検索については、以下のトピックを参照してください。

- MemoryDB クラスターのエンドポイントを検索する (MemoryDB API)

プライマリノードと一部のリードレプリカが失敗した場合の障害シナリオ

プライマリと 1 つ以上のレプリカに障害が発生すると、最新のレプリカがプライマリクラスターに昇格されます。新しいレプリカも作成され、障害が発生したノードと同じアベイラビリティーゾーンにプロビジョニングされます。

プライマリノードと一部のリードレプリカが失敗すると、MemoryDB Multi-AZは次の処理を行います。

1. 障害が発生したプライマリノードと故障したレプリカがオフラインになります。
2. 使用可能なレプリカがプライマリノードになります。

   フェイルオーバーが完了すると、書き込みは、通常は数秒で 再開できます。
3. 複数の置き換えレプリカを作成してプロビジョニングします。

   ノードのディストリビューションが維持されるように、障害が発生したノードのアベイラビリティーゾーンで置き換えレプリカが作成されます。
4. すべてのノードがトランザクションログと同期します。

クラスターのエンドポイントの検索については、以下のトピックを参照してください。

- (AWS CLI) MemoryDB クラスターのエンドポイントの検索
- MemoryDB クラスターのエンドポイントを検索する (MemoryDB API)

クラスター全体が失敗した場合の障害シナリオ

すべてに障害が発生した場合、すべてのノードは、元のノードと同じアベイラビリティーゾーンで再作成され、プロビジョニングされます。

このシナリオでは、データはトランザクションログに保持されているため、データが失われることはありません。

クラスター全体が失敗すると、MemoryDB のマルチ AZ は次の処理を行います。

1. 障害が発生したプライマリノードとレプリカがオフラインになります。

2. 代わりのプライマリノードが作成され、トランザクションログと同期してプロビジョニングされます。
3. 代替レプリカはトランザクションログと同期して作成され、プロビジョニングされます。

   ノードのディストリビューションが維持されるように、障害が発生したノードのアベイラビリティーゾーンで置き換えレプリカが作成されます。

クラスターのエンドポイントの検索については、以下のトピックを参照してください。

- (AWS CLI) MemoryDB クラスターのエンドポイントの検索
- MemoryDB クラスターのエンドポイントを検索する (MemoryDB API)

自動フェイルオーバーのテスト

MemoryDB コンソール、AWS CLI、および MemoryDB API を使用して自動フェイルオーバーをテストできます。

テストを行う場合、以下の点に注意してください。

- この操作は、24 時間に最大 5 回まで使用できます。
- 別のクラスターのシャードでこのオペレーションを呼び出す場合、同時に呼び出しを行うことができます。
- 場合によっては、同じ MemoryDBクラスター内の異なるシャードに対して、このオペレーションを複数回呼び出すことがあります。このような場合、後続の呼び出しを行う前に、最初のノードの置換が完了する必要があります。
- ノードの置換が完了しているかどうか調べるには、MemoryDB コンソール、AWS CLI、または MemoryDB API を使用してイベントを確認します。`FailoverShard`に関連する以下のイベントは、発生すると思われる順に一覧表示されます。

  1. クラスターメッセージ: `FailoverShard API called for shard <shard-id>`
  2. クラスターメッセージ: `Failover from primary node <primary-node-id> to replica node <node-id> completed`
  3. クラスターメッセージ: `Recovering nodes <node-id>`
  4. クラスターメッセージ: `Finished recovery for nodes <node-id>`

  詳細については次を参照してください:

  - MemoryDB API リファレンスの DescribeEvents
- この API は、MemoryDB でフェイルオーバーが発生した場合のアプリケーションの動作をテストするために設計されています。クラスターの問題に対処するためにフェイルオーバーを開始するための運用ツールとしては設計されていません。さらに、大規模な運用イベントなどの特定の条件下では、AWS がこの API をブロックする可能性があります。

トピック

- AWS マネジメントコンソール を使用した自動フェイルオーバーのテスト
- AWS CLI を使用した自動フェイルオーバーのテスト
- MemoryDB API を使用した自動フェイルオーバーのテスト

AWS マネジメントコンソール を使用した自動フェイルオーバーのテスト

コンソールで自動フェイルオーバーをテストするには、次の手順に従います。

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. テストしたいクラスターの左側にあるラジオボタンを選択します。このクラスターには、少なくとも 1 つのレプリカノードが必要です。
3. Details エリアで、このクラスターでマルチ AZ が有効になっていることを確認します。クラスターでマルチ AZ が有効になっていない場合は、別のクラスターを選択するか、このクラスターを変更してマルチ AZ を有効にします。詳細については、「MemoryDB クラスターの変更」を参照してください。
4. クラスターの名前を選択します。
5. [シャードとノード] ページで、フェイルオーバーをテストするシャード (API および CLI ではノードグループと呼ばれます) のシャード名を選択します。
6. ノード ページで [フェイルオーバープライマリ] を選択します。
7. Continue を選択してプライマリをフェイルオーバーするか、Cancel を選択してプライマリノードへのフェイルオーバーをキャンセルします。

   フェイルオーバープロセス中は、コンソールでノードのステータスが 使用可能 と継続して表示されます。フェイルオーバーテストの進捗状況を追跡するには、コンソールのナビゲーションペインから Events を選択します。Events タブで、フェイルオーバーの開始`FailoverShard API called`と完了`Recovery completed`を示すイベントを監視します。

AWS CLI を使用した自動フェイルオーバーのテスト

マルチ AZ が有効になっているクラスターで自動フェイルオーバーをテストするには、AWS CLI オペレーションfailover-shardの を使用できます。

パラメータ

- `--cluster-name` – 必須。テスト対象のクラスター。
- `--shard-name` – 必須。自動フェイルオーバーをテストするシャードの名前。24 時間以内に、最大 5 つのシャードをテストできます。

次の例では、AWS CLI を使用してMemoryDBクラスター my-cluster 内のシャード 0001 で failover-shard を呼び出します。

Linux、macOS、Unix の場合:

```
aws memorydb failover-shard \
   --cluster-name my-cluster \
   --shard-name 0001
```

Windows の場合:

```
aws memorydb failover-shard ^
   --cluster-name my-cluster ^
   --shard-name 0001
```

フェイルオーバーの進行状況を追跡するには、AWS CLI describe-events オペレーションを使用します。

以下のようなJSONレスポンスが返される :

```
{
    "Events": [
        {
            "SourceName": "my-cluster",
            "SourceType": "cluster",
            "Message": "Failover to replica node my-cluster-0001-002 completed",
            "Date": "2021-08-22T12:39:37.568000-07:00"
        },
        {
            "SourceName": "my-cluster",
            "SourceType": "cluster",
            "Message": "Starting failover for shard 0001",
            "Date": "2021-08-22T12:39:10.173000-07:00"
        }
    ]
}
```

詳細については次を参照してください:

- 「フェイルオーバーシャード」

- describe-events

MemoryDB API を使用した自動フェイルオーバーのテスト

次の例では、クラスター memorydb00 内のシャード 0003 で FailoverShard を呼び出します。

Example 自動フェイルオーバーのテスト

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=FailoverShard
    &ShardName=0003
    &ClusterName=memorydb00
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T192317Z
    &X-Amz-Credential=<credential>
```

フェイルオーバーの進行状況を追跡するには、MemoryDB DescribeEvents API オペレーションを使用します。

詳細については次を参照してください:

- 「フェイルオーバーシャード」
- DescribeEvents

レプリカの数の変更

AWS マネジメントコンソール、AWS CLI、または MemoryDB API を使用して、MemoryDB クラスターのリードレプリカ数を動的に増減できます。すべてのシャードのレプリカの数が同じである必要があります。

クラスターのレプリカの数を増やす

MemoryDB クラスター内のレプリカの数は、シャードごとに最大 5 個まで増やすことができます。AWS マネジメントコンソール、AWS CLI、または MemoryDB API を使用して行うことができます。

トピック

- の使用AWS マネジメントコンソール
- の使用AWS CLI
- MemoryDB API の使用

の使用AWS マネジメントコンソール

MemoryDB クラスター (コンソール) 内のレプリカの数を増やすには、「クラスターからのノードの追加/削除」を参照してください。

の使用AWS CLI

MemoryDB クラスターのレプリカ数を増やすには、以下のパラメータを指定して`update-cluster`コマンドを使用します：

- `--cluster-name` – 必須。レプリカの数を増やすクラスターを指定します。
- `--replica-configuration` – 必須。レプリカの数を設定できます。レプリカの数を増やすには、このオペレーションの終了後にこのシャードに必要なレプリカの数を `ReplicaCount` プロパティに設定します。

Example

次の例では、クラスター `my-cluster` 内のレプリカの数を 2 個に増やします。

Linux、macOS、Unix の場合:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --replica-configuration \
        ReplicaCount=2
```

Windows の場合:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --replica-configuration ^
        ReplicaCount=2
```

以下の JSON コードを返します。

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 1,
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

クラスターのステータスが更新中から利用可能に変わったら、更新されたクラスターの詳細を表示するには、次のコマンドを使用します。

Linux、macOS、Unix の場合:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Windows の場合:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

以下のようなJSONレスポンスが返される：

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-003",
```

```
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-22T12:59:31.844000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 3
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

CLI を使用してレプリカの数を増やす方法の詳細については、「AWS CLI コマンドリファレンス」の「クラスターの更新」を参照してください。

MemoryDB API の使用

MemoryDB シャードでレプリカの数を増やすには、以下のパラメータを設定して UpdateCluster アクションを使用します。

- ClusterName – 必須。レプリカの数を増やすクラスターを指定します。
- ReplicaConfiguration – 必須。レプリカの数を設定できます。レプリカの数を増やすには、このオペレーションの終了後にこのシャードに必要なレプリカの数を ReplicaCount プロパティに設定します。

Example

次の例では、クラスター sample-cluster 内のレプリカの数を 3 個に増やします。この例が終了すると、各シャードのレプリカは 3 個になります。この数は、単一のシャードを持つ MemoryDB クラスターでも、複数のシャードを持つ MemoryDB クラスターでも適用されます。

```
https://memory-db.us-east-1.amazonaws.com/
      ?Action=UpdateCluster
      &ReplicaConfiguration.ReplicaCount=3
      &ClusterName=sample-cluster
      &Version=2021-01-01
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20210802T192317Z
      &X-Amz-Credential=<credential>
```

API を使用したレプリカの数を増やす詳細については、「クラスターの更新」を参照してください。

クラスターのレプリカの数を減らす

MemoryDB のクラスター内のレプリカの数を減らせます。レプリカの数をゼロまで減らすことはできますが、プライマリノードに障害が発生した場合にレプリカにフェイルオーバーすることはできません。

AWS マネジメントコンソール、AWS CLI、または MemoryDB API を使用して、クラスター内のレプリカの数を減らせます。

トピック

- の使用AWS マネジメントコンソール
- の使用AWS CLI
- MemoryDB API の使用

の使用AWS マネジメントコンソール

MemoryDB クラスター (コンソール) 内のレプリカの数を減らすには、「クラスターからのノードの追加/削除」を参照してください。

の使用AWS CLI

MemoryDB クラスターでレプリカの数を減らすには、以下のパラメータを設定して `update-cluster` コマンドを使用します。

- `--cluster-name` – 必須。レプリカの数を減らすクラスターを指定します。
- `--replica-configuration` – 必須。

  `ReplicaCount` – レプリカノードの数を指定するには、このプロパティを設定します。

Example

次の例では、`--replica-configuration` を使用して、クラスター `my-cluster` 内のレプリカの数を、指定された値まで減らします。

Linux、macOS、Unix の場合:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --replica-configuration \
```

```
        ReplicaCount=1
```

Windows の場合:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --replica-configuration ^
        ReplicaCount=1 ^
```

以下のようなJSONレスポンスが返される：

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 1,
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

クラスターのステータスが更新中から利用可能に変わったら、更新されたクラスターの詳細を表示するには、次のコマンドを使用します：

Linux、macOS、Unix の場合:

```
aws memorydb describe-clusters \
```

```
    --cluster-name my-cluster
    --show-shard-details
```

Windows の場合:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

以下のようなJSONレスポンスが返される：

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
```

```
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

CLI を使用してレプリカの数を減らす方法の詳細については、「AWS CLI コマンドリファレンス」の「クラスターの更新」を参照してください。

MemoryDB API の使用

MemoryDB クラスターでレプリカの数を減らすには、以下のパラメータを設定して UpdateCluster アクションを使用します。

- ClusterName – 必須。レプリカの数を減らすクラスターを指定します。
- ReplicaConfiguration – 必須。レプリカの数を設定できます。

  ReplicaCount – レプリカノードの数を指定するには、このプロパティを設定します。

Example

次の例では、ReplicaCount を使用して、クラスター sample-cluster内のレプリカの数を 1 個に減らします。 この例が終了すると、各シャードのレプリカは 1 個になります。この数は、単一のシャードを持つ MemoryDB クラスターでも、複数のシャードを持つ MemoryDB クラスターでも適用されます。

```
https://memory-db.us-east-1.amazonaws.com/
      ?Action=UpdateCluster
      &ReplicaConfiguration.ReplicaCount=1
      &ClusterName=sample-cluster
      &Version=2021-01-01
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20210802T192317Z
      &X-Amz-Credential=<credential>
```

API を使用したレプリカの数を減らす詳細については、「クラスターの更新」を参照してください。

スナップショットおよび復元

MemoryDB クラスターは、データをマルチ AZ トランザクションログに自動的にバックアップしますが、クラスターのポイントインタイムスナップショットを定期的に作成するか、オンデマンドで作成するかを選択できます。これらのスナップショットを使用して、以前の時点でクラスターを再作成したり、新しいクラスターをシードしたりできます。スナップショットは、クラスター内の全データとクラスターのメタデータで構成されます。すべてのスナップショットは、耐久性のあるストレージを提供する Amazon Simple Storage Service (Amazon S3) に書き込まれます。いつでも、新しいMemoryDBクラスターを作成し、スナップショットからのデータをそのクラスターに挿入することでデータを復元できます。MemoryDB では、AWS マネジメントコンソール、AWS Command Line Interface (AWS CLI)、および MemoryDB API を使用してスナップショットを管理できます。

トピック

- スナップショットの制約:
- スナップショットの料金
- 自動スナップショットのスケジュール
- 手動スナップショットの作成
- 最終スナップショットの作成
- スナップショットの説明

- スナップショットのコピー
- のスナップショットをエクスポートする
- スナップショットからの復元
- 外部で作成されたスナップショットによる新しいクラスターのシード
- スナップショットのタグ付け
- スナップショットの削除

スナップショットの制約:

スナップショットを計画または作成するときは、以下の制約事項を考慮してください。

- MemoryDB クラスターでは、サポートされているすべてのノードタイプのスナップショットと復元が可能です。
- 連続する 24 時間で、クラスタードあたり 20 個までの手動スナップショットを作成できます。
- MemoryDB はクラスターレベルでのスナップショット作成のみをサポートします。MemoryDB はシャードレベルまたはノードレベルでのスナップショット作成をサポートしていません。
- スナップショットプロセス中は、クラスターで他の API または CLI オペレーションを実行できません。
- クラスターを削除し、最終スナップショットをリクエストした場合、MemoryDB は常にプライマリノードからスナップショットを取得します。これにより、クラスターが削除される前に、最新のデータがキャプチャされます。

スナップショットの料金

MemoryDB を使用して、アクティブな MemoryDB クラスターごとに 1 つのスナップショットを無料で保存できます。追加スナップショットのストレージ領域については、すべての AWS リージョンで 1 か月あたり $0.085/GB の料金が課金されます。 スナップショットの作成や、スナップショットからMemoryDBクラスターへのデータの復元には、データ転送料金はかかりません。

自動スナップショットのスケジュール

どの MemoryDB クラスターでも、自動スナップショットを有効にできます。自動スナップショットの有効になっている場合、MemoryDB はクラスターのスナップショットを毎日作成します。クラスタへの影響はなく、変更は即時に行われます。詳細については、「スナップショットからの復元」を参照してください。

自動スナップショットをスケジュールする場合は、次の設定を検討する必要があります:

- スナップショットウィンドウ — MemoryDB がスナップショットの作成を開始する各日の期間。スナップショットウィンドウの最短時間は 60 分です。スナップショットウィンドウは、いつでもお客様にとって都合の良い時間、つまり、特に使用率の高い時間と重ならないような時間に設定できます。

  指定しない場合、スナップショットウィンドウは MemoryDB によって自動的に割り当てられます。
- バックアップ保持期限– バックアップが Amazon S3 に保持される日数。例えば、保持期限を 5 に設定すると、今日作成されたスナップショットは 5 日間保持されます。保持期限が切れると、スナップショットは自動的に削除されます。

  最大スナップショット保持期限は 35 日です。スナップショット保持期限を 0 に設定すると、クラスターの自動スナップショットが無効になります。MemoryDB のデータは、自動スナップショットを無効にしても完全に保持されます。

MemoryDB コンソール、AWS CLI、または MemoryDB クラスターを作成するときに、自動スナップショットの有効または無効にすることができます。MemoryDB クラスターの作成時に自動スナップショットを有効にするには、[スナップショット] セクションの [自動バックアップを有効にする] ボックスをチェックします。詳細については、「MemoryDB クラスターの作成」。

手動スナップショットの作成

自動スナップショットに加えて、いつでも手動スナップショットを作成できます。指定された保持期間後に自動的に削除される自動スナップショットとは異なり、手動スナップショットには、保存期間後に自動的に削除される保持期間はありません。すべての手動スナップショットは手動で削除する必要があります。クラスターまたはノードを削除した場合でも、そのクラスターまたはノードの手動スナップショットはすべて保持されます。手動スナップショットを保持する必要がなくなった場合は、自分で明示的に削除する必要があります。

手動スナップショットはテストやアーカイブにも役立ちます。例えば、テスト目的で一連のベースラインデータを作成したとします。データの手動スナップショットを作成し、いつでも復元することができます。このデータを変更するアプリケーションをテストした後、新しいクラスターを作成し、ベースラインスナップショットから復元することによって、データをリセットできます。クラスターの準備ができたら、ベースラインデータに対してアプリケーションをテストし、必要に応じてこのプロセスを繰り返すことができます。

手動スナップショットの直接的な作成に加えて、以下のいずれかの方法で手動スナップショットを作成できます。

- 「スナップショットのコピー」ソーススナップショットが自動で作成されたか、手動で作成されたかは問題ではありません。
- 「最終スナップショットの作成」クラスターを削除する直前にスナップショットを作成します。

その他の重要なトピック

- スナップショットの制約:
- スナップショットの料金

AWS マネジメントコンソール、AWS CLI、またはMemoryDBAPI を使用して、ノードの手動スナップショットを作成できます。

手動スナップショットの作成 (コンソール)

クラスターのスナップショットを作成するには (コンソール)

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。

2. 左のナビゲーションペインで [クラスター] を選択します。

   MemoryDB クラスター画面が表示されます。

3. バックアップする MemoryDB クラスターの名前の左にあるラジオボタンを選択します。

4. [アクション]、[スナップショットの取得] の順に選択します。

5. [スナップショット] ウィンドウの [スナップショット名] ボックスに、スナップショットの名前を入力します。どのクラスターがバックアップされたか、スナップショットを作成した日付と時刻を示すような名前にすることをお勧めします。

   クラスターの命名に関する制約は次のとおりです。

   - 1～40 個の英数字またはハイフンを使用する必要があります。
   - 先頭は文字を使用する必要があります。
   - 連続する 2 つのハイフンを含めることはできません。
   - ハイフンで終わることはできません。

6. [暗号化] で、デフォルトの暗号化キーを使用するか、カスタマー管理のキーを使用するかを選択します。詳細については、「MemoryDBの転送時の暗号化 (TLS)」を参照してください。

7. [タグ] で、スナップショットの検索およびフィルタリングまたはコストの追跡を行うタグをオプションで追加します。また、スナップショットの検索やフィルタリングを行ったり、AWS コストを追跡したりできます。

8. [スナップショットの取得] を選択します。

   クラスターのステータスが snapshotting に変わります。ステータスが 使用可能に戻ると、バックアップの作成が完了です。

(AWS CLI) 手動スナップショットの作成

AWS CLI を使用してクラスターの手動スナップショットを作成するには、以下のパラメータを指定して `create-snapshot` AWS CLI オペレーションを使用します。

- `--cluster-name` – スナップショットのソースとして使用する MemoryDB クラスターの名前。このパラメータは、MemoryDB クラスターをバックアップするときに使用します。

  クラスターの命名に関する制約は次のとおりです。

  - 1～40 個の英数字またはハイフンを使用する必要があります。
  - 先頭は文字を使用する必要があります。

- 連続する 2 つのハイフンを含めることはできません。
- ハイフンで終わることはできません。

- `--snapshot-name` – 作成するスナップショットの名前。

関連トピック

詳細については、AWS CLI コマンドリファレンスの「`create-snapshot`」を参照してください。

手動スナップショットの作成 (MemoryDB API)

MemoryDB API を使用してクラスターの手動スナップショットを作成するには、以下のパラメータを指定して `CreateSnapshot` MemoryDB API オペレーションを使用します。

- `ClusterName` – スナップショットのソースとして使用する MemoryDB クラスターの名前。このパラメータは、MemoryDB クラスターをバックアップするときに使用します。

  クラスターの命名に関する制約は次のとおりです。
  - 1 ~ 40 個の英数字またはハイフンを使用する必要があります。
  - 先頭は文字を使用する必要があります。
  - 連続する 2 つのハイフンを含めることはできません。
  - ハイフンで終わることはできません。
- `SnapshotName` – 作成するスナップショットの名前。

関連トピック

詳細については、「スナップショットの作成」を参照してください。

最終スナップショットの作成

MemoryDB コンソール、AWS CLI、または MemoryDB の MoryDB を使用して、最終スナップショットを作成できます。

最終スナップショットの作成 (コンソール)

MemoryDB コンソールを使用して MemoryDB クラスターを削除すると、最終スナップショットを作成できます。

MemoryDB クラスターを削除するときに最終スナップショットを作成するには、削除ページで [はい] を選択し、ステップ 5: クラスターを削除する でスナップショットに名前を付けます。

最終スナップショットの作成 (AWS CLI)

AWS CLI を使用して MemoryDB クラスターを削除するときに、最終スナップショットを作成できます。

MemoryDB クラスターを削除する場合

クラスターの削除時に最終スナップショットを作成するには、以下のパラメータで、`delete-cluster` AWS CLI オペレーションを使用します。

- `--cluster-name` – 削除するクラスターの名前。
- `--final-snapshot-name` – 最終スナップショットの名前。

以下のコードは、最終スナップショット `bkup-20210515-final` をクラスター `myCluster` の削除時に作成します。

Linux、macOS、Unix の場合:

```
aws memorydb delete-cluster \
        --cluster-name myCluster \
        --final-snapshot-name bkup-20210515-final
```

Windows の場合:

```
aws memorydb delete-cluster ^
        --cluster-name myCluster ^
        --final-snapshot-name bkup-20210515-final
```

詳細については、AWS CLI CLI コマンドリファレンスの「delete-cluster」を参照してください。

最終スナップショットの作成 (MemoryDB API)

MemoryDB API を使用して、MemoryDB クラスターを削除するときに、最終スナップショットを作成できます。

MemoryDB クラスターを削除する場合

最終スナップショットを作成するには、以下のパラメータで、DeleteCluster MemoryDB API オペレーションを使用します。

- ClusterName – 削除するクラスターの名前。
- FinalSnapshotName - スナップショットの名前。

以下のMemoryDB オペレーションでは、クラスターmyCluster削除時にスナップショット bkup-20210515-final が作成されます。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DeleteCluster
    &ClusterName=myCluster
    &FinalSnapshotName=bkup-20210515-final
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210515T192317Z
    &X-Amz-Credential=<credential>
```

詳細については、「DeleteCluster」を参照してください。

スナップショットの説明

以下の手順では、スナップショットのリストを表示する方法を示しています。必要に応じて、特定のスナップショットの詳細を表示することもできます。

スナップショットの説明 (コンソール)

AWS マネジメントコンソール を使用してスナップショットを表示するには

1. コンソール にログインします
2. 左側のナビゲーションペインで、[ジョブ] を選択します。
3. 検索を使用して、[手動スナップショット]、[自動スナップショット]、または [すべてのスナップショット] をフィルタリングします。
4. 特定のスナップショットの詳細を表示するには、スナップショットの名前の左にあるラジオボタンを選択します。[アクション] を選択し、[詳細の表示] を選択します。
5. オプションで、[詳細表示] ページで、[コピー]、[復元]、[削除] などの追加のスナップショットアクションを実行できます。スナップショットにタグを追加することもできます

スナップショットの説明 (AWS CLI)

スナップショットのリストと必要に応じて特定のスナップショットの詳細を表示するには、`describe-snapshots` CLI オペレーションを使用します。

例

以下のオペレーションでは、パラメータ `--max-results` を使用して、アカウントに関連付けられた最大 20 個のスナップショットを一覧表示します。 パラメータ `--max-results` を省略すると、最大 50 個のスナップショットが一覧表示されます。

```
aws memorydb describe-snapshots --max-results 20
```

以下のオペレーションでは、パラメータ `--cluster-name` を使用して、クラスター `my-cluster` に関連付けられたスナップショットのみを一覧表示します。

```
aws memorydb describe-snapshots --cluster-name my-cluster
```

以下のオペレーションでは、パラメータ `--snapshot-name` を使用して、スナップショット `my-snapshot` の詳細を表示します。

```
aws memorydb describe-snapshots --snapshot-name my-snapshot
```

詳細については、「describe-snapshots」を参照してください。

スナップショットの説明 (MemoryDB API)

スナップショットのリストを表示するには、DescribeSnapshots オペレーションを使用します。

例

以下のオペレーションでは、パラメータ MaxResults を使用して、アカウントに関連付けられた最大 20 個のスナップショットを一覧表示します。 パラメータ MaxResults を省略すると、最大 50 個のスナップショットが一覧表示されます。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeSnapshots
    &MaxResults=20
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

以下のオペレーションでは、パラメータ ClusterName を使用して、クラスター MyCluster に関連付けられているすべてのスナップショットを一覧表示します。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeSnapshots
    &ClusterName=MyCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
```

```
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

以下のオペレーションでは、パラメータ SnapshotName を使用して、スナップショット MyBackup の詳細を表示します。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeSnapshots
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &SnapshotName=MyBackup
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

詳細については、「DescribeSnapshots」を参照してください。

スナップショットのコピー

自動で作成されたか手動で作成されたかにかかわらず、スナップショットのコピーを作成できます。スナップショットをコピーする場合、特にオーバーライドされない限り、ソースと同じ KMS 暗号化キーがターゲットに使用されます。スナップショットをエクスポートし、MemoryDB の外部からアクセスすることもできます。スナップショットのエクスポートについては、「のスナップショットをエクスポートする」を参照してください。

以下の手順では、スナップショットをコピーする方法を示しています。

スナップショットのコピー (コンソール)

スナップショットをコピーするには (コンソール)

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. スナップショットのリストを表示するには、左のナビゲーションペインから [スナップショット] を選択します。
3. スナップショットのリストで、コピーしたいスナップショットの名前の左側にあるラジオボタンを選択します。
4. [アクション] を選択して、[コピー] を選択します。
5. [スナップショットのコピー] ページで、次の操作を行います。

    a. [新しいスナップショット名] ボックスに新しいスナップショットの名前を入力します。

    b. オプションの ターゲットS3バケット ボックスは空白のままにします。このフィールドは、スナップショットのエクスポートにのみ使用され、S3 の特殊なアクセス権限を必要とします。スナップショットのエクスポートの詳細については、「のスナップショットをエクスポートする」を参照してください。

    c. AWS KMS デフォルトの暗号化キーを使用するか、カスタムキーを使用するかを選択します。詳細については、「MemoryDBの転送時の暗号化 (TLS)」を参照してください。

    d. オプションで、スナップショットのコピーにタグを追加することもできます。

    e. (コピー) を選択します。

(AWS CLI) スナップショットのコピー

スナップショットをコピーするには、 `copy-snapshot` オペレーションを使用します。

パラメータ

- --source-snapshot-name – コピーするスナップショットの名前。
- --target-snapshot-name – スナップショットのコピーの名前。
- --target-bucket – スナップショットのエクスポート用に予約されています。スナップショットのコピーを作成する場合は、このパラメータを使用しないでください。詳細については、「のスナップショットをエクスポートする」を参照してください。

以下の例では、自動スナップショットのコピーを作成します。

Linux、macOS、Unix の場合:

```
aws memorydb copy-snapshot \
    --source-snapshot-name automatic.my-primary-2021-03-27-03-15 \
    --target-snapshot-name my-snapshot-copy
```

Windows の場合:

```
aws memorydb copy-snapshot ^
    --source-snapshot-name automatic.my-primary-2021-03-27-03-15 ^
    --target-snapshot-name my-snapshot-copy
```

詳細については、「copy-snapshot」を参照してください。

スナップショットをコピーする (MemoryDB API)

スナップショットコピーするには、以下のパラメータを指定して、copy-snapshot オペレーションを使用します。

パラメータ

- SourceSnapshotName – コピーするスナップショットの名前。
- TargetSnapshotName – スナップショットのコピーの名前。
- TargetBucket – スナップショットのエクスポート用に予約されています。スナップショットのコピーを作成する場合は、このパラメータを使用しないでください。詳細については、「のスナップショットをエクスポートする」を参照してください。

以下の例では、自動スナップショットのコピーを作成します。

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=CopySnapshot
    &SourceSnapshotName=automatic.my-primary-2021-03-27-03-15
    &TargetSnapshotName=my-snapshot-copy
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

詳細については、「スナップショットのコピー」を参照してください。

のスナップショットをエクスポートする

MemoryDB は、MemoryDB スナップショットの Amazon Simple Storage Service (Amazon S3) バケットへのエクスポートをサポートしています。これにより、MemoryDB の外部からスナップショットにアクセスできます。エクスポートされた MemoryDB スナップショットは Valkey およびオープンソース Redis OSS に完全に準拠しており、適切なバージョンまたはツールで読み込むことができます。スナップショットは、MemoryDB コンソール、AWS CLI、または MemoryDB API を使用してエクスポートできます。

スナップショットのエクスポートは、他の AWS リージョンのクラスターを起動する必要がある場合に役立ちます。 1 つの AWS リージョンでデータをエクスポートし、.rdb ファイルを新しい AWS リージョンにコピーして、その .rdb ファイルを使うことにより、新しいクラスターの入力を待つのではなく、新しいクラスターをシードできます。新しいクラスターのシードについては、「外部で作成されたスナップショットによる新しいクラスターのシード」を参照してください。クラスターのデータをエクスポートする別の理由は、オフライン処理のために .rdb ファイルを使用するためです。

> **Important**
>
> - コピー先のMemoryDBスナップショットと Amazon S3 バケットは、同じ AWS リージョンにあることが必要です。
>
> Amazon S3 バケットにコピーされたスナップショットは暗号化されますが、スナップショットを保存する Amazon S3 バケットへのアクセス権を他の人に付与しないことを強くお勧めします。
> - データ階層化を使用するクラスターでは、Amazon S3 へのスナップショットのエクスポートはサポートされていません。詳細については、「データ階層化」を参照してください。

スナップショットをAmazon S3 バケットにエクスポートする前に、スナップショットと同じ AWS リージョンに Amazon S3 バケットが必要です。 バケットへのアクセスを MemoryDB に許可します。最初の 2 つのステップで、これを行う方法を示します。

> **Warning**
>
> 以下のシナリオでは、望まない方法でデータが公開される可能性があります。

- 他のユーザーがスナップショットのエクスポート先の Amazon S3 バケットにアクセスできる場合。

  スナップショットへのアクセスを制御するために、データへのアクセスを希望するユーザーにのみ Amazon S3 バケットへのアクセスを許可します。Amazon S3 バケットへのアクセスの管理については、Amazon S3 デベロッパーガイドの「アクセスの管理」を参照してください。
- 他のユーザーが CopySnapshot API を使用するためのアクセス許可を持っている。

  CopySnapshot API オペレーションの使用権限を持つユーザーまたはグループは、独自の Amazon S3 バケットを作成し、このバケットにスナップショットをコピーできます。スナップショットへのアクセスを制御するには、AWS Identity and Access Management (IAM) ポリシーを使用して、だれが CopySnapshot API を使用できるかを管理します。 IAM を使用して MemoryDB API オペレーションの使用を制御する方法については、MemoryDB ユーザーガイドの「MemoryDB でのアイデンティティとアクセス権の管理」を参照してください。

トピック

- ステップ 1: Amazon S3 バケットを作成する
- ステップ 2: Amazon S3 バケットへのアクセス権を MemoryDB に付与する
- ステップ 3: MemoryDB スナップショットをエクスポートする

ステップ 1: Amazon S3 バケットを作成する

以下の手順では、Amazon S3 コンソールを使用して、MemoryDBのスナップショットをエクスポートおよび保存する Amazon S3 バケットを作成します。

Amazon S3 バケットを作成するには

1. AWS マネジメントコンソール 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. バケットを作成する - バケット名と地域を選択する で、以下の操作を実行します。

   a. バケット名に Amazon S3 バケットの名前を入力します。

b. リージョン リストから、Amazon S3 バケットの AWS リージョンを選択します。この AWS リージョンは、エクスポートする MemoryDB スナップショットと同じ AWS リージョンである必要があります。

c. 作成を選択します。

Amazon 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「バケットの作成」を参照してください。

ステップ 2: Amazon S3 バケットへのアクセス権を MemoryDB に付与する

AWS2019 年 3 月 20 日より前に導入された リージョンはデフォルトで有効になっています。これらの AWS リージョンでは、すぐに使用を開始できます。2019 年 3 月 20 日以降に導入されたリージョンはデフォルトでは無効になっています。Managing AWS regions で説明されているように、これらのリージョンを使用する前に、それらを有効にするか、オプトインする必要があります。

MemoryDBにリージョンの S3 バケットへのアクセス権を AWS に付与する

AWS リージョンの Amazon S3 バケットで適切なアクセス許可を作成するには、以下の手順を実行します。

MemoryDBにS3 バケットへのアクセス権を付与するには

1. AWS マネジメントコンソール 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. スナップショットのコピー先とする Amazon S3 バケットの名前を選択します。これは、「ステップ 1: Amazon S3 バケットを作成する」で作成した S3 バケットとなります。
3. [許可] タブを選択し、[許可]で[バケットポリシー]を選択します。
4. ポリシーを更新して、オペレーションの実行に必要なアクセス許可を MemoryDB に付与します。

   - `Principal に [ "Service" : "region-full-name.memorydb-snapshot.amazonaws.com" ]` を追加します。
   - スナップショットを Amazon S3 バケットにエクスポートするために必要な、以下のアクセス許可を追加します。
     - `"s3:PutObject"`
     - `"s3:GetObject"`
     - `"s3:ListBucket"`

- "s3:GetBucketAcl"
- "s3:ListMultipartUploadParts"
- "s3:ListBucketMultipartUploads"

次に、更新されたポリシーの例を示します。

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Policy15397346",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "aws-region.memorydb-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

ステップ 3: MemoryDB スナップショットをエクスポートする

これで、S3 バケットを作成し、そのバケットにアクセスするためのアクセス許可を MemoryDB に付与しました。S3 オブジェクト所有権を ACL 対応に変更します (バケット所有者優先)。次は、MemoryDB コンソール、AWS CLI、または MemoryDB API を使用して、スナップショットをそ

のバケットにエクスポートできます。以下では、次の S3 固有の IAM アクセス許可を持っていることを前提としています。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "s3:GetBucketLocation",
   "s3:ListAllMyBuckets",
   "s3:PutObject",
   "s3:GetObject",
   "s3:DeleteObject",
   "s3:ListBucket"
  ],
  "Resource": "arn:aws:s3:::*"
 }]
}
```

MemoryDB スナップショットのエクスポート (コンソール)

以下のプロセスでは、MemoryDBコンソールを使用してスナップショットをAmazon S3バケットにエクスポートし、MemoryDBの外部からアクセスできるようにします。Amazon S3バケットは、MemoryDBスナップショットと同じAWSリージョンでなければなりません。

Amazon S3 バケットへの MemoryDB スナップショットのエクスポート

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. スナップショットのリストを表示するには、左のナビゲーションペインから [スナップショット] を選択します。
3. スナップショットのリストで、エクスポートしたいスナップショットの左側にあるラジオボタンを選択します。
4. コピー を選択します。
5. バックアップのコピーを作成しますかダイアログボックスで、以下の設定を指定します。

a. [新しいスナップショット名] ボックスに新しいスナップショットの名前を入力します。

この名前は 1 ~ 1,000 文字で、UTF-8 エンコードが可能である必要があります。

MemoryDB は、ここで入力した値に、シャード識別子と .rdb を追加します。例えば、「my-exported-snapshot」と入力した場合、MemoryDB が my-exported-snapshot-0001.rdb を作成します。

b. [ターゲットS3の場所] リストから、バックアップをコピーする Amazon S3 バケット「ステップ 1: Amazon S3 バケットを作成する」で作成したバケットの名前を選択します。

エクスポートプロセスが成功するためには、[ターゲットS3の場所] は、次のアクセス許可を持つバックアップの AWS リージョン内の Amazon S3 バケットである必要があります。

- オブジェクトアクセス –読み取りおよび 書き込み。
- アクセス許可 – 読み取り

詳細については、「ステップ 2: Amazon S3 バケットへのアクセス権を MemoryDB に付与する」を参照してください。

c. コピー を選択します。

> **Note**
>
> S3 バケットにMemoryDBがスナップショットをエクスポートするためのアクセス許可がない場合、以下のいずれかのエラーメッセージを受け取ります。「ステップ 2: Amazon S3 バケットへのアクセス権を MemoryDB に付与する」に戻り、示されたアクセス権限を追加して、スナップショットのエクスポートを再試行してください。
>
> - MemoryDB は S3 バケットで読み取り権限 % を付与されていません。
>
> 解決策: バケットで読み取りのアクセス権限を追加します。
> - MemoryDB は S3 バケットで % の WRITE 権限を付与されていません。
>
> 解決策: バケットで書き込みのアクセス権限を追加します。
> - MemoryDB は S3 バケットで % の READ_ACP 権限を付与されていません。
>
> 解決策: バケットで読み取りのアクセス権限を追加します。

別の AWS リージョンにスナップショットをコピーする場合は、Amazon S3 を使用してコピーします。 詳細については、Amazon Simple Storage Service ユーザーガイドの「オブジェクトのコピー」を参照してください。

MemoryDB スナップショットをエクスポートする (AWS CLI)

以下のパラメータを指定して copy-snapshot CLI オペレーションを使用することで、Amazon S3 バケットにスナップショットをエクスポートします。

パラメータ

- `--source-snapshot-name` – コピーするスナップショットの名前。
- `--target-snapshot-name` – スナップショットのコピーの名前。

  この名前は 1 ~ 1,000 文字で、UTF-8 エンコードが可能である必要があります。

  MemoryDB は、ここで入力した値に、シャード識別子と `.rdb` を追加します。例えば、「`my-exported-snapshot`」と入力した場合、MemoryDB が `my-exported-snapshot-0001.rdb` を作成します。
- `--target-bucket` – スナップショットをエクスポートするAmazon S3バケットの名前。スナップショットのコピーは、指定したバケットで作成されます。

  エクスポートプロセスが成功するためには、`--target-bucket` は、次のアクセス許可を持つスナップショットの AWS リージョン内の Amazon S3 バケットである必要があります。
  - オブジェクトアクセス – 読み取り および 書き込み。
  - アクセス許可 – 読み取り

  詳細については、「ステップ 2: Amazon S3 バケットへのアクセス権を MemoryDB に付与する」を参照してください。

以下のオペレーションは、amzn-s3-demo-bucket にスナップショットをコピーします。

Linux、macOS、Unix の場合:

```
aws memorydb copy-snapshot \
    --source-snapshot-name automatic.my-primary-2021-06-27-03-15 \
    --target-snapshot-name my-exported-snapshot \
    --target-bucket amzn-s3-demo-bucket
```

Windows の場合:

```
aws memorydb copy-snapshot ^
    --source-snapshot-name automatic.my-primary-2021-06-27-03-15 ^
    --target-snapshot-name my-exported-snapshot ^
    --target-bucket amzn-s3-demo-bucket
```

> **Note**
>
> S3 バケットにMemoryDBがスナップショットをエクスポートするためのアクセス許可がない場合、以下のいずれかのエラーメッセージを受け取ります。「ステップ 2: Amazon S3 バケットへのアクセス権を MemoryDB に付与する」に戻り、示されたアクセス権限を追加して、スナップショットのエクスポートを再試行してください。
>
> - MemoryDB は S3 バケットで読み取り権限 % を付与されていません。
>
>   解決策: バケットで読み取りのアクセス権限を追加します。
> - MemoryDB は S3 バケットで % の WRITE 権限を付与されていません。
>
>   解決策: バケットで書き込みのアクセス権限を追加します。
> - MemoryDB は S3 バケットで % の READ_ACP 権限を付与されていません。
>
>   解決策: バケットで読み取りのアクセス権限を追加します。

詳細については、AWS CLI コマンドリファレンスの「copy-snapshot」を参照してください。

別の AWS リージョンにスナップショットをコピーする場合は、Amazon S3 を使用してコピーします。 詳細については、Amazon Simple Storage Service ユーザーガイドの「オブジェクトのコピー」を参照してください。

MemoryDB スナップショットをエクスポートする (MemoryDB API)

以下のパラメータを指定して CopySnapshot API オペレーションを使用し、スナップショットを Amazon S3 バケットにエクスポートします。

パラメータ

- SourceSnapshotName – コピーするスナップショットの名前。
- TargetSnapshotName – スナップショットのコピーの名前。

  この名前は 1 ~ 1,000 文字で、UTF-8 エンコードが可能である必要があります。

MemoryDB は、ここで入力した値に、シャード識別子と .rdb を追加します。例えば、「my-exported-snapshot」と入力した場合、my-exported-snapshot-0001.rdb が返されます。

- TargetBucket – スナップショットをエクスポートするAmazon S3バケットの名前。スナップショットのコピーは、指定したバケットで作成されます。

  エクスポートプロセスが成功するためには、TargetBucket は、次のアクセス許可を持つスナップショットの AWS リージョン内の Amazon S3 バケットである必要があります。

  - オブジェクトアクセス – 読み取り および 書き込み。
  - アクセス許可 – 読み取り

  詳細については、「ステップ 2: Amazon S3 バケットへのアクセス権を MemoryDB に付与する」を参照してください。

以下の例では、Amazon S3 バケット amzn-s3-demo-bucket に自動スナップショットのコピーを作成します。

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=CopySnapshot
    &SourceSnapshotName=automatic.my-primary-2021-06-27-03-15
    &TargetBucket=&example-s3-bucket;
    &TargetSnapshotName=my-snapshot-copy
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

> **Note**
>
> S3 バケットにMemoryDBがスナップショットをエクスポートするためのアクセス許可がない場合、以下のいずれかのエラーメッセージを受け取ります。「ステップ 2: Amazon S3 バ

ケットへのアクセス権を MemoryDB に付与する」に戻り、示されたアクセス権限を追加して、スナップショットのエクスポートを再試行してください。

- MemoryDB は S3 バケットで読み取り権限 % を付与されていません。

  解決策: バケットで読み取りのアクセス権限を追加します。
- MemoryDB は S3 バケットで % の WRITE 権限を付与されていません。

  解決策: バケットで書き込みのアクセス権限を追加します。
- MemoryDB は S3 バケットで % の READ_ACP 権限を付与されていません。

  解決策: バケットで読み取りのアクセス権限を追加します。

詳細については、「スナップショットのコピー」を参照してください。

別の AWS リージョンにスナップショットをコピーする場合は、Amazon S3 のコピーを使用して、エクスポートされたスナップショットを別の AWS リージョンの Amazon S3 バケットにコピーします。 詳細については、Amazon Simple Storage Service ユーザーガイドの「オブジェクトのコピー」を参照してください。

スナップショットからの復元

MemoryDB または ElastiCache (Redis OSS) の .rdb スナップショットファイルから新しいクラスターにいつでもデータを復元できます。

MemoryDB の復元プロセスは以下をサポートします。

- ElastiCache (Redis OSS) から作成した 1 つ以上の .rdb スナップショットファイルから MemoryDB クラスターへの移行。

  復元を実行するには、.rdb ファイルは S3 に置かれている必要があります。
- スナップショットファイルの作成に使用されたクラスターのシャード数とは異なる、新しいクラスターのシャード数を指定します。
- 新しいクラスターに異なるノードタイプ (大きいまたは小さい) を指定します。より小さいノードタイプにスケールダウンする場合は、新しいノードタイプに、データとエンジンのオーバーヘッドに対する十分なメモリがあることを確認してください。
- 新しい MemoryDB クラスターのスロットを、スナップショットファイルの作成に使用したクラスターとは異なる方法で設定します。

> **Important**
>
> - MemoryDB クラスターは複数データベースをサポートしません。そのため、MemoryDB に復元すると、.rdb ファイルが複数のデータベースを参照している場合は復元に失敗します。
> - データ階層化を使用するクラスター (r6gd ノードタイプなど) から、データ階層化を使用しないクラスター (r6g ノードタイプなど) にスナップショットを復元することはできません。

スナップショットからクラスターを復元するときに変更を加えるかどうかは、選択によって決まります。これらの選択は、MemoryDB コンソールを使用して復元する場合は、[クラスターの復元] ダイアログボックスで行います。AWS CLI または MemoryDB API を使用して復元する場合は、パラメータ値を設定することで、これらの選択を行います。

復元オペレーション時に、MemoryDB は新しいクラスターを作成し、スナップショットファイルからのデータを使用して入力します。このプロセスが完了すると、クラスターはウォームアップ状態になり、リクエストを受け付けることができます。

> **Important**
>
> 先に進む前に、復元元のクラスターのスナップショットを作成したことを確認してください。詳細については、「手動スナップショットの作成」を参照してください。
> 外部で作成したスナップショットから復元する場合は、「外部で作成されたスナップショットによる新しいクラスターのシード」を参照してください。

次の手順は、MemoryDB コンソール、AWS CLI、またはMemoryDB API を使用して新しいクラスターへスナップショットを復元する手順を示しています。

スナップショットからの復元 (コンソール)

新規クラスターへスナップショットを復元するには (コンソール)

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. ナビゲーションペインで、[スナップショット] を選択します。
3. スナップショットのリストで、復元したいスナップショットの名前の横にあるボタンを選択します。
4. [アクション] を選択してから、[リストア] を選択します。
5. [クラスター設定] で以下を設定します。
   a. [クラスター名]– 必須。新しいクラスターの名前。
   b. [説明] - オプション 新規クラスターの説明。
6. [サブネットグループ] セクションを完了します。
   - [サブネットグループ] で、新しいサブネットグループを作成するか、このクラスターに適用する既存のサブネットグループを選択します。新しいものを作成する場合:
     - [名前] を入力する
     - [説明] を入力する
     - マルチ AZ を有効にした場合は、異なるアベイラビリティーゾーンに存在する少なくとも 2 つのサブネットをサブネットグループに含める必要があります。詳細については、「サブネットおよびサブネットグループ」を参照してください。

- 新しいサブネットグループを作成していて、既存の VPC がない場合は、VPC を作成するように求められます。詳細については、「Amazon VPC ユーザーガイド」の「Amazon VPC とは」を参照してください。

7. [クラスター設定] セクションに入力します。

   a. Valkey バージョンの互換性または Redis OSS バージョンの互換性については、デフォルトの `6.0` を受け入れます。

   b. [ポート] には、デフォルトのポート 6379 をそのまま使用するか、別のポートを使用する理由がある場合はポート番号を入力します。

   c. [パラメータグループ] には、`default.memorydb-redis6` パラメータグループをそのまま使用します。

      パラメータグループはクラスターのランタイムパラメータを制御します。パラメータグループの詳細については、「エンジン固有のパラメータ」を参照してください。

   d. [ノードタイプ] では、必要なノードタイプの値 (および関連するメモリサイズ) を選択します。

      r6gdノードタイプファミリーのメンバーを選択すると、クラスターのデータ階層化が自動的に有効になります。詳細については、「データ階層化」を参照してください。

   e. [シャード数] で、このクラスターに必要なシャード (パーティション/ノードグループ) の数を選択します。

      クラスター内のシャード数を動的に変更できます。詳細については、「MemoryDB クラスターのスケーリング」を参照してください。

   f. シャード当たりのレプリカ数 で、各シャードに必要なリードレプリカのノード数を選択します。

      次の制限があります。

      - マルチ AZ が有効になっている場合は、シャードごとに少なくとも 1 つのレプリカがあることを確認してください。
      - コンソールを使用してクラスターを作成する場合、シャードごとのレプリカ数は同じになります。

   g. 次へ を選択します。

   h. [詳細設定] セクションを完了します。

i. セキュリティグループ で、このクラスターに必要なセキュリティグループを選択します。セキュリティグループは、クラスターへのネットワークアクセスを制御するためのファイアウォールとして機能します。VPC のデフォルトのセキュリティグループを使用するか、新しいセキュリティグループを作成できます。

VPC セキュリティグループの詳細については、Amazon VPC ユーザーガイドの「VPC のセキュリティグループ」を参照してください。

ii. データは、次の方法で暗号化されます。

- 保管時の暗号化– ディスクに保存されているデータの暗号化を有効にします。詳細については、「保管時の暗号化」を参照してください。

> **Note**
>
> 別の暗号化キーを指定するには、カスタマー管理の AWS KMS キー を選択し、キーを選択することもできます。

- 転送時の暗号化 – 転送中のデータの暗号化を有効にします。これはデフォルトで有効になっています。詳細については、「転送中の暗号化」を参照してください。

暗号化なしを選択すると、「オープンアクセス」というオープンアクセスコントロールリストがデフォルトユーザーで作成されます。詳細については、「アクセスコントロールリスト (ACL) によるユーザー認証」を参照してください。

iii. [スナップショット] には、オプションでスナップショットの保存期間とスナップショットウィンドウを指定します。デフォルトでは、[自動スナップショットを有効にする] が選択されています。

iv. [メンテナンスウィンドウ] には、オプションでメンテナンスウィンドウを指定します。メンテナンスウィンドウは、MemoryDB がクラスターのシステムメンテナンスを毎週スケジュールする時間の長さ (通常は 1 時間単位) です。MemoryDB がメンテナンス期間の日時を選択することを許可するか ([指定なし])、自分で日時と期間を選択できます ([メンテナンス期間の指定])。リストから [メンテナンスウィンドウを指定] を選択した場合は、メンテナンスウィンドウの [開始日]、[開始時間]、および [期間] (時間単位) を選択します。すべての時刻は協定世界時 (UCT) です。

詳細については、「メンテナンスの管理」を参照してください。

v. [通知] で、既存の Amazon Simple Notification Service (Amazon SNS) トピックを選択するか、手動 ARN 入力を選択してトピックの Amazon リソースネーム (ARN) を入力します。Amazon SNS では、インターネットに接続されたスマートデバイスに通知をプッシュすることができます。デフォルトでは、通知は無効になります。詳細については、https://aws.amazon.com/sns/ を参照してください。

i. [タグ] には、オプションでタグを適用してクラスターを検索およびフィルタリングしたり、AWS コストを追跡したりできます。

j. すべてのエントリと選択を確認し、必要な修正を行います。準備が完了したら、クラスタの作成 を選択してクラスターを起動するか、キャンセル を選択してオペレーションをキャンセルします。

クラスターのステータスが 使用可能 になり次第、EC2 にアクセス権を付与して接続し、使用を開始できます。詳細については、「ステップ 3: クラスターへのアクセスの許可」および「ステップ 4: クラスターに接続する」を参照してください。

> **Important**
>
> クラスターが使用可能になった直後から、クラスターがアクティブである間は (実際に使用していない場合でも)、時間に応じた料金が発生します。このクラスターに対する課金を中止するには、クラスターを削除する必要があります。「ステップ 5: クラスターを削除する」を参照してください。

(AWS CLI) スナップショットからの復元

`create-cluster --snapshot-name` または `--snapshot-arns` を含めて、新しいクラスターをスナップショットからのデータでシードします。

詳細については次を参照してください:

- 「MemoryDB ユーザーガイド」の「クラスターの作成 (AWS CLI)」。
- AWS CLI コマンドリファレンスの create-cluster。

スナップショットからの復元 (MemoryDB API)

MemoryDB API オペレーション `CreateCluster` を使用して MemoryDB スナップショットを復元できます。

`CreateCluster` `SnapshotName` または `SnapshotArns` を含めて、新しいクラスターをスナップショットからのデータでシードします。

詳細については次を参照してください:

- 「MemoryDB ユーザーガイド」の「クラスターの作成 (MemoryDB API)」。
- 「MemoryDB API リファレンス」の「CreateCluster」。

外部で作成されたスナップショットによる新しいクラスターのシード

新しい MemoryDB クラスターを作成するときに、Valkey または Redis OSS .rdb スナップショットファイルのデータでシードできます。

MemoryDB スナップショットまたは ElastiCache (Redis OSS) スナップショットから新しい MemoryDB クラスターをシードする方法については、「スナップショットからの復元」を参照してください。

.rdb ファイルを使用して新しい MemoryDB クラスターをシードするときは、以下を実行できます。

- 新しいクラスターのシャード数を指定します。新しいクラスター内のシャードの数を指定します。この数は、スナップショットファイルの作成に使用されたクラスター内のシャードの数とは異なる場合があります。
- 新しいクラスターに、スナップショットを作成したクラスターで使用されているものより大きいか小さい、異なるノードタイプを指定します。より小さいノードタイプにスケールダウンする場合は、新しいノードタイプに、データとエンジンのオーバーヘッドに対する十分なメモリがあることを確認してください。

> **Important**
>
> - スナップショットデータがノードのリソースを超えていないことを確認する必要があります。
>
> スナップショットが大きすぎる場合、クラスターのステータスは `restore-failed` になります。 その場合は、クラスターを削除してやり直す必要があります。
>
> ノードタイプおよび仕様の完全なリストについては、「MemoryDB ノードタイプ固有のパラメータ」を参照してください。
> - .rdb ファイルは、Amazon S3 サーバー側の暗号化 (SSE-S3) でのみ暗号化できます。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」を参照してください。

ステップ 1: 外部クラスターで、スナップショットを作成する

MemoryDB クラスターにシードするスナップショットを作成するには

1. 既存の Valkey または Redis OSS インスタンスに接続します。

2. BGSAVE オペレーションまたは SAVE オペレーションを実行してスナップショットを作成します。 .rdb ファイルの場所を書き留めておきます。

   BGSAVE は非同期処理であり、処理中も他のクライアントをブロックしません。詳細については、「BGSAVE」を参照してください。

   SAVE が同期され、完了するまで他のプロセスがブロックされます。詳細については、「SAVE」を参照してください。

スナップショットの作成の詳細については、「永続性」を参照してください。

ステップ 2: Amazon S3 バケットとフォルダを作成する

スナップショットを作成したら、Amazon S3 バケット内のフォルダにアップロードする必要があります。これを行うには、最初にそのバケット内に Amazon S3 バケットとフォルダが必要です。既に適切なアクセス許可を持つ Amazon S3 バケットフォルダがある場合は、「ステップ 3: スナップショットを Amazon S3 にアップロードする」に進むことができます。

Amazon S3 バケットを作成するには

1. AWS マネジメントコンソール 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. Amazon S3 バケットを作成するには、Amazon Simple Storage Service ユーザーガイドの「バケットの作成」の手順に従います。

   Amazon S3 バケットの名前は DNS に準拠している必要があります。それ以外の場合、MemoryDB はバックアップファイルにアクセスできません。DNS コンプライアンスのルールは次のとおりです。

   - 名前は、3～63 文字以内にする必要があります。
   - 名前は、ピリオド (.) で区切られた 1 つのラベルまたは一連の複数のラベルとして指定します。
     - 先頭の文字には小文字の英文字または数字を使用します。
     - 終了の文字には小文字の英文字または数字を使用します。
     - 小文字の英文字、数字、およびダッシュのみを含めます。
   - 名前は IP アドレスの形式にすることはできません (例: 192.0.2.0)。

新しいMemoryDBクラスターと同じAWSリージョンにAmazon S3バケットを作成することを強くお勧めします。このアプローチにより、MemoryDB が Amazon S3 から .rdb ファイルを読み取る場合のデータ転送速度が最大限に速くなります。

> **Note**
>
> データを可能な限り安全に保つには、Amazon S3 バケットに対するアクセス許可をできるだけ制限します。同時に、バケットとその内容を使用して新しい MemoryDB クラスターをシードするためのアクセス許可を付与する必要があります。

Amazon S3 バケットにフォルダを追加するには

1. AWS マネジメントコンソール 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. .rdb ファイルのアップロード先となるバケットの名前を選択します。
3. Create folder (フォルダの作成) を選択します。
4. 新しいフォルダの名前を入力します。
5. 保存 を選択します。

   バケット名とフォルダ名の両方の名前を書き留めます。

ステップ 3: スナップショットを Amazon S3 にアップロードする

次に、「ステップ 1: 外部クラスターで、スナップショットを作成する」で作成した .rdb ファイルをアップロードします。アップロード先は、「ステップ 2: Amazon S3 バケットとフォルダを作成する」で作成した Amazon S3 バケットとフォルダです。このタスクの詳細については、オブジェクトのアップロードをご参照ください。ステップ 2 と 3 の間に、作成したフォルダ名を選択します。

.rdb ファイルを Amazon S3 フォルダにアップロードするには

1. AWS マネジメントコンソール 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. ステップ 2 で作成した Amazon S3 バケットの名前を選択します。
3. ステップ 2 で作成したフォルダの名前を選択します。

4. アップロードを選択します。
5. ファイルの追加を選択します。
6. アップロードする 1 つまたは複数のファイルを参照して見つけ、そのファイルを選択します。複数のファイルを選択するには、Ctrl キーを押しながら各ファイル名を選択します。
7. 開く をクリックします。
8. 正しいファイルが [アップロード] ダイアログボックスに表示されることを確認してから、[アップロード] を選択します。

.rdb ファイルへのパスを記録します。例えば、バケット名が `amzn-s3-demo-bucket` で、パスが `myFolder/redis.rdb` の場合は、「`amzn-s3-demo-bucket/myFolder/redis.rdb`」と入力します。新しいクラスターをこのスナップショットのデータでシードする際にこのパスが必要です。

詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットの命名規則」を参照してください。

ステップ 4: MemoryDBに.rdb ファイルへの読み込みアクセスを許可する

AWS2019 年 3 月 20 日より前に導入された リージョンはデフォルトで有効になっています。これらの AWS リージョンでは、すぐに使用を開始できます。2019 年 3 月 20 日以降に導入されたリージョンはデフォルトでは無効になっています。Managing AWS regions で説明されているように、これらのリージョンを使用する前に、それらを有効にするか、オプトインする必要があります。

MemoryDBに.rdb ファイルへの読み込みアクセスを許可する

スナップショットファイルへの読み込みアクセスを MemoryDB に許可するには

1. AWS マネジメントコンソール 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. .rdb ファイルを含む S3 バケットの名前を選択します。
3. .rdb ファイルを含むフォルダの名前を選択します。
4. .rdb スナップショットファイルの名前を選択します。選択したファイルの名前は、ページ先頭のタブの上に表示されます。
5. アクセス許可 タブを選択します。
6. 許可 で、バケットポリシーを選択し、編集を選択します。
7. ポリシーを更新して、オペレーションの実行に必要なアクセス許可を MemoryDB に付与します。

- Principal に ["Service" : "*region-full-name*.memorydb-snapshot.amazonaws.com"] を追加します。
- スナップショットを Amazon S3 バケットにエクスポートするために必要な、以下のアクセス許可を追加します。
  - "s3:GetObject"
  - "s3:ListBucket"
  - "s3:GetBucketAcl"

次に、更新されたポリシーの例を示します。

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Policy15397346",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "us-east-1.memorydb-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/snapshot1.rdb",
                "arn:aws:s3:::amzn-s3-demo-bucket/snapshot2.rdb"
            ]
        }
    ]
}
```

8. 保存を選択します。

ステップ 5: MemoryDB クラスターと .rdb ファイルデータを提携させる

これで MemoryDB クラスターを作成し、.rdb ファイルのデータと提携する準備が整いました。クラスターを作成するには、MemoryDB クラスターの作成 の指示に従います。

お客様が MemoryDB に使用するよう指示した方法は、Amazon S3 にアップロードしたバックアップがどこにあるのかを検索するものですが、これはクラスターの作成に使用する方法によって異なります。

MemoryDB クラスターと .rdb ファイルデータを提携させる

- MemoryDB コンソールの使用

  エンジンを選択したら、[詳細設定] セクションを展開し、[データをクラスターにインポートする] を見つけます。[シード RDB ファイルの S3 ロケーション] ボックスに、ファイルの Amazon S3 パスを入力します。複数の .rdb ファイルがある場合は、カンマ区切りのリストで各ファイルのパスを入力します。Amazon S3 パスは *amzn-s3-demo-bucket*/*myFolder*/*myBackupFilename*.rdb のようになります。
- の使用AWS CLI

  `create-cluster` または `create-cluster` オペレーションを使用する場合、パラメータ `--snapshot-arns` を使用して、各 .rdb ファイルの完全修飾 ARN を指定します 例えば、`arn:aws:s3:::`*amzn-s3-demo-bucket*/*myFolder*/*myBackupFilename*.rdb。ARN は、Amazon S3 に保存したスナップショットファイルに分解される必要があります。
- MemoryDB API の使用

  `CreateCluster` または `CreateCluster` MemoryDB API オペレーションを使用する場合、パラメータ `SnapshotArns` を使用して、各 .rdb ファイルの完全修飾 ARN を指定します。例えば、`arn:aws:s3:::`*amzn-s3-demo-bucket*/*myFolder*/*myBackupFilename*.rdb。ARN は、Amazon S3 に保存したスナップショットファイルに分解される必要があります。

クラスターの作成処理中、スナップショットのデータがクラスターに書き込まれます。MemoryDB イベントメッセージを表示して、進行状況をモニタリングできます。これを行うには、MemoryDB コンソールを参照し、[イベント] を選択します。AWS MemoryDB コマンドラインインターフェイスまたは MemoryDB API を使用して、イベントメッセージを取得することもできます。

スナップショットのタグ付け

タグ形式で各スナップショットに独自のメタデータを割り当てることができます。タグを使用すると、用途別、所有者別、環境別などのさまざまな方法でスナップショットを分類できます。これは同じタイプのリソースが多数ある場合に役立ちます。割り当てたタグに基づいて、特定のリソースをすばやく識別できます。詳細については、「タグを付けることができるリソース」を参照してください。

コスト配分タグは、費用を請求書タグ値別にグループ化することで、複数の AWS サービスのコストを追跡する方法です。コスト配分タグの詳細については、「コスト配分タグの使用」を参照してください。

MemoryDBコンソール、AWS CLI、またはMemoryDB API を使用して、スナップショットのコスト配分タグを追加、一覧表示、変更、削除、またはコピーできます。詳細については、「コスト配分タグによるコストのモニタリング」を参照してください。

スナップショットの削除

自動スナップショットは、保持期限を過ぎると自動的に削除されます。クラスターを削除すると、その自動スナップショットもすべて削除されます。

MemoryDB には、スナップショットが自動と手動のいずれで作成されたかにかかわらず、いつでもスナップショットを削除できる削除 API オペレーションが用意されています。手動スナップショットには保持期限がないため、手動削除は手動スナップショットを削除する唯一の方法です。

スナップショットを削除するには、MemoryDB コンソール、AWS CLI、または MemoryDB API を使用します。

スナップショットの削除 (コンソール)

以下の手順では、MemoryDB コンソールを使用してスナップショットを削除します。

スナップショットを削除するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 左のナビゲーションペインで [スナップショット]を選択します。

   スナップショット 画面に、スナップショットのリストが表示されます。
3. 削除するスナップショットの名前の左にあるラジオボタンを選択します。
4. アクションを選択してから、削除を選択します。
5. このスナップショットを削除する場合は、テキストボックスに `delete` と入力し、[削除] を選択します。ルールをキャンセルするには、[キャンセル] を選択します。ステータスが deleting に変わります。

スナップショットの削除 (AWS CLI)

スナップショットを削除するには、以下のパラメータを指定して delete-snapshot AWS CLI オペレーションを使用します。

- `--snapshot-name` - 削除するスナップショット。

次のコードは、スナップショット `myBackup` を削除します。

```
aws memorydb delete-snapshot --snapshot-name myBackup
```

詳細については、AWS CLI CLI コマンドリファレンスの「delete-snapshot」を参照してください。

スナップショットを削除する (MemoryDB API)

スナップショットを削除するには、以下のパラメータを指定して DeleteSnapshot API オペレーションを使用します。

- SnapshotName - 削除するスナップショット。

次のコードは、スナップショット myBackup を削除します。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DeleteSnapshot
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SnapshotName=myBackup
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

詳細については、「スナップショットの削除」 を参照してください。

Scaling (スケーリング)

アプリケーションが処理しなければならないデータの量は、一定ではありません。業務の拡大または または通常の変動が発生すると、需要は増加します。アプリケーションを自己管理する場合は、需要のピークに合わせて十分なハードウェアをプロビジョニングする必要がありますが、それにより費用が高くなります。MemoryDB を使用して、現在の需要を満たすためにスケールすることができます。支払いは使用した分のみとなります。

以下は、実行するスケーリングアクションに適したトピックの検索に役立ちます。

MemoryDB のスケーリング

アクション	MemoryDB	
スケールアウト	MemoryDB のオンラインリシャーディング	
ノードタイプの変更		

アクション	MemoryDB	
	ノードタイプの変更によるオンライン垂直スケーリング	
シャード数の変更	MemoryDB クラスターのスケーリング	

MemoryDB クラスターのスケーリング

クラスターの需要の変化に応じて MemoryDB のクラスター内のシャード数を変更することで、パフォーマンスを向上させたりコストを削減したりできます。そのために、スケーリングプロセス中でもクラスターがリクエストを処理し続けることができる、オンライン水平スケーリングの使用をお勧めします。

クラスターを再スケーリングするかどうかの判断条件には、次のようなものがあります。

- メモリプレッシャー:

  クラスター内のノードがメモリプレッシャーを受けている場合、より多くのリソースがより効率よくデータを保存してリクエストを処理するようにスケールアウトできます。

  ノードがメモリプレッシャーを受けているかどうかは、FreeableMemory、SwapUsage や BytesUsedForMemoryDB といったメトリクスをモニタリングすることで判断できます。
- CPU やネットワークボトルネック:

  レイテンシーやスループットがクラスターの問題となっている場合、問題解決のためにスケールアウトが必要な場合があります。

  CPUUtilization、NetworkBytesIn、NetworkBytesOut、CurrConnections、および NewConnections といったメトリクスをモニタリングすることで、レイテンシーとスループットのレベルを監視できます。
- クラスターのサイズが大きすぎます:

  現在のクラスターの需要からすると、スケールインを行ってもパフォーマンスに影響せず、コストも削減できます。

  FreeableMemory、SwapUsage、BytesUsedForMemoryDB、CPUUtilization、NetworkBytesIn、NetworkB および NewConnections といったメトリクスを使用して、安全にスケーリング可能かどうかを判断するためにクラスターの使用を監視できます。

パフォーマンスに対するスケーリングの影響

オフライン処理を使用してスケーリングすると、処理の大部分でクラスターがオフラインになるため、リクエストに対応できなくなります。オンラインメソッドを使用してスケーリングすると、スケーリングは大量の演算を行うオペレーションであるため、パフォーマンスがある程度低下します。

その場合でも、クラスターはスケーリングオペレーション全体を通してリクエストに対応しつづけます。エクスペリエンスがどれほど低下するかは、通常の CPU 使用率とデータによって異なります。

MemoryDB クラスターをスケーリングするには、2 つの方法として水平スケーリングと垂直スケーリングがあります。

- 水平スケーリングでは、シャードを追加または削除することで、クラスター内のシャードの数を変更できます。オンラインのリシャーディングプロセスでは、クラスターが着信リクエストの処理を継続しながら、スケールイン/スケールアウトが可能です。
- 垂直スケーリング - ノードタイプを変更することで、クラスターのサイズを変更します。オンラインの垂直スケーリングでは、クラスターが着信リクエストの処理を継続しながら、スケールアップ/ダウンが可能です。

スケールインまたはスケールダウンによってクラスターのサイズとメモリ容量を減らす場合は、新しい構成にデータとエンジンのオーバーヘッド用の十分なメモリがあることを確認します。

MemoryDB のオフラインリシャーディング

オフラインのシャード再構成の主な利点は、単にクラスターにシャードを追加または削除する以上のことが行えることです。オフラインでリシャーディングすると、クラスター内のシャード数の変更に加えて、次のことを実行できます。

- クラスターのノードタイプを変更します。
- 新しいエンジンバージョンに更新します。

> **Note**
>
> オフラインリシャーディングは、データ階層化が有効になっているクラスターではサポートされません。詳細については、「データ階層化」を参照してください。

オフラインのシャード再構成の主な欠点は、クラスターが復元処理の開始からオフラインになり、アプリケーションのエンドポイントを更新するまで継続することです。クラスターがオフラインになる時間の長さは、クラスターのデータ量によって変わります。

オフラインでシャード MemoryDB クラスターを再構成するには

1. 既存の MemoryDB クラスターの手動スナップショットを作成します。詳細については、「手動スナップショットの作成」を参照してください。
2. スナップショットから復元して新しいクラスターを作成します。詳細については、「スナップショットからの復元」を参照してください。
3. アプリケーション内のエンドポイントを、新しいクラスターのエンドポイントに更新します。詳細については、「接続エンドポイントの検索」を参照してください。

MemoryDB のオンラインリシャーディング

MemoryDB でオンラインリシャーディングを使用して、ダウンタイムなしで MemoryDB を動的にスケールできます。このアプローチでは、クラスターはスケーリングや再分散が処理中でもリクエストに対応し続けることができます。

以下の操作を行うことができます。

- スケールアウト — MemoryDB クラスターにシャードを追加して、読み取りと書き込みの容量を増やします。

  クラスターに 1 つ以上のシャードを追加する場合、新しい各シャードのノード数は既存の最小のシャードのノード数と同じになります。
- スケールイン読み込みおよび書き込みキャパシティーを減らして、MemoryDB クラスターからシャードを削除することでコストを削減します。

現在、MemoryDB のオンラインリシャーディングには、次の制限が適用されます。

- スロットまたはキースペース、および大きなアイテムには制限があります。

  シャード内のキーのいずれかに大きなアイテムが含まれる場合、スケールアウトの際にそのキーは新しいシャードに移行されません。この機能により、アンバランスなシャードになる可能性があります。

  シャード内のキーのいずれかに大きなアイテム (シリアル化後 256 MB より大きいアイテム) が含まれる場合、シャードはスケールイン時に削除されません。この機能により、一部のシャードは削除されない可能性があります。
- スケールアウトの際、新しいシャードのノード数は、既存のシャードのノード数と等しくなります。

詳細については、「ベストプラクティス: オンラインクラスターのサイズ変更」を参照してください。

AWS マネジメントコンソール、AWS CLI、MemoryDB API を使用して、MemoryDB クラスターを水平にスケールできます。

オンラインリシャーディングによるシャードの追加

AWS マネジメントコンソール、AWS CLI、または MemoryDB API を使用して、MemoryDB クラスターにシャードを追加できます。

シャードの追加 (コンソール)

AWS マネジメントコンソール を使用して、MemoryDB クラスターに 1 つ以上のシャードを追加できます。以下の手順では、このプロセスについて説明します。

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. クラスターの一覧から、シャードを追加するクラスターの名前を選択します。
3. [シャードとノード] タブで、[シャードの追加/削除] を選択します
4. [新しいシャード数] に、必要なシャードの数を入力します。
5. [確認] を選択して変更を保存するか、[キャンセル] を選択して破棄します。

シャードの追加 AWS CLI

以下のプロセスでは、AWS CLI を使用してシャードを追加し、MemoryDB クラスターでシャードの再構成を行う方法について説明します。

`update-cluster` を使って以下のパラメータを使用します。

パラメータ

- `--cluster-name` – 必須。シャードの再構成オペレーションを実行するクラスター (クラスター) を指定します。
- `--shard-configuration` – 必須。シャードの数を設定できます。
  - `ShardCount` – このプロパティを設定して、必要なシャードの数を指定します。

Example

次の例では、my-cluster クラスター内のシャードの数を 2 に変更しています。

Linux、macOS、Unix の場合:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --shard-configuration \
        ShardCount=2
```

Windows の場合:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --shard-configuration ^
        ShardCount=2
```

以下の JSON コードを返します。

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 2,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
```

```
    }
}
```

クラスターのステータスが更新中から利用可能に変わったら、更新されたクラスターの詳細を表示するには、次のコマンドを使用します。

Linux、macOS、Unix の場合:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Windows の場合:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

以下のようなJSONレスポンスが返される：

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 2,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-8191",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
```

```
                        }
                    },
                    {
                        "Name": "my-cluster-0001-002",
                        "Status": "available",
                        "AvailabilityZone": "us-east-1b",
                        "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                        "Endpoint": {
                            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                            "Port": 6379
                        }
                    }
                ],
                "NumberOfNodes": 2
            },
            {
                "Name": "0002",
                "Status": "available",
                "Slots": "8192-16383",
                "Nodes": [
                    {
                        "Name": "my-cluster-0002-001",
                        "Status": "available",
                        "AvailabilityZone": "us-east-1b",
                        "CreateTime": "2021-08-22T14:26:18.693000-07:00",
                        "Endpoint": {
                            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                            "Port": 6379
                        }
                    },
                    {
                        "Name": "my-cluster-0002-002",
                        "Status": "available",
                        "AvailabilityZone": "us-east-1a",
                        "CreateTime": "2021-08-22T14:26:18.765000-07:00",
                        "Endpoint": {
                            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                            "Port": 6379
                        }
                    }
                ],
```

```
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

詳細については、「AWS CLI コマンドリファレンス」の「update-cluster」を参照してください。

シャードの追加 (MemoryDB API)

UpdateCluster オペレーションを使うことで、MemoryDB API を使用して MemoryDB クラスターのシャードをオンラインで再構成できます。

UpdateCluster を使って以下のパラメータを使用します。

パラメータ

- ClusterName – 必須。シャードの再構成オペレーションを実行するクラスターを指定します。
- ShardConfiguration – 必須。シャードの数を設定できます。
  - ShardCount – このプロパティを設定して、必要なシャードの数を指定します。

詳細については、「クラスターの更新」を参照してください。

オンラインリシャーディングによるシャードの削除

AWS マネジメントコンソール、AWS CLI、または MemoryDB API を使用して、MemoryDB クラスターからシャードを削除できます。

シャードの削除 (コンソール)

以下のプロセスでは、AWS マネジメントコンソール を使用してシャードを削除し、MemoryDB クラスターでシャードの再構成を行う方法について説明します。

> **Important**
>
> クラスターからシャードを削除する前に、MemoryDB はすべてのデータが残りのシャードに収まるようにします。データが収まる場合、シャードは要求に応じてクラスターから削除されます。データが残りのシャードに収まらない場合、プロセスは終了し、クラスターはリクエスト前と同じシャード設定のままになります。

AWS マネジメントコンソール を使用して、MemoryDB クラスターから 1 つ以上のシャードを削除できます。クラスター内のシャードをすべて削除することはできません。代わりに、クラスターを削除する必要があります。詳細については、「ステップ 5: クラスターを削除する」を参照してください。次の手順では、1 つ以上のシャードを削除する手順を説明します。

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. クラスターの一覧から、シャードを削除するクラスターの名前を選択します。
3. [シャードとノード] タブで、[シャードの追加/削除] を選択します
4. [新しいシャード数] に、必要なシャードの数を入力します (最低 1 つ)。
5. [確認] を選択して変更を保存するか、[キャンセル] を選択して破棄します。

シャードの削除 AWS CLI

以下のプロセスでは、AWS CLI を使用してシャードを削除し、MemoryDB クラスターでシャードの再構成を行う方法について説明します。

> **Important**
>
> クラスターからシャードを削除する前に、MemoryDB はすべてのデータが残りのシャードに収まるようにします。データが収まる場合、指定されたシャードはリクエストに応じてクラスターから削除され、キースペースは残りのシャードにマッピングされます。データが残りのシャードに収まらない場合、プロセスは終了し、クラスターはリクエスト前と同じシャード設定のままになります。

AWS CLI を使用して、MemoryDB クラスターから 1 つ以上のシャードを削除できます。クラスター内のシャードをすべて削除することはできません。代わりに、クラスターを削除する必要があります。詳細については、「ステップ 5: クラスターを削除する」を参照してください。

`update-cluster` を使って以下のパラメータを使用します。

パラメータ

- `--cluster-name` – 必須。シャードの再構成オペレーションを実行するクラスター (クラスター) を指定します。
- `--shard-configuration` – 必須。`ShardCount` プロパティを使用してシャードの数を設定できます。

  `ShardCount` – このプロパティを設定して、必要なシャードの数を指定します。

Example

次の例では、`my-cluster` クラスター内のシャードの数を 2 に変更しています。

Linux、macOS、Unix の場合:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --shard-configuration \
        ShardCount=2
```

Windows の場合:

```
aws memorydb update-cluster ^
```

```
    --cluster-name my-cluster ^
    --shard-configuration ^
        ShardCount=2
```

以下の JSON コードを返します。

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 2,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

クラスターのステータスが更新中から利用可能に変わったら、更新されたクラスターの詳細を表示するには、次のコマンドを使用します：

Linux、macOS、Unix の場合:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Windows の場合:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

以下のようなJSONレスポンスが返される：

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 2,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-8191",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
```

```
                },
                {
                    "Name": "0002",
                    "Status": "available",
                    "Slots": "8192-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0002-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-22T14:26:18.693000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0002-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-22T14:26:18.765000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
```

```
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

詳細については、「AWS CLI コマンドリファレンス」の「update-cluster」を参照してください。

シャードの削除 (MemoryDB API)

UpdateCluster オペレーションを使うことで、MemoryDB API を使用して MemoryDB クラスターのシャードをオンラインで再構成できます。

以下のプロセスでは、MemoryDB API を使用してシャードを削除し、 MemoryDBクラスターでシャードの再構成を行う方法について説明します。

> **Important**
>
> クラスターからシャードを削除する前に、MemoryDB はすべてのデータが残りのシャードに収まるようにします。データが収まる場合、指定されたシャードはリクエストに応じてクラスターから削除され、キースペースは残りのシャードにマッピングされます。データが残りのシャードに収まらない場合、プロセスは終了し、クラスターはリクエスト前と同じシャード設定のままになります。

MemoryDB API を使用して、MemoryDB クラスターから 1 つ以上のシャードを削除できます。クラスター内のシャードをすべて削除することはできません。代わりに、クラスターを削除する必要があります。詳細については、「ステップ 5: クラスターを削除する」を参照してください。

UpdateCluster を使って以下のパラメータを使用します。

パラメータ

- ClusterName – 必須。シャードの再構成オペレーションを実行するクラスター (クラスター) を指定します。

- `ShardConfiguration` – 必須。`ShardCount` プロパティを使用してシャードの数を設定できます。

  `ShardCount` – このプロパティを設定して、必要なシャードの数を指定します。

ノードタイプの変更によるオンライン垂直スケーリング

MemoryDB でオンラインの垂直スケーリングを使用すると、最小限のダウンタイムでクラスターを動的にスケーリングできます。これにより、クラスターはスケーリング中であってもリクエストを処理できます。

> **Note**
>
> データ階層化を使用するクラスター (r6gd ノードタイプを使用するクラスターなど) と、データ階層化を使用しないクラスター (r6g ノードタイプを使用するクラスターなど) 間のスケーリングはサポートされていません。詳細については、「データ階層化」を参照してください。

以下の操作を行うことができます。

- スケールアップ – より大きいノードタイプを使用するように Redis クラスターのノードタイプを調整して、読み取りおよび書き込み容量を増やします。

  MemoryDB は、オンラインのままリクエストを処理しながら、クラスターのサイズを動的に変更します。

- [スケールダウン] – より小さいノードを使用するようにノードタイプを調整することで、読み取りおよび書き込み容量を減らします。スケールダウンする 同様に、MemoryDB は、オンラインのままリクエストを処理しながら、クラスターのサイズを動的に変更します。この場合、ノードのサイズを小さくすることでコストを削減します。

> **Note**
>
> スケールアップおよびスケールダウンプロセスは、新しく選択されたノードタイプでクラスターを作成し、新しいノードを以前のノードと同期させることに依存します。スケールアップ/ダウンフローをスムーズにするには、以下の手順を実行します。

- 垂直スケーリングプロセスは、完全にオンラインのままになるように設計されており、古いノードと新しいノードとの間でデータを同期させることに依存します。データトラフィックが最小になると予想される時間帯にスケールアップ/ダウンを開始することをお勧めします。
- 可能であれば、ステージング環境でのスケーリング中にアプリケーションの動作をテストします。

オンラインスケールアップ

トピック

- MemoryDB クラスターのスケールアップ (コンソール)
- MemoryDB クラスターのスケールアップ (AWS CLI)
- MemoryDB クラスターのスケールアップ (MemoryDB API)

MemoryDB クラスターのスケールアップ (コンソール)

以下の手順では、AWS マネジメントコンソール を使用して MemoryDBクラスターをスケールアップする方法について説明しています。このプロセス中、MemoryDB クラスターは最小限のダウンタイムでリクエストを処理し続けます。

クラスターをスケールアップするには (コンソール)

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. クラスターのリストから、クラスターを選択します。
3. アクション を選択してから、変更 を選択します。
4. [クラスターの変更] ダイアログで以下を行います。
    - Node type リストから、スケーリングするノードタイプを選択します。スケールアップするには、既存のノードよりも大きいノードタイプを選択します。
5. 変更の保存 をクリックします。

    クラスターのステータスが修正中に変わります。ステータスが 使用可能 に変わると、変更は完了し、新しいクラスターの使用を開始できます。

MemoryDB クラスターのスケールアップ (AWS CLI)

以下の手順では、AWS CLI を使用して MemoryDBクラスターをスケールアップする方法について説明しています。このプロセス中、MemoryDB クラスターは最小限のダウンタイムでリクエストを処理し続けます。

MemoryDB クラスターをスケールアップするには (AWS CLI)

1. 以下のパラメータを指定して AWS CLI `list-allowed-node-type-updates` コマンドを実行することで、スケールアップできるノードタイプを調べます。

   Linux、macOS、Unix の場合:

   ```
   aws memorydb list-allowed-node-type-updates \
        --cluster-name my-cluster-name
   ```

   Windows の場合:

   ```
   aws memorydb list-allowed-node-type-updates ^
        --cluster-name my-cluster-name
   ```

   上のコマンドによる出力は以下のような JSON 形式になります。

   ```
   {
        "ScaleUpNodeTypes": [
            "db.r6g.2xlarge",
            "db.r6g.large"
        ],
        "ScaleDownNodeTypes": [
            "db.r6g.large"
        ],
   }
   ```

   詳細については、「AWS CLI リファレンス」の「リストで許可されるノードタイプの更新」を参照してください。

2. AWS CLI `update-cluster` と次のパラメータクラスターを変更して、新しいより大きなノードタイプにスケールアップします。

   - `--cluster-name` – スケールアップするクラスターの名前。

- `--node-type` – クラスターのスケーリング後の新しいノードタイプ。この値は、ステップ 1 で `list-allowed-node-type-updates` コマンドによって返されるノードタイプのいずれかであることが必要です。

Linux、macOS、Unix の場合:

```
aws memorydb update-cluster  \
 --cluster-name my-cluster \
 --node-type db.r6g.2xlarge
```

Windows の場合:

```
aws memorydb update-cluster ^
     --cluster-name my-cluster ^
     --node-type db.r6g.2xlarge ^
```

詳細については、「update-cluster」を参照してください。

MemoryDB クラスターのスケールアップ (MemoryDB API)

以下のプロセスでは、MemoryDB API を使用して、キャッシュクラスターをその現在のノードタイプから新しいより大きいノードタイプにスケーリングします。このプロセスでは、MemoryDB は DNS エントリを更新し、新しいノードを参照します。クラスターがオンラインのままで受信リクエストを処理している間に、自動フェイルオーバー対応クラスターをスケーリングできます。

より大きいノードタイプへのスケールアップにかかる時間はノードタイプと現在のクラスターのデータ量によって異なります。

MemoryDB クラスターをスケールアップするには (MemoryDB API)

1. 以下のパラメータを指定して MemoryDB API `ListAllowedNodeTypeUpdates` アクションを使用することで、スケールアップできるノードタイプを調べます。

   - `ClusterName` - クラスターの名前。すべてのクラスターではなく特定のクラスターの定義を表示するには、このパラメータを使用します。

```
https://memory-db.us-east-1.amazonaws.com/
```

```
    ?Action=ListAllowedNodeTypeUpdates
    &ClusterName=MyCluster
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &X-Amz-Credential=<credential>
```

詳細については、「MemoryDB API リファレンス」の「ListAllowedNodeTypeUpdates」を参照してください。

2. 以下のパラメータを指定して UpdateClusterMemoryDB API アクションを使用することで、現在のクラスターを新しいノードタイプにスケールアップします。

   - ClusterName - クラスターの名前。
   - NodeType – このクラスターの新しいより大きいノードタイプ。この値は、手順 1 で ListAllowedNodeTypeUpdates アクションによって返されるインスタンスタイプのいずれかであることが必要です。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &NodeType=db.r6g.2xlarge
    &ClusterName=myCluster
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

詳細については、「クラスターの更新」を参照してください。

オンラインスケールダウン

トピック

- MemoryDB クラスターのスケールダウン (コンソール)

- MemoryDB クラスターのスケールダウン (AWS CLI)
- MemoryDB クラスターのスケールダウン (MemoryDB API)

MemoryDB クラスターのスケールダウン (コンソール)

以下の手順では、AWS マネジメントコンソール を使用して MemoryDB クラスターをスケールダウンする方法について説明しています。このプロセス中、MemoryDB クラスターは最小限のダウンタイムでリクエストを処理し続けます。

MemoryDB クラスターをスケールダウンするには (コンソール)

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. クラスターのリストから、希望するクラスターを選択します。
3. アクション を選択してから、変更 を選択します。
4. [クラスターの変更] ダイアログで以下を行います。
   - Node type リストから、スケーリングするノードタイプを選択します。スケールダウンするには、既存のノードより小さいノードタイプを選択します。すべてのノードタイプがスケールダウンできるわけではないことに注意してください。
5. 変更の保存をクリックします。

   クラスターのステータスが修正中に変わります。ステータスが 使用可能 に変わると、変更は完了し、新しいクラスターの使用を開始できます。

MemoryDB クラスターのスケールダウン (AWS CLI)

以下の手順では、AWS CLI を使用して MemoryDB クラスターをスケールダウンする方法について説明しています。このプロセス中、MemoryDB クラスターは最小限のダウンタイムでリクエストを処理し続けます。

MemoryDB クラスターをスケールダウンするには (AWS CLI)

1. 以下のパラメータを指定して AWS CLI `list-allowed-node-type-updates` コマンドを実行することで、スケールダウンできるノードタイプを調べます。

   Linux、macOS、Unix の場合:

```
aws memorydb list-allowed-node-type-updates \
     --cluster-name my-cluster-name
```

Windows の場合:

```
aws memorydb list-allowed-node-type-updates ^
     --cluster-name my-cluster-name
```

上のコマンドによる出力は以下のような JSON 形式になります。

```
{
     "ScaleUpNodeTypes": [
         "db.r6g.2xlarge",
         "db.r6g.large"
     ],
     "ScaleDownNodeTypes": [
         "db.r6g.large"
     ],
}
```

詳細については、「リスト許可ノードタイプ更新」を参照してください。

2. 以下のパラメータを指定して update-cluster コマンドを使用することで、クラスターを変更して、新しいより小さなノードタイプにスケールダウンします。

   - --cluster-name – スケールダウンするクラスターの名前。
   - --node-type – クラスターのスケーリング後の新しいノードタイプ。この値は、ステップ 1 で list-allowed-node-type-updates コマンドによって返されるノードタイプのいずれかであることが必要です。

   Linux、macOS、Unix の場合:

   ```
   aws memorydb update-cluster  \
        --cluster-name my-cluster \
        --node-type db.r6g.large
   ```

   Windows の場合:

```
aws memorydb update-cluster ^
     --cluster-name my-cluster ^
     --node-type db.r6g.large
```

詳細については、「update-cluster」を参照してください。

MemoryDB クラスターのスケールダウン (MemoryDB API)

以下のプロセスでは、MemoryDB API を使用して、クラスターを現在のノードタイプから新しいより小さなノードタイプにスケーリングします。このプロセス中、MemoryDB クラスターは最小限のダウンタイムでリクエストを処理し続けます。

より小さいノードタイプへのスケールダウンにかかる時間はノードタイプと現在のクラスターのデータ量によって異なります。

スケールダウン (MemoryDB API)

1. 以下のパラメータを指定した ListAllowedNodeTypeUpdates APIを使用して、どのノードタイプにスケールダウンできるか決定します。

   - `ClusterName` - クラスターの名前。すべてのクラスターではなく特定のクラスターの定義を表示するには、このパラメータを使用します。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=ListAllowedNodeTypeUpdates
    &ClusterName=MyCluster
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &X-Amz-Credential=<credential>
```

2. 以下のパラメータを指定した UpdateCluster APIを使用して、現在のクラスターを新しいノードタイプにスケールダウンします。

   - `ClusterName` - クラスターの名前。

- NodeType – このクラスターの新しいより小さいノードタイプ。この値は、手順 1 で ListAllowedNodeTypeUpdates アクションによって返されるインスタンスタイプのいずれかであることが必要です。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &NodeType=db.r6g.2xlarge
    &ClusterName=myReplGroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

パラメータグループを使用したエンジンパラメータの設定

MemoryDB はパラメータを使用して、ノードとクラスターの実行時のプロパティを制御します。通常、新しいエンジンバージョンには新しい機能をサポートするための追加のパラメータが含まれます。パラメータのテーブルについては、「エンジン固有のパラメータ」を参照してください。

もちろん、maxmemory などのパラメータ値はエンジンやノードのタイプによって決まります。ノードタイプ別のパラメータ値のテーブルについては、「MemoryDB ノードタイプ固有のパラメータ」を参照してください。

トピック

- パラメータの管理
- パラメータグループの階層
- パラメータグループを作成する
- パラメータグループを名前別に一覧表示する
- パラメータグループの値を一覧表示する
- パラメータグループを変更する

- パラメータグループを削除する
- エンジン固有のパラメータ

パラメータの管理

パラメータの管理を容易にするために、パラメータは名前付きのパラメータグループに分類されます。パラメータグループは、起動時にエンジンソフトウェアに渡されるパラメータの特定の値の組み合わせを表しています。これらの値により、各ノードのエンジンプロセスが実行時にどのように動作するかが決まります。特定のパラメータグループのパラメータ値は、クラスターが属するグループに関係なく、そのグループに関連付けられているすべてのノードに適用されます。

クラスターのパフォーマンスをファインチューニングするには、パラメータ値を変更するか、またはクラスターのパラメータグループを変更できます。

- デフォルトのパラメータグループの変更や削除はできません。カスタムパラメータ値が必要な場合は、カスタムパラメータグループを作成する必要があります。
- パラメータグループファミリーとユーザーが割り当てているクラスターには、互換性が必要です。例えば、クラスターが Redis OSS バージョン 6 を実行している場合は、memorydb_redis6 ファミリーのパラメータグループ、デフォルト値またはカスタム値のみを使用できます。
- クラスターのパラメータを変更すると、その変更は即座にクラスターに適用されます。これは、クラスターのパラメータグループ自体を変更するか、クラスターのパラメータグループ内のパラメータ値を変更するかに関係なく当てはまります。

パラメータグループの階層

MemoryDB パラメータグループ階層用

グローバルデフォルト

リージョン内のすべての MemoryDB のお客様向け最上位ルートパラメータグループ。

グローバルデフォルトのパラメータグループ:

- MemoryDB 向けに確保されており、お客様が使用することはできません。

お客様デフォルト

グローバルデフォルトのパラメータグループのコピーは、お客様が使用するために作成されています。

お客様デフォルトのパラメータグループ:

- MemoryDB が作成、所有します。
- このパラメータグループでサポートされているエンジンのバージョンを実行しているすべてのクラスターのパラメータグループとして使用できます。
- お客様が編集することはできません。

お客様所有

お客様デフォルトのパラメータグループのコピー。お客様所有のパラメータグループは、お客様がパラメータグループを作成する度に作成されます。

お客様所有のパラメータグループ:

- お客様が作成、所有します。
- お客様の互換性のあるいずれのクラスターにも割り当てることができます。
- カスタムパラメータグループを作成するようにお客様が変更できます。

  すべてのパラメータ値を変更できるわけではありません。詳細については、「エンジン固有のパラメータ」を参照してください。

パラメータグループを作成する

デフォルト値から変更するパラメータの値が 1 つ以上ある場合、新しいパラメータグループを作成する必要があります。パラメータグループは、MemoryDB コンソール、AWS CLI、または MemoryDB API を使って作成できます。

パラメータグループを作成する (コンソール)

次の手順では、MemoryDB コンソールを使用してパラメータグループを編集する方法を示します。

MemoryDB コンソールを使用してパラメータグループを作成するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 使用可能なすべてのパラメータグループのリストを表示するには、左側のナビゲーションペインで Parameter Groups を選択します。
3. パラメータグループを作成するには、[パラメータグループの作成] を選択します。

   [DB パラメータグループの作成] ページが表示されます。
4. Name ボックスで、このパラメータグループの一意の名前を入力します。

   クラスターを作成、またはクラスターのパラメータグループを変更するときは、パラメータグループを名前で選択します。したがって、わかりやすくパラメータグループのファミリーを特定するのに役立つ名前をお勧めします。

   パラメータグループの命名に関する制約は次のとおりです。

   - 先頭を ASCII 文字にする必要があります。
   - ASCII 文字、数字、ハイフンのみを含めることができます。
   - 1 ~ 255 文字にする必要があります。
   - 連続する 2 つのハイフンを含めることはできません。
   - ハイフンで終わることはできません。
5. Description ボックスに、パラメータグループの説明を入力します。
6. [エンジンバージョン互換性] ボックスで、このパラメータグループが対応するエンジンバージョンを選択します。
7. [タグ] には、パラメータグループを検索してフィルタリングしたり、AWS コストを追跡したりするためのタグをオプションで追加できます。

8. パラメータグループを作成するには、作成 を選択します。

   パラメータグループを作成しないでプロセスを終了するには、Cancel を選択します。

9. パラメータグループが作成されると、ファミリーのデフォルト値が設定されます。デフォルト値を変更するには、パラメータグループを変更する必要があります。詳細については、「パラメータグループを変更する」を参照してください。

パラメータグループを作成する AWS

AWS CLI を使用してパラメータグループを作成するには、以下のパラメータを指定して `create-parameter-group` コマンドを使用します。

- `--parameter-group-name` — パラメータグループの名前。

  パラメータグループの命名に関する制約は次のとおりです。

  - 先頭を ASCII 文字にする必要があります。
  - ASCII 文字、数字、ハイフンのみを含めることができます。
  - 1 ~ 255 文字にする必要があります。
  - 連続する 2 つのハイフンを含めることはできません。
  - ハイフンで終わることはできません。
- `--family` — パラメータグループのエンジンとバージョンファミリー。
- `--description` — パラメータグループについてユーザーが入力する説明。

Example

次の例では、memorydb_redis6 ファミリーをテンプレートとして使用して、myRedis6x という名前のパラメータグループを作成します。

Linux、macOS、Unix の場合:

```
aws memorydb create-parameter-group \
    --parameter-group-name myRedis6x  \
    --family memorydb_redis6 \
    --description "My first parameter group"
```

Windows の場合:

```
aws memorydb create-parameter-group ^
    --parameter-group-name myRedis6x  ^
    --family memorydb_redis6 ^
    --description "My first parameter group"
```

このコマンドの出力は次のようになります。

```
{
    "ParameterGroup": {
        "Name": "myRedis6x",
        "Family": "memorydb_redis6",
        "Description": "My first parameter group",
        "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x"
    }
}
```

パラメータグループが作成されると、ファミリーのデフォルト値が設定されます。デフォルト値を変更するには、パラメータグループを変更する必要があります。詳細については、「パラメータグループを変更する」を参照してください。

詳細については、 を参照してください。create-parameter-group

パラメータグループを作成する (MemoryDB API)

MemoryDB API を使用してパラメータグループを作成するには、以下のパラメータを指定して CreateParameterGroup アクションを使用します。

- ParameterGroupName — パラメータグループの名前。

  パラメータグループの命名に関する制約は次のとおりです。

  - 先頭を ASCII 文字にする必要があります。
  - ASCII 文字、数字、ハイフンのみを含めることができます。
  - 1 ~ 255 文字にする必要があります。
  - 連続する 2 つのハイフンを含めることはできません。
  - ハイフンで終わることはできません。
- Family — パラメータグループのエンジンとバージョンファミリー。例えば、memorydb_redis6。
- Description — パラメータグループについてユーザーが入力する説明。

Example

次の例では、memorydb_redis6 ファミリーをテンプレートとして使用して、myRedis6x という名前のパラメータグループを作成します。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=CreateParameterGroup
   &Family=memorydb_redis6
   &ParameterGroupName=myRedis6x
   &Description=My%20first%20parameter%20group
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

このアクションからの応答は、次のようになります。

```
<CreateParameterGroupResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
  <CreateParameterGroupResult>
    <ParameterGroup>
      <Name>myRedis6x</Name>
      <Family>memorydb_redis6</Family>
      <Description>My first parameter group</Description>
      <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
    </ParameterGroup>
  </CreateParameterGroupResult>
  <ResponseMetadata>
    <RequestId>d8465952-af48-11e0-8d36-859edca6f4b8</RequestId>
  </ResponseMetadata>
</CreateParameterGroupResponse>
```

パラメータグループが作成されると、ファミリーのデフォルト値が設定されます。デフォルト値を変更するには、パラメータグループを変更する必要があります。詳細については、「パラメータグループを変更する」を参照してください。

詳細については、 を参照してください。CreateParameterGroup

パラメータグループを名前別に一覧表示する

MemoryDB コンソール、AWS CLI、または MemoryDB API を使用して、パラメータグループを一覧できます。

パラメータグループを名前別に一覧表示する (コンソール)

次の手順は、MemoryDB コンソールを使用してパラメータグループのリストを表示する方法を示します。

MemoryDB コンソールを使用してパラメータグループを一覧するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 使用可能なすべてのパラメータグループのリストを表示するには、左側のナビゲーションペインで Parameter Groups を選択します。

パラメータグループを名前別に一覧表示する (AWS CLI)

AWS CLI を使用してパラメータグループのリストを生成するには、`describe-parameter-groups` コマンドを使用します。パラメータグループの名前を指定した場合は、そのパラメータグループのみが一覧表示されます。パラメータグループの名前を指定しない場合は、最大で `--max-results` のパラメータグループが一覧表示されます。いずれの場合も、パラメータグループの名前、ファミリー、および説明が表示されます。

Example

次のサンプルコードは、パラメータグループ myRedis6x のリストです。

Linux、macOS、Unix の場合:

```
aws memorydb describe-parameter-groups \
    --parameter-group-name myRedis6x
```

Windows の場合:

```
aws memorydb describe-parameter-groups ^
    --parameter-group-name myRedis6x
```

このコマンドの出力は、名前の一覧、ファミリー、パラメータグループの説明となります。

```
{
    "ParameterGroups": [
     {
         "Name": "myRedis6x",
         "Family": "memorydb_redis6",
         "Description": "My first parameter group",
                "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
myredis6x"
     }
    ]
}
```

Example

次のサンプルコードは、Valkey または Redis OSS エンジンバージョン 5.0.6 以降で実行されているパラメータグループのパラメータグループ myRedis6x を示しています。

Linux、macOS、Unix の場合:

```
aws memorydb describe-parameter-groups \
    --parameter-group-name myRedis6x
```

Windows の場合:

```
aws memorydb describe-parameter-groups ^
    --parameter-group-name myRedis6x
```

このコマンドの出力は、名前の一覧、ファミリー、パラメータグループの説明となります。

```
{
    "ParameterGroups": [
     {
         "Name": "myRedis6x",
         "Family": "memorydb_redis6",
         "Description": "My first parameter group",
                "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
myredis6x"
     }
    ]
```

```
}
```

Example

次のサンプルコードリストには、最大で 20 個のパラメータグループが一覧されています。

```
aws memorydb describe-parameter-groups --max-results 20
```

このコマンドの JSON 出力は、名前の一覧、ファミリー、各パラメータグループの説明となります。

```
{
    "ParameterGroups": [
        {
            "ParameterGroupName": "default.memorydb-redis6",
            "Family": "memorydb_redis6",
            "Description": "Default parameter group for memorydb_redis6",
            "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
default.memorydb-redis6"
        },
        ...
    ]
}
```

詳細については、 を参照してください。describe-parameter-groups

パラメータグループを名前別に一覧表示する (MemoryDB API)

MemoryDB API を使用してパラメータグループのリストを生成するには、DescribeParameterGroups アクションを使用します。パラメータグループの名前を指定した場合は、そのパラメータグループのみが一覧表示されます。パラメータグループの名前を指定しない場合は、最大で MaxResults のパラメータグループが一覧表示されます。いずれの場合も、パラメータグループの名前、ファミリー、および説明が表示されます。

Example

次のサンプルコードリストには、最大で 20 個のパラメータグループが一覧されています。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeParameterGroups
   &MaxResults=20
```

```
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

このアクションからの応答は次のようになります。memorydb_redis6 の場合はパラメータグループごとに名前、ファミリー、説明が一覧表示されます。

```
<DescribeParameterGroupsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
  <DescribeParameterGroupsResult>
    <ParameterGroups>
      <ParameterGroup>
        <Name>myRedis6x</Name>
        <Family>memorydb_redis6</Family>
        <Description>My custom Redis OSS 6 parameter group</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
      </ParameterGroup>
       <ParameterGroup>
        <Name>default.memorydb-redis6</Name>
        <Family>memorydb_redis6</Family>
        <Description>Default parameter group for memorydb_redis6</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/default.memorydb-
redis6</ARN>
      </ParameterGroup>
    </ParameterGroups>
  </DescribeParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeParameterGroupsResponse>
```

Example

次のサンプルコードは、パラメータグループ myRedis6x のリストです。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeParameterGroups
   &ParameterGroupName=myRedis6x
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
```

```
    &Version=2021-01-01
    &X-Amz-Credential=<credential>
```

このアクションからの応答は、名前、ファミリー、説明となります。

```
<DescribeParameterGroupsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
  <DescribeParameterGroupsResult>
    <ParameterGroups>
      <ParameterGroup>
        <Name>myRedis6x</Name>
        <Family>memorydb_redis6</Family>
        <Description>My custom Redis OSS 6 parameter group</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
      </ParameterGroup>
    </ParameterGroups>
  </DescribeParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeParameterGroupsResponse>
```

詳細については、 を参照してください。DescribeParameterGroups

パラメータグループの値を一覧表示する

MemoryDB コンソール、AWS CLI、または MemoryDB API を使用して、パラメータグループのパラメータとその値を一覧表示できます。

パラメータグループの値を一覧表示する (コンソール)

次の手順は、MemoryDB コンソールを使用してパラメータグループのパラメータと値を一覧する方法を示しています。

MemoryDB コンソールを使用してパラメータグループのパラメータとその値を表示するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 使用可能なすべてのパラメータグループのリストを表示するには、左側のナビゲーションペインで Parameter Groups を選択します。
3. パラメータグループ名 (横にあるボックスではなく) を選択して、パラメータと値を一覧表示するパラメータグループを選択します。

   パラメータと値は画面の下部に表示されます。パラメータの数によっては、スクロールして関心のあるパラメータを検索するに必要がある場合もあります。

パラメータグループの値を一覧表示する (AWS CLI)

AWS CLI を使用してパラメータグループのパラメータとその値の一覧を表示するには、`describe-parameters` コマンドを使用します。

Example

次のサンプルコードは、パラメータグループ myRedis6x のすべてのパラメータと値リストを一覧します。

Linux、macOS、Unix の場合:

```
aws memorydb describe-parameters \
    --parameter-group-name myRedis6x
```

Windows の場合:

```
aws memorydb describe-parameters ^
    --parameter-group-name myRedis6x
```

詳細については、 を参照してください。describe-parameters

パラメータグループの値を一覧表示する (MemoryDB API)

MemoryDB API を使用してパラメータグループのパラメータとその値の一覧を表示するには、DescribeParameters アクションを使用します。

詳細については、 を参照してください。DescribeParameters

パラメータグループを変更する

> **Important**
>
> デフォルトのパラメータグループを変更することはできません。

パラメータグループでいくつかのパラメータを変更できます。これらのパラメータ値は、パラメータグループに関連付けられるクラスターに適用されます。パラメータ値の変更がパラメータグループに適用される場合の詳細については、「エンジン固有のパラメータ」を参照してください。

パラメータグループを変更する (コンソール)

次の手順では、MemoryDB コンソールでパラメータ値を変更する方法を説明します。同じ手順を使用して、すべてのパラメータを変更します。

MemoryDB コンソールを使用してパラメータ値を変更するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 使用可能なすべてのパラメータグループのリストを表示するには、左側のナビゲーションペインで Parameter Groups を選択します。
3. パラメータグループ名の左側にあるラジオボタンを選択して、変更するパラメータグループを選択します。

   [アクション] を選択し、[詳細の表示] を選択します。または、パラメータグループ名を選択して詳細ページに移動することもできます。

4. パラメータの横にある [編集] を選択します。編集可能なパラメータはすべて編集可能になります。変更したいパラメータを見つけるには、ページを移動しなければならない場合があります。または、名前、値、または検索ボックスに入力してパラメータを検索することもできます。
5. 必要なパラメータ修正を行います。
6. 変更を保存するには、変更の保存を選択します。
7. 複数のページにわたってパラメータ値を変更した場合は、[変更をプレビュー] を選択してすべての変更を確認できます。変更を確定するには、[保存] を選択します。さらに変更を加えるには、[戻る] を選択します。
8. [パラメータの詳細] ページには、デフォルト値にリセットするオプションもあります。デフォルト値にリセットするには、[デフォルトにリセット] を選択します。チェックボックスはすべてのパラメータの左側に表示されます。リセットしたいものを選択し、[リセットに進む] を選択して確定します。

   [確認] を選択し、ダイアログボックスでリセット操作を確定します。
9. パラメータの詳細ページでは、各ページに表示するパラメータの数を設定できます。右側の歯車を使って変更を行います。詳細ページで必要な列を有効/無効にすることもできます。これらの変更は、コンソールのセッション中ずっと続きます。

   変更したパラメータの名前を検索するには、「エンジン固有のパラメータ」を参照してください。

パラメータグループを変更する (AWS CLI)

AWS CLI を使用してパラメータの値を変更するには、`update-parameter-group` コマンドを使用します。

変更するパラメータの名前と許容値を検索するには、「エンジン固有のパラメータ」を参照してください。

詳細については、「update-parameter-group」を参照してください。

パラメータグループを変更する (MemoryDB API)

MemoryDB API を使用してパラメータグループのパラメータ値を変更するには、`UpdateParameterGroup` アクションを使用します。

変更するパラメータの名前と許容値を検索するには、「エンジン固有のパラメータ」を参照してください。

詳細については、 を参照してください。UpdateParameterGroup

パラメータグループを削除する

MemoryDB コンソール、AWS CLI、または MemoryDB API を使用してカスタムのパラメータグループを削除できます。

パラメータグループがクラスターに関連付けられている場合は、パラメータグループを削除できません。デフォルトのパラメータグループも削除できません。

パラメータグループを削除する (コンソール)

次の手順では、MemoryDB コンソールを使用してパラメータグループを削除する方法を示します。

MemoryDB コンソールを使用してパラメータグループを削除するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 使用可能なすべてのパラメータグループのリストを表示するには、左側のナビゲーションペインで Parameter Groups を選択します。
3. パラメータグループ名の左側にあるラジオボタンを選択して、削除するパラメータグループを選択します。

   アクション を選択してから、削除 を選択します。
4. パラメータグループの削除の確認画面が表示されます。
5. パラメータグループを削除するには、確認テキストボックスに [削除] と入力します。

   パラメータグループを保持するには、キャンセルを選択します。

パラメータグループを削除する (AWS CLI)

AWS CLI を使用してパラメータグループを削除するには、`delete-parameter-group` コマンドを使用します。削除するパラメータグループで、`--parameter-group-name` で指定されたパラメータグループは、それに関連付けられるクラスターを持つことはできません。また、デフォルトのパラメータグループも持つことはできません。

次のサンプルコードは、myRedis6x パラメータグループを削除します。

Example

Linux、macOS、Unix の場合:

```
aws memorydb delete-parameter-group \
    --parameter-group-name myRedis6x
```

Windows の場合:

```
aws memorydb delete-parameter-group ^
    --parameter-group-name myRedis6x
```

詳細については、「delete-parameter-group」を参照してください。

パラメータグループを削除する (MemoryDB API)

MemoryDB API を使用したパラメータグループを削除するには、DeleteParameterGroup アクションを使用します。削除するパラメータグループで、ParameterGroupName で指定されたパラメータグループは、それに関連付けられるクラスターを持つことはできません。また、デフォルトのパラメータグループも持つことはできません。

Example

次のサンプルコードは、myRedis6x パラメータグループを削除します。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DeleteParameterGroup
   &ParameterGroupName=myRedis6x
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

詳細については、 を参照してください。DeleteParameterGroup

エンジン固有のパラメータ

Valkey または Redis OSS クラスターにパラメータグループを指定しない場合、エンジンのバージョンに適したデフォルトのパラメータグループが使用されます。デフォルトのパラメータグループのパラメータの値を変更することはできません。しかし、カスタムパラメータグループを作成し、いつでもクラスターに割り当てることはできます。ただし、条件付きで変更可能なパラメータの値が両方のパラメータグループで同じである場合に限ります。詳細については、「パラメータグループを作成する」を参照してください。

トピック

- Valkey 7 および Redis OSS 7 パラメータの変更
- Redis OSS 6 パラメータ
- MemoryDB ノードタイプ固有のパラメータ

Valkey 7 および Redis OSS 7 パラメータの変更

> **Note**
>
> MemoryDB は、新しいイミュータブルなパラメータグループ `default.memorydb-valkey7.search` を含むベクトル検索のプレビューリリースを導入しました。このパラメータグループは、MemoryDB コンソールで、および create-cluster CLI コマンドを使用して新しいベクトル検索対応クラスターを作成する際に使用できます。プレビューリリースは、次の AWS リージョンで利用できます: 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)、アジアパシフィック(東京)、および欧州 (アイルランド)。

パラメータグループファミリー: memorydb_valkey7

Valkey 7 および Redis OSS 7 で追加されたパラメータは次のとおりです。

名前	詳細	説明
`latency-tracking`	許可される値: `yes`、`no` デフォルト: `no` タイプ: 文字列	[yes] に設定すると、コマンドごとのレイテンシーが追跡され、`INFO` レイテンシー統計コマンドを使用してパーセンタイル分布をエクスポートし、`LATENCY` コマンドを使用して累

名前	詳細	説明
	変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	積レイテンシー分布 (ヒストグラム) をエクスポートできます。
hash-max-listpack-entries	許可される値: 0+ デフォルト: 512 型: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	データセットを圧縮するためのハッシュエントリの最大数。
hash-max-listpack-value	許可される値: 0+ デフォルト: 64 型: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	データセットを圧縮するための最大ハッシュエントリのしきい値。
zset-max-listpack-entries	許可される値: 0+ デフォルト: 128 型: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	データセットを圧縮するためにソートされたセットエントリの最大数。

名前	詳細	説明
zset-max-listpack-value	許可される値: 0+ デフォルト: 64 型: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	データセットを圧縮するためにソートされたセットエントリの最大しきい値。
search-enabled	許可される値: yes, no デフォルト: no タイプ: 文字列 変更可能: はい 変更は有効になります: 新しいクラスターのみ。 エンジンの最小バージョン: 7.1	[はい] に設定すると、検索機能が有効になります。

名前	詳細	説明
search-query-timeout-ms	許可される値: 1 - 60,000 デフォルト: 10,000 型: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時 エンジンの最小バージョン: 7.1	検索クエリの実行が許可されるミリ秒単位の最大時間。

Redis OSS 7 で変更されたパラメータは次のとおりです。

名前	詳細	説明
activerehashing	変更可能: no。Redis OSS 7 では、このパラメーターはデフォルトで非表示および有効になっています。無効にするには、サポートケースを作成する必要があります。	変更可能は Yes でした。

Redis OSS 7 で削除されたパラメータは次のとおりです。

名前	詳細	説明
hash-max-ziplist-entries	許可される値: 0+ デフォルト: 512 型: 整数	小さなハッシュエンコーディングを表現するために listpack を ziplist の代わりに使用する

名前	詳細	説明
	変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	
hash-max-ziplist-value	許可される値: 0+ デフォルト: 64 型: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	小さなハッシュエンコーディングを表現するために listpack を ziplist の代わりに使用する
zset-max-ziplist-entries	許可される値: 0+ デフォルト: 128 型: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	小さなハッシュエンコーディングを表現するために listpack を ziplist の代わりに使用します。
zset-max-ziplist-value	許可される値: 0+ デフォルト: 64 型: 整数 変更可能: はい 変更の適用: クラスター内のすべてのノードにわたって即時	小さなハッシュエンコーディングを表現するために listpack を ziplist の代わりに使用します。

Redis OSS 6 パラメータ

> **Note**
>
> Redis OSS エンジンバージョン 6.2 では、データ階層化 で使用するために r6gd ノードファミリーが導入された場合、r6gd ノードタイプでは `noeviction`、`volatile-lru`、および `allkeys-lru` max-memory ポリシーのみがサポートされます。

パラメータグループファミリー:memorydb_redis6

Redis OSS 6 で追加されたパラメータは次のとおりです。

名前	詳細	説明
`maxmemory-policy`	型: 文字列 許容値:volatile-lru、allkeys-lru、volatile-lfu、allkeys-lfu、volatile-random、allkeys-random、volatile-ttl、noeviction デフォルト:エビクションなし	メモリの最大使用量に到達したときのキーの削除ポリシー。 Valkey または Redis OSS を LRU キャッシュとして使用する方法の詳細については、「Key eviction」を参照してください。
`list-compress-depth`	型: 整数 許可される値: 0- デフォルト: 0	圧縮の深さは、圧縮から除外するリストの端からのクイックリスト ziplist ノードの数です。リストの先頭と末尾は、プッシュおよびポップオペレーションを高速にするために常に圧縮されません。設定は以下のとおりです。 • 0: すべての圧縮を無効にします。 • 1: 先頭から末尾までの最初のノードで圧縮を開始します。 先頭->ノード->ノード->...->ノード->末尾 先頭 と 末尾 を除くすべてのノードで圧縮を実行します。

名前	詳細	説明
		• 2: 先頭から末尾までの 2 番目のノードで圧縮を開始します。 先頭->次->ノード->ノード->...->ノード->前->末尾 先頭、次、前、末尾 は圧縮されません。他のすべてのノードで圧縮を実行します。 • その他
hll-sparse-max-bytes	型: 整数 許可される値: 1 ~ 16000 デフォルト: 3000	HyperLogLog のスパースな表示バイト制限。この制限には 16 バイトのヘッダーが含まれます。スパースな表現を使用する HyperLogLog がこの制限を超えると、デンスな表現に変換されます。 16,000 より大きい値はお勧めしません。その時点では、デンスな表現の方がメモリ効率が高くなるためです。 PFADD の速度を下げすぎることなく領域効率の良いエンコードの利点を活かせる (スパースなエンコードで O(N) になる) ように、値は約 3,000 にすることをお勧めします。問題が CPU ではなく領域であり、データセットが 0 ~ 15,000 の濃度の大量の HyperLogLog で構成されているときは、値を 10,000 まで大きくすることができます。
lfu-log-factor	型: 整数 許可される値: 1- デフォルト: 10	LFU エビクションポリシーのキーカウンターをインクリメントするためのログファクター。

名前	詳細	説明
lfu-decay-time	型: 整数 許可される値: 0- デフォルト: 1	LFU エビクションポリシーのキーカウンターをデクリメントする期間 (分単位)。
active-defrag-max-scan-fields	型: 整数 許可される値: 1 ~ 1000000 デフォルト: 1000	アクティブなデフラグメンテーション中にメインディクショナリスキャンから処理される set/hash/zset/list フィールドの最大数。
active-defrag-threshold-upper	型: 整数 許可される値: 1 ~ 100 デフォルト: 100	最大の労力を使用するフラグメントの最大割合。
client-output-buffer-limit-pubsub-hard-limit	型: 整数 許可される値: 0- デフォルト: 33554432	Redis OSS 発行/サブスクライブクライアントの場合: クライアントの出力バッファが指定されたバイト数に達した場合、クライアントの接続が切断されます。
client-output-buffer-limit-pubsub-soft-limit	型: 整数 許可される値: 0- デフォルト: 8388608	Redis OSS 発行/サブスクライブクライアントの場合: クライアントの出力バッファが指定されたバイト数に達した場合、クライアントの接続が切断されますが、この条件が client-output-buffer-limit-pubsub-soft-seconds。 の間存続した場合に限ります

名前	詳細	説明
client-output-buffer-limit-pubsub-soft-seconds	型: 整数 許可される値: 0- デフォルト: 60	Redis OSS 発行/サブスクライブクライアントの場合: クライアントの出力バッファがこの秒数より長い間 client-output-buffer-limit-pubsub-soft-limit バイトのままの場合、クライアントの接続が切断されます。
timeout	型: 整数 許可される値: 0,20- デフォルト: 0	ノードがタイムアウトまで待機する秒数。値は次のとおりです。 • 0 – アイドル状態のクライアントは切断しません。 • 1-19 – 無効な値です。 • >=20 – ノードがアイドル状態のクライアントを切断するまでに待機する秒数。
notify-keyspace-events	型: 文字列 許可される値: NULL デフォルト: NULL	Redis OSS が Pub/Sub クライアントに通知するためのキースペースイベント。デフォルトではすべての通知は無効になっています。
maxmemory-samples	型: 整数 許可される値: 1- デフォルト: 3	least-recently-used (LRU) と time-to-live (TTL) の計算の場合、このパラメータはチェックするキーのサンプルサイズを表します。デフォルトで、Redis OSS は 3 個のキーを選択し、最も長い間使用されていないキーを使用します。

名前	詳細	説明
slowlog-max-len	型: 整数 許可される値: 0- デフォルト: 128	Redis OSS スローログの最大長。この長さには制限はありません。ただ、メモリを消費することになるので注意してください。スローログが使用していたメモリは、SLOWLOG RESET. のようにして再利用することができます。
activerehashing	型: 文字列 許可される値: はい,いいえ デフォルト: はい	主要なハッシュテーブルは、1 秒あたり 10 回再ハッシュされます。再ハッシュ操作ごとに 1 ミリ秒の CPU が消費されます。 パラメータグループを作成するとき、この値を設定します。クラスターに新しいパラメータグループを割り当てるとき、この値は以前のパラメータグループと新しいパラメータグループで一致している必要があります。
client-output-buffer-limit-normal-hard-limit	型: 整数 許可される値: 0- デフォルト: 0	クライアントの出力バッファーが指定されたバイト数に達した場合、クライアントの接続が切断されます。デフォルトは 0 です (ハード制限なし)。
client-output-buffer-limit-normal-soft-limit	型: 整数 許可される値: 0- デフォルト: 0	クライアントの出力バッファーが指定されたバイト数に達した場合、クライアントの接続が切断されますが、この条件が client-output-buffer-limit-normal-soft-seconds の間存続した場合に限ります。デフォルトは 0 です (ソフト制限なし)。

名前	詳細	説明
client-output-buffer-limit-normal-soft-seconds	型: 整数 許可される値: 0- デフォルト: 0	クライアントの出力バッファーが、この秒数より長い時間 client-output-buffer-limit-normal-soft-limit バイトのままの場合、クライアントの接続が切断されます。デフォルトは 0 です (時間制限なし)。
tcp-keepalive	型: 整数 許可される値: 0- デフォルト: 300	0 以外の値 (N) に設定した場合、接続が維持されていることを確認するためにノードクライアントが N 秒ごとにポーリングされます。デフォルト設定の 0 では、このようなポーリングが行われません。
active-defrag-cycle-min	型: 整数 許可される値: 1 ~ 75 デフォルト: 5	デフラグの最小の労力 (CPU 使用率)。
stream-node-max-bytes	型: 整数 許可される値: 0- デフォルト: 4096	ストリームデータ構造は、内部の複数のアイテムをエンコードするノードの基数ツリーです。基数ツリーの単一ノードの最大サイズをバイト単位で指定するには、この設定を使用します。0 に設定されている場合、ツリーノードのサイズは無制限です。

名前	詳細	説明
stream-node-max-entries	型: 整数 許可される値: 0- デフォルト: 100	ストリームデータ構造は、内部の複数のアイテムをエンコードするノードの基数ツリーです。新しいストリームエントリを追加するとき、新しいノードに切り替える前に単一ノードに含めることができるアイテムの最大数を指定するには、この設定を使用します。0 に設定されている場合、ツリーノードのアイテムの数は無制限です。
lazyfree-lazy-eviction	型: 文字列 許可される値: はい,いいえ デフォルト: いいえ	削除で、非同期削除を実行します。
active-defrag-ignore-bytes	型: 整数 許可される値: 1048576- デフォルト: 104857600	アクティブなデフラグを開始するためのフラグメントの最小量。
lazyfree-lazy-expire	型: 文字列 許可される値: はい,いいえ デフォルト: いいえ	期限切れのキーで、非同期削除を実行します。
active-defrag-threshold-lower	型: 整数 許可される値: 1 ~ 100 デフォルト: 10	アクティブなデフラグを開始するためのフラグメントの割合。

名前	詳細	説明
active-defrag-cycle-max	型: 整数 許可される値: 1 ~ 75 デフォルト: 75	デフラグの最大の労力 (CPU 使用率)。
lazyfree-lazy-server-del	型: 文字列 許可される値: はい,いいえ デフォルト: いいえ	値を更新するコマンドに対して非同期削除を実行します。
slowlog-log-slower-than	型: 整数 許可される値: 0- デフォルト: 10000	コマンドが Redis OSS Slow Log 機能によってログに記録されるために超過すべき最大実行時間 (マイクロ秒単位)。負の数値ではスローログは無効になり、値が 0 の場合はすべてのコマンドのロギングが強制されることに注意してください。
hash-max-ziplist-entries	型: 整数 許可される値: 0- デフォルト: 512	ハッシュに使用されるメモリ量を決定します。エントリが指定された数より少ないハッシュは、領域を節約する特殊なエンコードを使用して格納されます。
hash-max-ziplist-value	型: 整数 許可される値: 0- デフォルト: 64	ハッシュに使用されるメモリ量を決定します。エントリが指定されたバイト数より小さいハッシュは、領域を節約する特殊なエンコードを使用して格納されます。

名前	詳細	説明
set-max-intset-entries	型: 整数 許可される値: 0- デフォルト: 512	特定のタイプのセットに使用されるメモリの量を決定します (64 ビット符号付き整数の範囲に収まる基数 10 の整数である文字列)。エントリが指定された数より少ないセットは、領域を節約する特殊なエンコードを使用して格納されます。
zset-max-ziplist-entries	型: 整数 許可される値: 0- デフォルト: 128	ソート対象セットに使用されるメモリ量を決定します。要素が指定された数より少ないソート対象セットは、領域を節約する特殊なエンコードを使用して格納されます。
zset-max-ziplist-value	型: 整数 許可される値: 0- デフォルト: 64	ソート対象セットに使用されるメモリ量を決定します。エントリが指定されたバイト数より小さいソート対象セットは、領域を節約する特殊なエンコードを使用して格納されます。
tracking-table-max-keys	型: 整数 許可される値: 1 ~ 100000000 デフォルト:1000000	クライアント側のキャッシュを支援するために、Redis OSS では、どのクライアントがどのキーにアクセスしたかの追跡をサポートします。 追跡されたキーが変更されると、無効化メッセージがすべてのクライアントに送信され、キャッシュされた値が無効になったことが通知されます。この値により、このテーブルの上限を指定できます。

名前	詳細	説明
acllog-max-len	型: 整数 許可される値: 1 ~ 10000 デフォルト: 128	ACL ログ内のエントリの最大数。
active-expire-effort	型: 整数 許可される値: 1 ~ 10 デフォルト: 1	Redis OSS は、2 つのメカニズムによって、有効期限を越えたキーを削除します。1 つでは、キーがアクセスされ、期限切れであることが判明します。もう 1 つでは、定期的なジョブがキーをサンプリングし、有効期限 (TTL) を超えたキーを期限切れにします。このパラメータは、Redis OSS が定期ジョブ内のアイテムを期限切れにするために使用する作業量を定義します。 デフォルト値の 1 では、期限切れのキーの 10% 以上をメモリに残さないようにします。また、合計メモリの 25% 以上を消費しないようにし、システムにレイテンシーを追加しようとします。この値を最大 10 まで増やすと、キーの期限切れに費やす労力を増やすことができます。トレードオフは、CPU が高くなると、潜在的にレイテンシーが高くなることです。メモリ使用率が高く、CPU 使用率の増加が許容される場合を除き、値 1 を推奨します。
lazyfree-lazy-user-del	型: 文字列 許可される値: はい,いいえ デフォルト: いいえ	DEL コマンドのデフォルト動作が UNLINK と同じ動作をするかどうかを指定します。

名前	詳細	説明
activedefrag	型: 文字列 許可される値: はい,いいえ デフォルト: いいえ	有効化されているアクティブなメモリのデフラグメンテーション。
maxclients	型: 整数 許容される値: 65000 デフォルト: 65000	一度に接続できるクライアントの最大数。変更不可。
client-query-buffer-limit	型: 整数 許容される値: 1048576 ~ 1073741824 デフォルト: 1073741824	単一のクライアントクエリバッファの最大サイズ。変更は直ちに行われます。
proto-max-bulk-len	型: 整数 許容される値: 1048576 ~ 536870912 デフォルト: 536870912	1 つの要素リクエストの最大サイズ。変更は直ちに行われます。

MemoryDB ノードタイプ固有のパラメータ

ほとんどのパラメータの値は 1 つですが、一部のパラメータには、使用されているノードタイプによって複数の値が設定されることがあります。次の表は、各ノードタイプの maxmemory のデフォルト値を示しています。maxmemory の値は、ノードでデータやその他の用途に使用できる最大バイト数です。

ノードの種類	Maxmemory
db.r7g.large	14037181030
db.r7g.xlarge	28261849702
db.r7g.2xlarge	56711183565
db.r7g.4xlarge	113609865216
db.r7g.8xlarge	225000375228
db.r7g.12xlarge	341206346547
db.r7g.16xlarge	450000750456
db.r6gd.xlarge	28261849702
db.r6gd.2xlarge	56711183565
db.r6gd.4xlarge	113609865216
db.r6gd.8xlarge	225000375228
db.r6g.large	14037181030
db.r6g.xlarge	28261849702
db.r6g.2xlarge	56711183565
db.r6g.4xlarge	113609865216
db.r6g.8xlarge	225000375228
db.r6g.12xlarge	341206346547
db.r6g.16xlarge	450000750456
db.t4g.small	1471026299
db.t4g.medium	3317862236

> **Note**
>
> MemoryDB インスタンスタイプはすべて Amazon 仮想プライベートクラウド (VPC) に作成する必要があります。

制限されるコマンド

マネージドサービスを提供するために、MemoryDB では高度な特権を必要とする特定のコマンドへのアクセスが制限されています。以下のコマンド使用できません。

- `acl deluser`
- `acl load`
- `acl save`
- `acl setuser`
- `bgrewriteaof`
- `bgsave`
- `cluster addslot`
- `cluster delslot`
- `cluster setslot`
- `config`
- `debug`
- `migrate`
- `module`
- `psync`
- `replicaof`
- `save`
- `shutdown`
- `slaveof`
- `sync`

チュートリアル: Amazon VPC の MemoryDB にアクセスする Lambda 関数の設定

このチュートリアルの学習内容は次のとおりです。

- デフォルトの us-east-1 リージョンの Amazon Virtual Private Cloud (Amazon VPC) に MemoryDB クラスターを作成します。
- クラスターにアクセスするための Lambda 関数を作成します。Lambda 関数を作成する際には、Lambda 関数で VPC 内のリソースにアクセスできるように、Amazon VPC 内のサブネット ID と VPC セキュリティグループを指定します。このチュートリアルでは、例示のため、この Lambda 関数で UUID の生成、クラスターへの書き込み、クラスターからの取得を行います。
- Lambda 関数を手動で呼び出し、この関数が VPC 内のクラスターにアクセスしたことを確認します。
- このチュートリアルに設定された Lambda 関数、クラスター、IAM ロールをクリーンアップします。

トピック

- ステップ 1: クラスターを作成する
- ステップ 2: Lambda 関数を作成する
- ステップ 3: Lambda 関数をテストする
- ステップ 4: クリーンアップする (オプション)

ステップ 1: クラスターを作成する

クラスターを作成するには、次のステップに従います。

クラスターを作成する

このステップでは、AWS Command Line Interface (CLI) を使用して、アカウントの us-east-1 リージョンにあるデフォルトの Amazon VPC にクラスターを作成します。MemoryDB コンソールまたは API を使用してクラスターを作成する方法については、「ステップ 2: クラスターを作成する」を参照してください。

```
aws memorydb create-cluster --cluster-name cluster-01 --engine-version 7.0 --acl-name
 open-access \
```

```
--description "MemoryDB IAM auth application" \
--node-type db.r6g.large
```

[ステータス] フィールドの値が CREATING に設定されていることに注意してください。MemoryDB がクラスターの作成を完了するまで、数分かかる場合があります。

クラスターエンドポイントのコピー

describe-clusters コマンドを使用して MemoryDB がクラスターの作成を完了したことを確認します。

```
aws memorydb describe-clusters \
--cluster-name cluster-01
```

出力に表示されたクラスターエンドポイントアドレスをコピーします。Lambda 関数のデプロイパッケージを作成するときに、このアドレスが必要になります。

IAM ロールを作成する

1. アカウントが新しいロールを引き継ぐことを許可するロール用の IAM 信頼ポリシードキュメントを以下に示すように作成します。ポリシーを trust-policy.json というファイルに保存します。このポリシーの account_id 123456789012 は account_id に置き換えてください。

   JSON

   ```
   {
   "Version":"2012-10-17",
       "Statement": [{
           "Effect": "Allow",
           "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
           "Action": "sts:AssumeRole"
       },
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "lambda.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }]
   ```

```
}
```

2. 以下に示すように、IAM ポリシードキュメントを作成します。ポリシーを policy.json というファイルに保存します。このポリシーの account_id 123456789012 は account_id に置き換えてください。

   JSON

```
{
"Version":"2012-10-17",
  "Statement": [
    {
      "Effect" : "Allow",
      "Action" : [
        "memorydb:Connect"
      ],
      "Resource" : [
        "arn:aws:memorydb:us-east-1:123456789012:cluster/cluster-01",
        "arn:aws:memorydb:us-east-1:123456789012:user/iam-user-01"
      ]
    }
  ]
}
```

3. IAM ロールを作成します。

```
aws iam create-role \
--role-name "memorydb-iam-auth-app" \
--assume-role-policy-document file://trust-policy.json
```

4. IAM ポリシーを作成します。

```
aws iam create-policy \
  --policy-name "memorydb-allow-all" \
  --policy-document file://policy.json
```

5. IAM ポリシーをロールにアタッチします。この policy-arn の account_id 123456789012 を account_id に置き換えてください。

```
aws iam attach-role-policy \
 --role-name "memorydb-iam-auth-app" \
```

```
 --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"
```

アクセスコントロールリスト (ACL) を作成します

1. IAM を有効にしている新しいユーザーを作成します。

```
aws memorydb create-user \
  --user-name iam-user-01 \
--authentication-mode Type=iam \
--access-string "on ~* +@all"
```

2. ACL を作成し、クラスターにアタッチします。

```
aws memorydb create-acl \
  --acl-name iam-acl-01 \
  --user-names iam-user-01

aws memorydb update-cluster \
  --cluster-name cluster-01 \
  --acl-name iam-acl-01
```

ステップ 2: Lambda 関数を作成する

Lambda 関数を作成するには、以下の手順を実行します。

デプロイパッケージの作成

このチュートリアルでは、Lambda 関数のコード例を Python で示します。

Python

次の例の Python コードでは、MemoryDB クラスターに対して項目の読み取り/書き込みを行います。コードを app.py という名前のファイルに保存します。コード内の `cluster_endpoint` 値を、ステップ 1 でコピーしたエンドポイントアドレスに必ず置き換えてください。

```
from typing import Tuple, Union
from urllib.parse import ParseResult, urlencode, urlunparse

import botocore.session
import redis
```

```
from botocore.model import ServiceId
from botocore.signers import RequestSigner
from cachetools import TTLCache, cached
import uuid

class MemoryDBIAMProvider(redis.CredentialProvider):
    def __init__(self, user, cluster_name, region="us-east-1"):
        self.user = user
        self.cluster_name = cluster_name
        self.region = region

        session = botocore.session.get_session()
        self.request_signer = RequestSigner(
            ServiceId("memorydb"),
            self.region,
            "memorydb",
            "v4",
            session.get_credentials(),
            session.get_component("event_emitter"),
        )

    # Generated IAM tokens are valid for 15 minutes
    @cached(cache=TTLCache(maxsize=128, ttl=900))
    def get_credentials(self) -> Union[Tuple[str], Tuple[str, str]]:
        query_params = {"Action": "connect", "User": self.user}

        url = urlunparse(
            ParseResult(
                scheme="https",
                netloc=self.cluster_name,
                path="/",
                query=urlencode(query_params),
                params="",
                fragment="",
            )
        )
        signed_url = self.request_signer.generate_presigned_url(
            {"method": "GET", "url": url, "body": {}, "headers": {}, "context": {}},
            operation_name="connect",
            expires_in=900,
            region_name=self.region,
        )
        # RequestSigner only seems to work if the URL has a protocol, but
        # MemoryDB only accepts the URL without a protocol
```

```
        # So strip it off the signed URL before returning
        return (self.user, signed_url.removeprefix("https://"))

def lambda_handler(event, context):
    username = "iam-user-01" # replace with your user id
    cluster_name = "cluster-01" # replace with your cache name
    cluster_endpoint = "clustercfg.cluster-01.xxxxxx.memorydb.us-east-1.amazonaws.com"
 # replace with your cluster endpoint
    creds_provider = MemoryDBIAMProvider(user=username, cluster_name=cluster_name)
    redis_client = redis.Redis(host=cluster_endpoint, port=6379,
 credential_provider=creds_provider, ssl=True, ssl_cert_reqs="none")

    key='uuid'
    # create a random UUID - this will be the sample element we add to the cluster
    uuid_in = uuid.uuid4().hex
    redis_client.set(key, uuid_in)
    result = redis_client.get(key)
    decoded_result = result.decode("utf-8")
    # check the retrieved item matches the item added to the cluster and print
    # the results
    if decoded_result == uuid_in:
        print(f"Success: Inserted {uuid_in}. Fetched {decoded_result} from MemoryDB.")
    else:
        raise Exception(f"Bad value retrieved. Expected {uuid_in}, got
 {decoded_result}")

    return "Fetched value from MemoryDB"
```

このコードは Python redis-py ライブラリを使用してアイテムをクラスターに格納し、取得します。このコードでは、cachetools を使用して、生成された IAM 認証トークンを 15 分間キャッシュします。redis-py および cachetools を含むデプロイパッケージを作成するには、次のステップを実行します。

app.py ソースコードファイルを含むプロジェクトディレクトリに、redis-py および cachetools ライブラリをインストールするフォルダを作成します。

```
mkdir package
```

pip を使用して redis-py および cachetools をインストールします。

```
pip install --target ./package redis
pip install --target ./package cachetools
```

redis-py および cachetools ライブラリを含む zip ファイルを作成します。Linux および MacOS では、次のコマンドを実行します。Windows では、任意の zip ユーティティを使用して、redis-py および cachetools ライブラリをルートに置く .zip ファイルを作成します。

```
cd package
zip -r ../my_deployment_package.zip .
```

.zip ファイルに関数コードを追加します。Linux および macOS では、次のコマンドを実行します。Windows では、任意の zip ユーティリティを使用して .zip ファイルのルートに app.py を追加します。

```
cd ..
zip my_deployment_package.zip app.py
```

IAM ロール (実行ロール) を作成します

AWSLambdaVPCAccessExecutionRole という名前の AWS マネージドポリシーをロールにアタッチします。

```
aws iam attach-role-policy \
 --role-name "memorydb-iam-auth-app" \
 --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"
```

デプロイパッケージをアップロードします (Lambda 関数を作成する)

このステップでは、create-function AWS CLI コマンドを使用して Lambda 関数 (AccessMemoryDB) を作成します。

デプロイパッケージの.zip ファイルを含むプロジェクトディレクトリから、次の Lambda CLI create-function コマンドを実行します。

ロールのオプションには、前のステップで作成した実行ロールの ARN を使用します。vpc-config には、デフォルト VPC のサブネットとデフォルト VPC のセキュリティグループ ID をカンマで区切ったリストを入力します。これらの値は Amazon VPC コンソール にあります。デフォルト VPC のサブネットを確認するには、[VPC] を選択し、次にお使いの AWS アカウントのデフォルト VPC を選択します。この VPC のセキュリティグループを確認するには、[セキュリティ] に移動して [セキュリティグループ] を選択します。us-east-1 リージョンが選択されていることを確認します。

```
aws lambda create-function \
```

```
--function-name AccessMemoryDB  \
--region us-east-1 \
--zip-file fileb://my_deployment_package.zip \
--role arn:aws:iam::123456789012:role/memorydb-iam-auth-app \
--handler app.lambda_handler \
--runtime python3.12  \
--timeout 30 \
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-
security-group-id
```

ステップ 3: Lambda 関数をテストする

このステップでは、invoke コマンドを使用して Lambda 関数を手動で呼び出します。Lambda 関数を実行すると、UUID が生成され、Lambda コードで指定した ElastiCache キャッシュにその UUID が書き込まれます。次に、Lambda 関数はキャッシュから項目を取得します。

1. AWS Lambda invoke コマンドを使用して Lambda 関数 (AccessMemoryDB) を呼び出します。

   ```
   aws lambda invoke \
   --function-name AccessMemoryDB  \
   --region us-east-1 \
   output.txt
   ```

2. Lambda 関数が正常に実行されたことを、次のように確認します。

   - output.txt ファイルを確認します。
   - CloudWatch コンソールを開き、関数のロググループ (/aws/lambda/AccessRedis) を選択して、CloudWatch ログの結果を検証します。ログストリームには、以下と同様のコマンドの出力が含まれます。

     ```
     Success: Inserted 826e70c5f4d2478c8c18027125a3e01e. Fetched
      826e70c5f4d2478c8c18027125a3e01e from MemoryDB.
     ```

   - AWS Lambda コンソールで結果を確認します。

ステップ 4: クリーンアップする (オプション)

クリーンアップするには、以下の手順を実行します。

Lambda 関数を削除する

```
aws lambda delete-function \
 --function-name AccessMemoryDB
```

MemoryDB クラスターの削除

クラスターを削除します。

```
aws memorydb delete-cluster \
 --cluster-name cluster-01
```

ユーザーと ACL を削除します。

```
aws memorydb delete-user \
 --user-id iam-user-01

aws memorydb delete-acl \
 --acl-name iam-acl-01
```

IAM ロールとポリシーを削除する

```
aws iam detach-role-policy \
 --role-name "memorydb-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"

aws iam detach-role-policy \
--role-name "memorydb-iam-auth-app" \
--policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"

aws iam delete-role \
 --role-name "memorydb-iam-auth-app"

 aws iam delete-policy \
  --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"
```

ベクトル検索

MemoryDB のベクトル検索は、MemoryDB の機能を拡張します。ベクトル検索は、既存の MemoryDB 機能と組み合わせて使用できます。ベクトル検索を使用しないアプリケーションは、ベクトル検索が存在していることによる影響を受けません。ベクトル検索は、MemoryDB が利用可能なすべてのリージョンで使用できます。

ベクトル検索は、高速ベクトル検索を実現しながらアプリケーションのアーキテクチャを簡素化します。MemoryDB のベクトル検索は、ピークのパフォーマンスとスケールが最も重要な選択基準であるユースケースに最適です。既存の MemoryDB データ、または Valkey または Redis OSS API を使用して、機械学習と生成 AI のユースケースを構築できます。これには、取得拡張生成、異常検出、ドキュメント取得、リアルタイムレコメンデーションが含まれます。

2024 年 6 月 26 日の時点で、AWS MemoryDB は AWS の一般的なベクトルデータベースの中で最高のリコール率で最速のベクトル検索パフォーマンスを提供します。

トピック

- ベクトル検索の概要
- ユースケース
- ベクター検索の機能と制限
- ベクトル検索が有効になっているクラスターを作成する
- ベクトル検索コマンド

ベクトル検索の概要

ベクトル検索は、インデックスの作成、メンテナンス、使用を基盤として構築されています。各ベクトル検索オペレーションは単一のインデックスを指定し、そのオペレーションはそのインデックスに限定されます。すなわち、1 つのインデックスに対するオペレーションは、他のインデックスに対するオペレーションの影響を受けません。インデックスの作成および破棄のオペレーションを除き、任意のインデックスに対して任意の数のオペレーションをいつでも発行できます。これは、クラスターレベルでは、複数のインデックスに対する複数のオペレーションが同時に進行する可能性があることを意味します。

個々のインデックスは、他の Valkey および Redis OSS 名前空間 (キー、関数など) とは異なる、一意の名前空間に存在する名前付きオブジェクトです。各インデックスは、列と行の 2 つの次元で構造化されているという点で、概念的には従来のデータベーステーブルと似ています。テーブル内の各

行はキーに対応します。インデックス内の各列は、そのキーのメンバーまたは部分に対応します。このドキュメント内では、キー、行、レコードという用語は同一のものであり、相互互換的に使用されます。同様に、列、フィールド、パス、メンバーという用語は本質的に同一のものであり、これらも相互互換的に使用されます。

インデックス付きデータを追加、削除、変更するための特別なコマンドはありません。むしろ、インデックス内のキーを変更する既存の HASH または JSON コマンドが、インデックスも自動的に更新します。

トピック

- インデックスと Valkey および Redis OSS キースペース
- インデックスフィールドの型
- ベクトルインデックスアルゴリズム
- ベクトル検索のクエリ式
- INFO コマンド
- ベクトル検索のセキュリティ

インデックスと Valkey および Redis OSS キースペース

インデックスは、Valkey および Redis OSS キースペースのサブセット上で構築および維持されます。複数のインデックスは、制限なく、キースペースの素のサブセットまたは重複するサブセットを選択できます。各インデックスのキースペースは、インデックスの作成時に提供されるキープレフィックスのリストによって定義されます。プレフィックスのリストはオプションであり、省略した場合、キースペース全体がそのインデックスの一部になります。また、インデックスは、型が一致するキーのみをカバーするという点で型指定されます。現在、JSON インデックスと HASH インデックスのみがサポートされています。HASH インデックスは、プレフィックスリストによってカバーされる HASH キーのインデックスのみを作成し、同様に、JSON インデックスは、プレフィックスリストによってカバーされる JSON キーのインデックスのみを作成します。インデックスのキースペースプレフィックスリスト内のキーのうち、指定されたタイプを持たないキーは無視され、検索オペレーションに影響を及ぼしません。

HASH または JSON コマンドがインデックスのキースペース内にあるキーを変更すると、そのインデックスが更新されます。このプロセスには、各インデックスについて宣言されたフィールドを抽出し、新しい値でインデックスを更新することが含まれます。更新プロセスはバックグラウンドスレッドで実行されます。これは、インデックスが最終的にキースペースの内容と一致することを意味します。そのため、キーの挿入または更新は、すぐに検索結果に表示されません。システム負荷および/

またはデータのミューテーションが多い期間中は、表示できるようになるまでの遅延が長くなる可能性があります。

インデックスの作成は複数のステップからなるプロセスです。最初のステップは、インデックスを定義する FT.CREATE コマンドを実行することです。作成が正常に実行されると、2 番目のステップであるバックフィルが自動的に開始されます。バックフィルプロセスはバックグラウンドスレッドで実行され、キースペースをスキャンして、新しいインデックスのプレフィックスリスト内にあるキーを探します。見つかった各キーはインデックスに追加されます。最終的にはキースペース全体がスキャンされて、インデックス作成プロセスが完了します。バックフィルプロセスの実行中は、インデックス付きキーのミューテーションが許可され、制限はなく、すべてのキーのインデックスが適切に作成されるまではインデックスバックフィルプロセスが完了しないことに留意してください。インデックスのバックフィル中に試行されるクエリオペレーションは許可されず、エラーで終了します。バックフィルプロセスの完了は、そのインデックスについての FT.INFO コマンドの出力 (「backfill_status」) から判断できます。

インデックスフィールドの型

インデックスの各フィールド (列) には、インデックスの作成時に宣言される特定の型と、キー内の場所が含まれます。HASH キーの場合、その場所は HASH 内のフィールド名です。JSON キーの場合、その場所は JSON パスの説明です。キーが変更されると、宣言されたフィールドに関連付けられたデータが抽出され、宣言された型に変換されてインデックスに格納されます。データが欠落している場合、または宣言された型に正常に変換できない場合、そのフィールドはインデックスから省略されます。次で説明するように、フィールドには 4 つのタイプがあります:

- 数値フィールドには 1 つの数値が含まれます。JSON フィールドの場合、JSON 数値の数値規則に従う必要があります。HASH の場合、フィールドには、固定小数点数または浮動小数点数の標準形式で記述された数値の ASCII テキストが含まれることが想定されます。キー内の表現にかかわらず、このフィールドはインデックス内に格納するために 64 ビットの浮動小数点数に変換されます。数値フィールドは範囲検索演算子とともに使用できます。基になる数値は精度制限のある浮動小数点で格納されるため、浮動小数点数の数値比較に関する通常のルールが適用されます。
- タグフィールドには、単一の UTF-8 文字列としてコード化された 0 個以上のタグ値が含まれます。文字列は、先頭と末尾の空白が削除された区切り文字 (デフォルトはカンマだが、上書きが可能) を使用してタグ値に解析されます。単一のタグフィールドには、任意の数のタグ値を含めることができます。タグフィールドを使用すると、大文字と小文字を区別する比較または大文字と小文字を区別しない比較で、タグ値の同等性についてクエリをフィルタリングできます。

- テキストフィールドには、UTF-8 に準拠している必要のないバイトの BLOB が含まれています。テキストフィールドを使用すると、アプリケーションにとって意味のある値でクエリ結果を装飾できます。例えば、URL やドキュメントのコンテンツなどです。
- ベクトルフィールドには、埋め込みとも呼ばれる数値のベクトルが含まれます。ベクトルフィールドは、指定されたアルゴリズムと距離メトリクスを使用した固定サイズのベクトルの K 最近傍検索 (KNN) をサポートします。HASH インデックスの場合、フィールドにはバイナリ形式 (リトルエンディアン IEEE 754) でエンコードされたベクトル全体が含まれている必要があります。JSON キーの場合、パスは数字が入力されている正しいサイズの配列を参照する必要があります。なお、JSON 配列をベクトルフィールドとして使用すると、JSON キー内の配列の内部表現が、選択したアルゴリズムに必要な形式に変換され、メモリ消費量と精度が削減されます。JSON コマンドを使用した後続の読み取りオペレーションでは、精度の値が小さくなります。

ベクトルインデックスアルゴリズム

2 つのベクトルインデックスアルゴリズムが提供されています。

- Flat – Flat アルゴリズムは、インデックス内の各ベクトルの総当たり線形処理であり、距離計算の精度の範囲内で正確な答えを生成します。インデックスの線形処理のため、インデックスが大きい場合、このアルゴリズムの実行時間が非常に長くなる可能性があります。
- HNSW (Hierarchical Navigable Small Worlds) - HNSW アルゴリズムは、実行時間を大幅に短縮する代わりに、正しい答えの近似値を提供する代替手段です。このアルゴリズムは、`M`、`EF_CONSTRUCTION`、`EF_RUNTIME` の 3 つのパラメータによって制御されます。最初の 2 つのパラメータはインデックスの作成時に指定され、変更できません。`EF_RUNTIME` パラメータにはインデックスの作成時に指定されるデフォルト値が含まれていますが、その後の個々のクエリオペレーションで上書きできます。これらの 3 つのパラメータは相互に影響し合って、取り込みおよびクエリオペレーション中のメモリと CPU 消費のバランスを取るだけでなく、正確な KNN 検索の近似値の質 (再現率と呼ばれます) を制御します。

両方のベクトル検索アルゴリズム (Flat および HNSW) は、オプションの `INITIAL_CAP` パラメータをサポートしています。このパラメータを指定すると、インデックス用にメモリが事前に割り当てられるため、メモリ管理のオーバーヘッドが削減され、ベクトルの取り込み速度が向上します。

HNSW などのベクトル検索アルゴリズムは、以前に挿入されたベクトルの削除または上書きを効率的に処理できない場合があります。これらのオペレーションを使用すると、インデックスメモリが過剰に消費されたり、再現の質が低下したりする可能性があります。インデックスの再作成は、最適なメモリ使用量および/または再現を復元するための 1 つの方法です。

ベクトル検索のクエリ式

FT.SEARCH および FT.AGGREGATE コマンドにはクエリ式が必要です。この式は、1 つ以上の演算子で構成される単一の文字列パラメータです。各演算子はインデックス内の 1 つのフィールドを使用して、インデックス内のキーのサブセットを識別します。ブール結合子や括弧を使用して複数の演算子を結合し、収集されたキーのセット (または結果セット) をさらに拡張または制限できます。

ワイルドカード

ワイルドカード演算子であるアスタリスク (「*」) は、インデックス内のすべてのキーと一致します。

数値範囲

数値範囲演算子の構文は次のとおりです:

```
<range-search> ::= '@' <numeric-field-name> ':' '[' <bound> <bound> ']'
<bound>  ::= <number> | '(' <number>
<number> ::= <integer> | <fixed-point> | <floating-point> | 'Inf' | '-Inf' | '+Inf'
```

<numeric-field-name> は、NUMERIC の型の宣言されたフィールドである必要があります。デフォルトでは、境界は包括的ですが、先頭の開き括弧 [「(」] を使用して境界を排他的にすることができます。範囲検索は、Inf、+Inf、または -Inf を使用して単一のリレーショナル比較 (<、<=、>、>=) に変換できます。指定された数値形式 (整数、固定小数点、浮動小数点、無限大) にかかわらず、数値は比較を実行するために 64 ビットの浮動小数点に変換され、それに応じて精度が低下します。

Example 例

```
@numeric-field:[0 10]                    // 0   <= <value> <= 10
@numeric-field:[(0 10]                   // 0   <  <value> <= 10
@numeric-field:[0 (10]                   // 0   <= <value> <  10
@numeric-field:[(0 (10]                  // 0   <  <value> <  10
@numeric-field:[1.5 (Inf]                // 1.5 <= value
```

タグ比較

タグ比較演算子の構文は次のとおりです:

```
<tag-search> ::= '@' <tag-field-name> ':' '{' <tag> [ '|' <tag> ]* '}'
```

演算子のタグのいずれかがレコードのタグフィールドのタグのいずれかに一致する場合、そのレコードは結果セットに含まれます。<tag-field-name> によって設計されるフィールドは、型 TAG で宣言されたインデックスのフィールドである必要があります。タグ比較の例は次のとおりです:

```
@tag-field:{ atag }
@tag-field: { tag1 | tag2 }
```

ブール値の組み合わせ

数値演算子またはタグ演算子の結果セットは、次のブール論理を使用して組み合わせることができます: and/or。括弧を使用すると、演算子をグループ化したり、評価順序を変更したりできます。ブール論理演算子の構文は次のとおりです:

```
<expression> ::= <phrase> | <phrase> '|' <expression> | '(' <expression> ')'
<phrase> ::= <term> | <term> <phrase>
<term> ::= <range-search> | <tag-search> | '*'
```

語句に結合された複数の用語は「and」で結合されます。パイプ (「|」) で結合された複数の語句は「or」で結合されます。

ベクトル検索

ベクトルインデックスは、最も近い近隣と範囲の 2 つの異なる検索方法をサポートしています。最も近い近隣検索では、指定された (参照) ベクトルに最も近いインデックス内のベクトルの数 K が検索されます。これは、「K」に最も近い近隣の KNN と呼ぶものです。KNN 検索の構文は次のとおりです。

```
<vector-knn-search> ::= <expression> '=>[KNN' <k> '@' <vector-field-name> '$'
 <parameter-name> <modifiers> ']'
<modifiers> ::= [ 'EF_RUNTIME' <integer> ] [ 'AS' <distance-field-name>]
```

ベクトル KNN 検索は、<expression> (上記で定義した演算子の任意の組み合わせ: ワイルドカード、範囲検索、タグ検索、および/またはそれらのブール値の組み合わせ) を満たすベクトルにのみ適用されます。

- <k> は、返される最近傍ベクトルの数を指定する整数です。
- <vector-field-name> は、型 VECTOR の宣言されたフィールドを指定する必要があります。
- <parameter-name> フィールドは、FT.SEARCH または FT.AGGREGATE コマンドの PARAM テーブルのエントリの 1 つを指定します。このパラメータは、距離計算の参照ベクトル値です。ベク

トルの値は、リトルエンディアン IEEE 754 バイナリ形式の PARAM の値にエンコードされます (HASH ベクトルフィールドの場合と同じようにエンコードされます)。

- 型が HNSW であるベクトルインデックスの場合、オプションの EF_RUNTIME 句を使用して、インデックスの作成時に設定された EF_RUNTIME パラメータのデフォルト値を上書きできます。
- オプションの <distance-field-name> は、参照ベクトルと見つかったキーの間の計算された距離を含む結果セットのフィールド名を提供します。

範囲検索は、参照ベクトルから指定された距離 (半径) 内のすべてのベクトルを検索します。範囲検索の構文は次のとおりです。

```
<vector-range-search> ::= ‘@’ <vector-field-name> ‘:’ ‘[’ ‘VECTOR_RANGE’ ( <radius> |
 ‘$’ <radius-parameter> )  $<reference-vector-parameter> ‘]’ [ ‘=’ ‘>’ ‘{’ <modifiers>
 ‘}’ ]
<modifiers> ::= <modifier> | <modifiers>, <modifier>
<modifer> ::= [ ‘$yield_distance_as’ ‘:’ <distance-field-name> ] [ ‘$epsilon’ ‘:’
 <epsilon-value> ]
```

コードの説明は以下のとおりです。

- <vector-field-name> は、検索するベクトルフィールドの名前です。
- <radius> or $<radius-parameter> は、検索の距離制限の数値です。
- $<reference-vector-parameter> は、参照ベクトルを含むパラメータの名前です。ベクトルの値は、リトルエンディアン IEEE 754 バイナリ形式の PARAM の値にエンコードされます (HASH ベクトルフィールドの場合と同じようにエンコードされます)。
- オプションの <distance-field-name> は、参照ベクトルと見つかったキーの間の計算された距離を含む結果セットのフィールド名を提供します。
- オプションの <epsilon-value> は検索オペレーションの境界を制御し、距離 <radius> * (1.0 + <epsilon-value>) 内のベクトルは候補結果を探して横断されます。デフォルトは .01 です。

INFO コマンド

ベクトル検索は、統計とカウンターのいくつかの追加セクションで Valkey および Redis OSS INFO コマンドを強化します。セクション SEARCH を取得するリクエストは、次のすべてのセクションを取得します:

`search_memory` セクション

名前	説明
search_used_memory_bytes	すべての検索データ構造で消費されるメモリのバイト数
search_used_memory_human	上記の人間が読めるバージョン

`search_index_stats` セクション

名前	説明
search_number_of_indexes	作成されたインデックスの数
search_num_fulltext_indexes	すべてのインデックス内の非ベクトルフィールドの数
search_num_vector_indexes	すべてのインデックス内のベクトルフィールドの数
search_num_hash_indexes	HASH 型キーのインデックスの数
search_num_json_indexes	JSON 型キーのインデックスの数
search_total_indexed_keys	すべてのインデックス内のキーの総数
search_total_indexed_vectors	すべてのインデックス内のベクトルの総数
search_total_indexed_hash_keys	すべてのインデックス内の型が HASH であるキーの総数
search_total_indexed_json_keys	すべてのインデックス内の型が JSON であるキーの総数
search_total_index_size	すべてのインデックスによって使用されるバイト数

名前	説明
search_total_fulltext_index_size	非ベクトルインデックス構造によって使用されるバイト数
search_total_vector_index_size	ベクトルインデックス構造によって使用されるバイト数
search_max_index_lag_ms	最後の取り込みバッチ更新時の取り込みの遅延

`search_ingestion` セクション

名前	説明
search_background_indexing_status	取り込みのステータス。`NO_ACTIVITY` はアイドル状態であることを意味します。他の値は、取り込み中のキーがあることを示します。
search_ingestion_paused	再起動中を除き、これは常に「no」である必要があります。

`search_backfill` セクション

> **Note**
>
> このセクションに記載されているフィールドの一部は、操作しているときにバックフィルが進行中の場合にのみ表示されます。

名前	説明
search_num_active_backfills	現在のバックフィルアクティビティの数
search_backfills_paused	メモリ不足の場合を除いて、これは常に「no」である必要があります。

名前	説明
search_current_backfill_progress_percentage	現在のバックフィルの完了率 (0 ~ 100)

`search_query` セクション

名前	説明
search_num_active_queries	現在進行中の `FT.SEARCH` および `FT.AGGREGATE` コマンドの数

ベクトル検索のセキュリティ

コマンドアクセスとデータアクセスの両方のための ACL (アクセスコントロールリスト) セキュリティメカニズムは、検索機能を制御するために拡張されています。個々の検索コマンドの ACL コントロールは完全にサポートされています。新しい ACL カテゴリ `@search` が提供され、既存のカテゴリの多く (`@fast`、`@read`、`@write` など) が新しいコマンドを含むように更新されます。検索コマンドはキーデータを変更しません。これは、書き込みアクセス用の既存の ACL 機構が保持されることを意味します。HASH および JSON オペレーションのアクセスルールは、インデックスが存在することで変更されることはありません。これらのコマンドには、通常のキーレベルのアクセスコントロールが引き続き適用されます。

また、インデックスを使用した検索コマンドでは、ACL を通じてアクセスが制御されます。アクセスチェックは、キーごとのレベルではなく、インデックス全体のレベルで実行されます。これは、そのインデックスのキースペースプレフィックスリストに含まれているすべての可能なキーにアクセスするための許可がユーザーに付与されている場合にのみ、インデックスに対するアクセスがユーザーに付与されることを意味します。つまり、インデックスの実際の内容はアクセスを制御しません。むしろ、これは、セキュリティチェックのために使用されるプレフィックスリストによって定義されるインデックスの理論的な内容です。キーに対する読み取りおよび/または書き込みアクセスがユーザーに付与されているにもかかわらず、そのキーを含むインデックスにアクセスできない状況が容易に発生する可能性があります。インデックスの作成または使用にはキースペースに対する読み取りアクセスのみが必要であり、書き込みアクセスの有無は考慮されないことに留意してください。

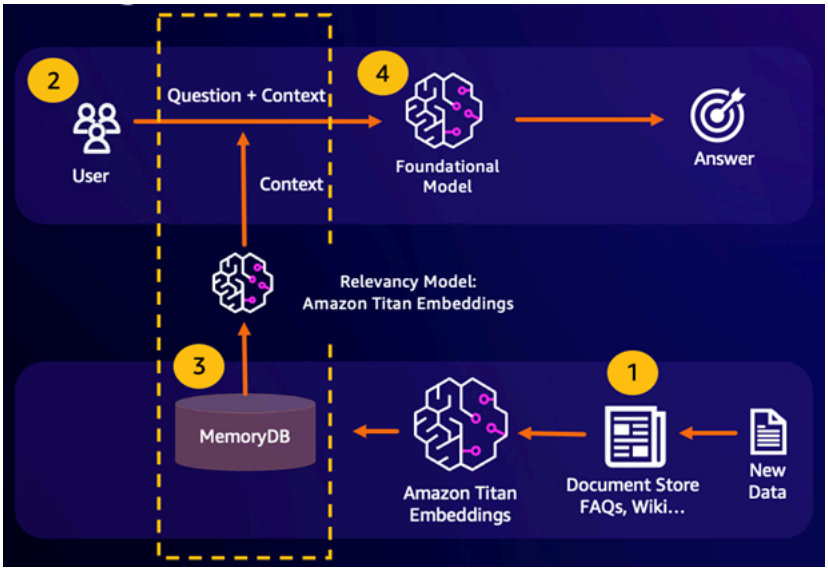
MemoryDB での ACL の使用の詳細については、「アクセスコントロールリスト (ACL) によるユーザー認証」を参照してください。

ユースケース

ベクトル検索のユースケースを次に示します。

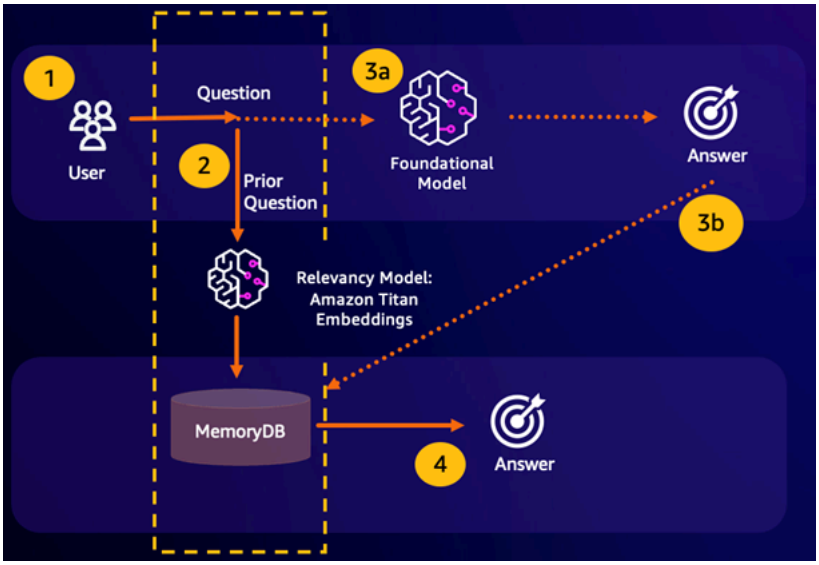
取得拡張生成 (RAG)

検索拡張生成 (RAG) は、ベクトル検索を活用して、大規模なデータコーパスから関連するパッセージを取得し、大規模言語モデル (LLM) を拡張します。具体的には、エンコーダーは入力コンテキストと検索クエリをベクトルに埋め込み、近似最近傍検索を使用して意味的に類似したパッセージを見つけます。これらの取得されたパッセージは元のコンテキストと連結され、LLM に追加の関連情報を提供し、より正確な応答をユーザーに返します。



耐久性のあるセマンティックキャッシュ

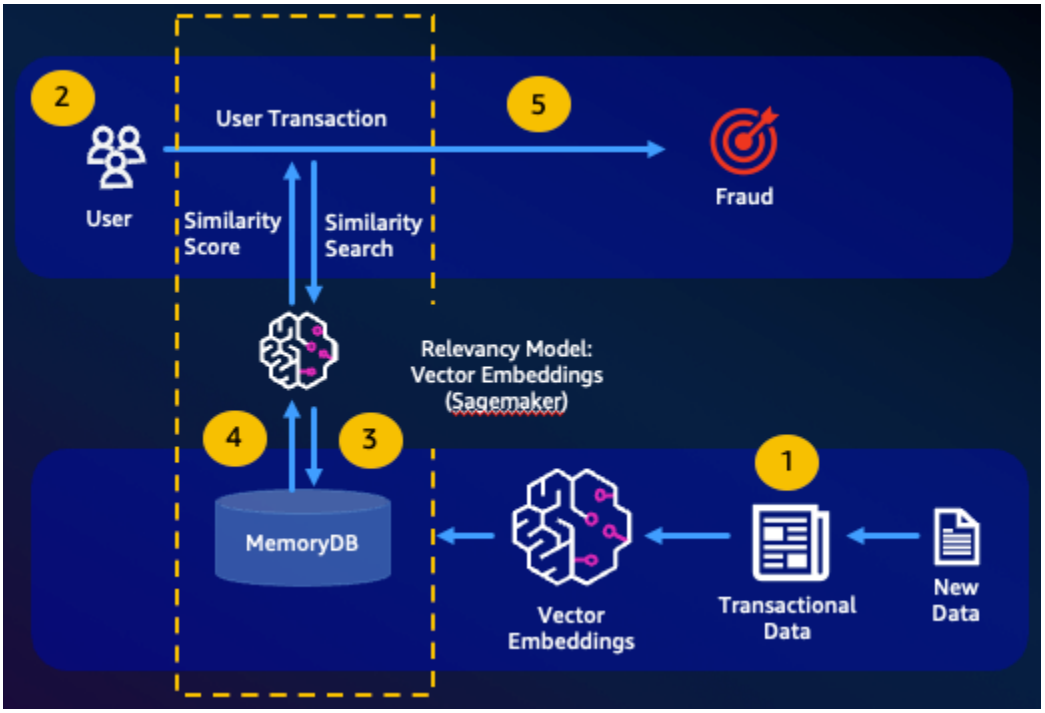
セマンティックキャッシュは、FM の以前の結果を保存して計算コストを削減するプロセスです。セマンティックキャッシュは、過去の推論の以前の結果を再計算するのではなく、再利用することにより、FM を通じた推論中に必要な計算量を削減します。MemoryDB は、耐久性のあるセマンティックキャッシュを可能にし、過去の推論のデータ損失を回避します。これにより、生成 AI アプリケーションは、不要な LLM 推論を回避してコストを削減しながら、以前の意味的に類似した質問からの回答を 1 桁ミリ秒以内に応答できます。



- セマンティック検索のヒット – お客様のクエリが以前の質問に対する定義された類似性スコアに基づいて意味的に類似している場合、FM バッファメモリ (MemoryDB) は、ステップ 4 で以前の質問に対する回答を返し、ステップ 3 を通じて FM を呼び出しません。これにより、基盤モデル (FM) のレイテンシーと発生するコストが回避することができ、より高速なエクスペリエンスがお客様に提供されます。
- セマンティック検索ミス – お客様のクエリが、以前のクエリに対する、定義された類似性スコアに基づいて意味的に類似していない場合、お客様は、ステップ 3a でお客様に応答を提供するように FM を呼び出します。FM から生成された応答は、意味的に類似した質問に対する FM のコストを最小限に抑えるために、将来のクエリのためにベクトルとして MemoryDB に保存されます (ステップ 3b)。このフローでは、意味的に類似した質問が元のクエリに含まれていないため、ステップ 4 は呼び出されません。

不正検出

異常検出の一種である不正検出は、純新規トランザクションのベクトル表現を比較しながら、有効なトランザクションをベクトルとして表現します。不正は、これらの純新規トランザクションが、有効なトランザクションデータを表すベクトルとの類似性が低い場合に検出されます。これにより、考えられるあらゆる不正行為事例の予測を試みるのではなく、通常の動作をモデル化することで不正を検出できます。MemoryDB を使用すると、組織は、高スループット時に、誤検知を最小限に抑えながら、1 桁ミリ秒のレイテンシーでこれを実行できます。



その他のユースケース

- レコメンデーションエンジンは、項目をベクトルとして表現することで、類似した製品やコンテンツをユーザーのために見つけることができます。ベクトルは、属性とパターンを分析することによって作成されます。ユーザーのパターンと属性に基づいて、ユーザーに合わせて既に肯定的に評価されている最も類似したベクトルを見つけることで、これまでに表示されていない新しい項目をユーザーに推奨できます。
- 文書検索エンジンは、数値の密なベクトルとしてテキスト文書を表現し、意味論的意味を捉えます。検索時に、エンジンは検索クエリをベクトルに変換し、近似最近傍検索を使用して、クエリに対して最も類似したベクトルを持つドキュメントを検索します。このベクトル類似性アプローチにより、単にキーワードを照合するのではなく、意味に基づいてドキュメントを照合できます。

ベクター検索の機能と制限

ベクトル検索が利用可能なリージョン

ベクトル検索対応の MemoryDB 設定は、R6g, R7g 、T4g ノードタイプでサポートされており、MemoryDB が利用可能なすべての AWS リージョンで使用できます。

既存のクラスターは、検索を有効にするように変更することはできません。ただし、検索が無効なクラスターのスナップショットから検索対応クラスターを作成できます。

パラメトリック制限

次の表は、プレビューにおけるさまざまなベクトル検索項目の制限を示しています。

Item	最大値
ベクトルの次元の数	32768
作成できるインデックスの数	10
インデックス内のフィールドの数	50
FT.SEARCH および FT.AGGREGATE TIMEOUT 句 (ミリ秒)	10000
FT.AGGREGATE コマンドのパイプラインステージの数	32
FT.AGGREGATE LOAD 句のフィールドの数	1024
FT.AGGREGATE GROUPBY 句のフィールドの数	16
FT.AGGREGATE SORTBY 句のフィールドの数	16
FT.AGGREGATE PARAM 句のパラメータの数	32
HNSW M パラメータ	512
HNSW EF_CONSTRUCTION パラメータ	4096
HNSW EF_RUNTIME パラメータ	4096

[Scaling limits] (スケーリング履歴)

現在、MemoryDB のベクトル検索は単一のシャードに制限されており、水平方向のスケーリングはサポートされていません。ベクトル検索は、垂直スケーリングとレプリカスケーリングをサポートしています。

オペレーションの制限

インデックスの永続化とバックフィル

ベクトル検索機能は、インデックスの定義とインデックスの内容を保持します。つまり、ノードの起動または再起動を引き起こすオペレーションリクエストまたはイベント中に、インデックス定義と内容が最新のスナップショットから復元され、保留中のトランザクションがジャーナルから再生されます。これを開始するには、ユーザーアクションは必要ありません。再構築は、データが復元されるとすぐにバックフィルオペレーションとして実行されます。これは、定義されたインデックスごとに FT.CREATE コマンドを自動的に実行するシステムと機能的に同等です。データが復元されると、アプリケーションのオペレーションのためにすぐにノードを使用できるようになりますが、これはインデックスバックフィルが完了する前である可能性が高いことに留意してください。これは、アプリケーションが再びバックフィルを認識できるようになることを意味します。例えば、バックフィルインデックスを使用した検索コマンドは拒否される可能性があります。バックフィルの詳細については、「ベクトル検索の概要」を参照してください。

インデックスバックフィルの完了は、プライマリとレプリカの間で同期されません。この同期の欠如は予期せずアプリケーションにとって認識可能になる可能性があるため、検索オペレーションを開始する前に、アプリケーションでプライマリとすべてのレプリカでバックフィルが完了したことを検証することをお勧めします。

スナップショットのインポート/エクスポートとライブ移行

RDB ファイル内に検索インデックスが存在する場合、そのデータの互換性のある転送可能性が制限されます。MemoryDB ベクトル検索機能で定義されるベクトルインデックスの形式は、別の MemoryDB ベクトル対応クラスターでのみ理解されます。また、プレビュークラスターの RDB ファイルは、MemoryDB クラスターの GA バージョンによってインポートできます。これにより、RDB ファイルのロード時にインデックスコンテンツが再構築されます。

ただし、インデックスを含まない RDB ファイルについては、このような制限はありません。そのため、エクスポート前にインデックスを削除することで、プレビュークラスター内のデータを、非プレビュークラスターにエクスポートできます。

メモリ消費

メモリ消費量は、ベクトルの数、ディメンションの数、M 値、およびベクトルに関連付けられたメタデータやインスタンス内に保存された他のデータなどのベクトル以外のデータの量に基づいています。

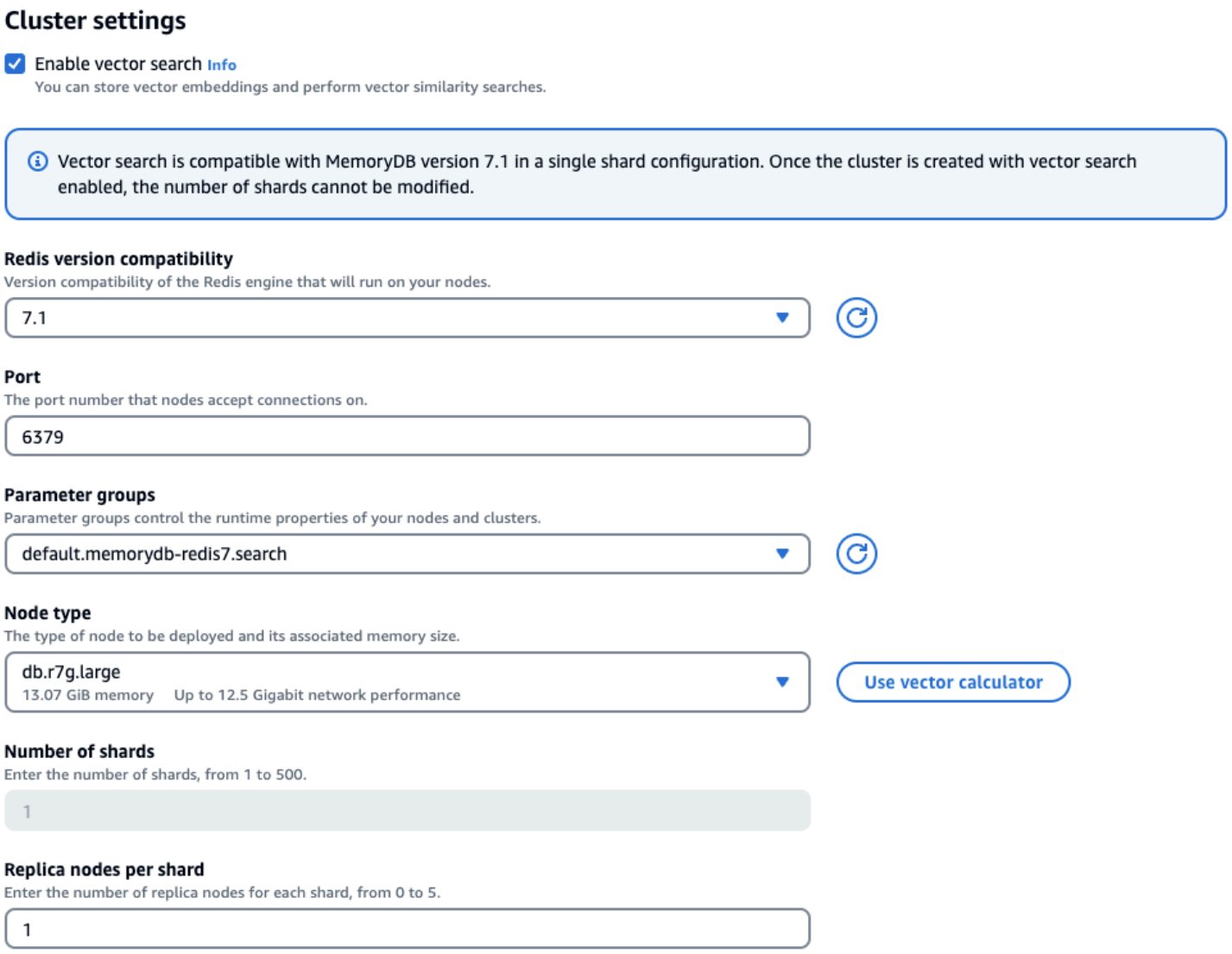
必要な合計メモリは、実際のベクトルデータに必要なスペースとベクトルインデックスに必要なスペースの組み合わせです。ベクトルデータに必要なスペースは、最適なメモリ割り当てのために、HASH または JSON データ構造内にベクトルを保存するために必要な実際の容量と、最も近いメモリスラブへのオーバーヘッドを測定して計算されます。各ベクトルインデックスは、これらのデータ構造に保存されているベクトルデータへの参照を使用し、効率的なメモリ最適化を使用して、インデックス内のベクトルデータの重複コピーを削除します。

ベクトルの数は、データをベクトルとして表す方法によって異なります。例えば、1 つのドキュメントを複数のチャンクに表現することを選択できます。各チャンクはベクトルを表します。または、ドキュメント全体を単一のベクトルとして表現することもできます。

ベクトルのディメンションの数は、選択した埋め込みモデルによって異なります。例えば、AWSTitan 埋め込みモデルを使用する場合、ディメンションの数は 1536 になります。

M パラメータは、インデックス構築中に新しい要素ごとに作成された双方向リンクの数を表します。MemoryDB のデフォルト値は 16 ですが、これを上書きできます。高い M パラメータは、高いディメンションおよび/または高いリコール要件に対してより適しており、低い M パラメータは、低いディメンションおよび/または低いリコール要件に対してより適しています。M 値は、インデックスが大きくなるにつれてメモリの消費を増加させ、メモリの消費量が増加します。

コンソールエクスペリエンス内で、MemoryDB は、[クラスター設定でベクトル検索を有効にする] をチェックした後、ベクトルワークロードの特性に基づいて適切なインスタンスタイプを簡単に選択する方法を提供します。

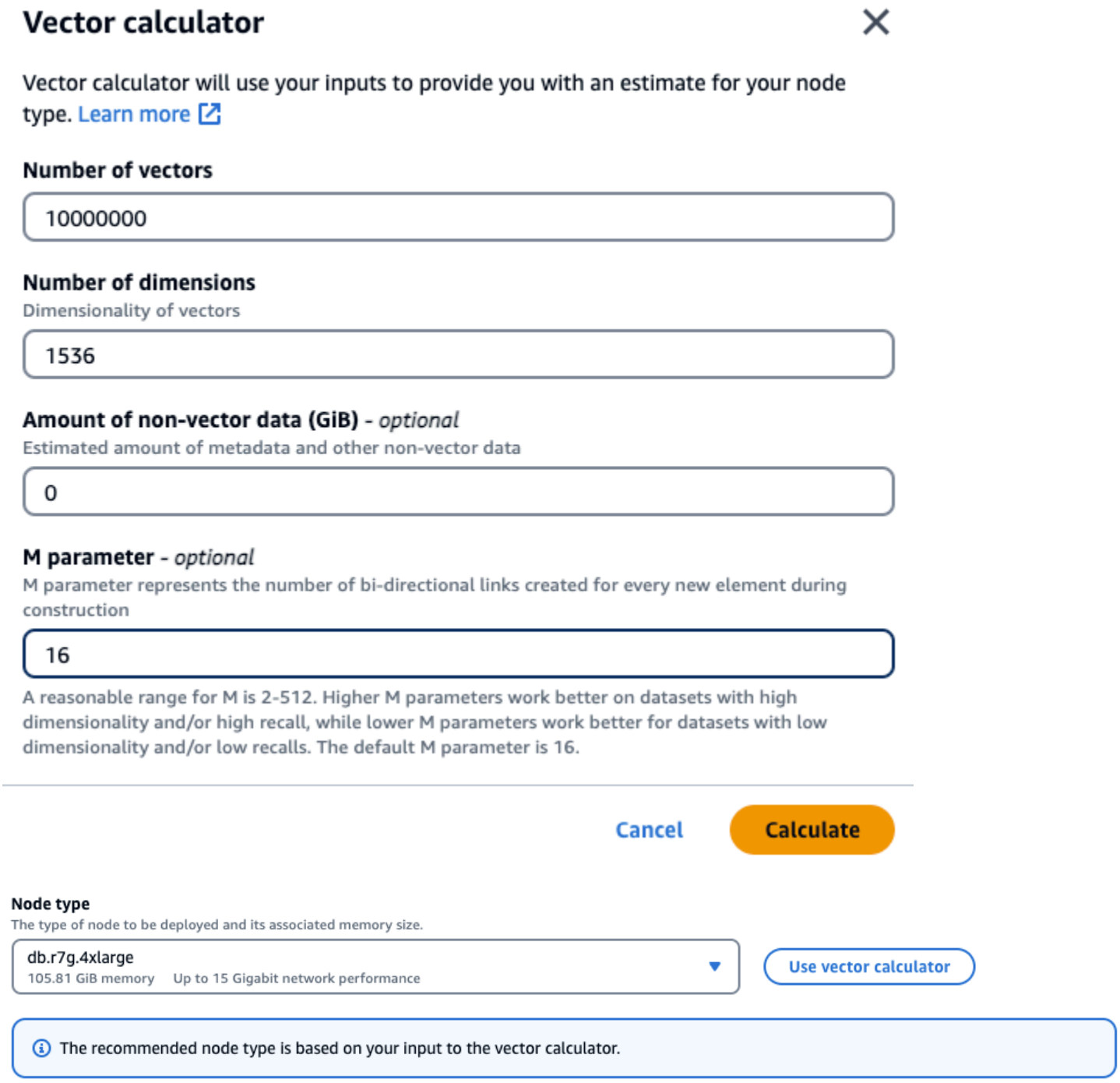


サンプルワークロード

あるお客様が、内部財務ドキュメントに基づいたセマンティック検索エンジンの構築を希望しています。現在、1536 ディメンションの Titan 埋め込みモデルを使用して、ドキュメントごとに 10 ベクトルに分割され、非ベクトルデータが含まれない 100 万件の財務ドキュメントが保管しています。お客様は、M パラメータとしてデフォルト 16 を使用することにしました。

- ベクトル: 100 万 * 10 チャンク = 1,000 万ベクトル
- ディメンション: 1536
- 非ベクトルデータ (GB): 0 GB
- M パラメータ: 16

このデータを使用して、お客様はコンソール内のベクトル計算ツールの使用ボタンをクリックし、パラメータに基づいて推奨されるインスタンスタイプを取得できます。



この例では、ベクトル計算ツールは、指定されたパラメータに基づいてベクトルを保存するために必要なメモリを保持できる最小の MemoryDB r7g ノードタイプを探します。これは近似値であり、インスタンスタイプをテストして要件に適合していることを確認する必要があることに注意してください。

上記の計算方法とサンプルワークロードのパラメータに基づくと、このベクトルデータの保存には 104.9 GB と単一のインデックスが必要です。この場合、使用可能なメモリが 105.81 GB であるため、db.r7g.4xlarge インスタンスタイプが推奨されます。次に小さいノードタイプは小さすぎて、ベクトルワークロードを保持できません。

各ベクトルインデックスは、保存されたベクトルデータへの参照を使用し、ベクトルインデックスにベクトルデータの追加のコピーを作成しないため、インデックスの消費スペースも比較的少なくなります。これは、複数のインデックスを作成する場合や、ベクトルデータの一部が削除され、HNSW グラフを再構築する場合にも非常に役立ちます。これにより、高品質のベクトル検索結果に最適なノード接続を作成できます。

バックフィル中のメモリ不足

Valkey および Redis OSS の書き込みオペレーションと同様に、インデックスバックフィルにはメモリ不足の制限があります。バックフィルの進行中にエンジンメモリがいっぱいになると、すべてのバックフィルが一時停止します。メモリが使用可能になると、バックフィルプロセスが再開されます。メモリ不足によりバックフィルが一時停止されたときに、削除してインデックスを作成することも可能です。

トランザクション

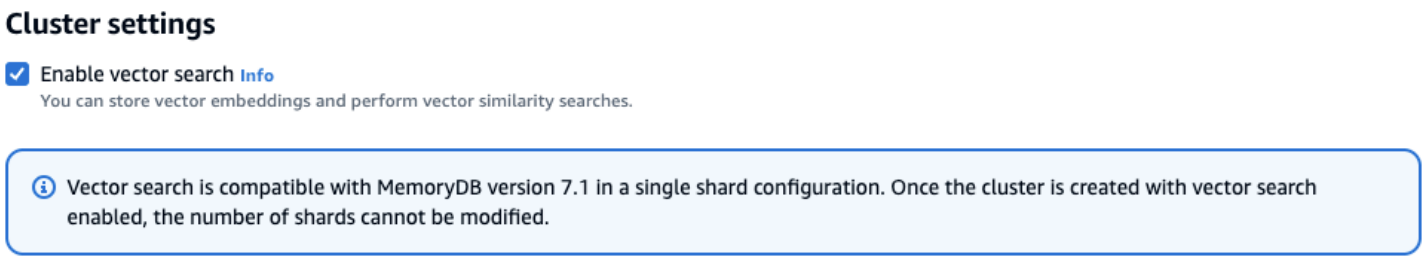
コマンド FT.CREATE、FT.DROPINDEX、FT.ALIASADD、FT.ALIASDEL、および FT.ALIASUPDATE は、トランザクションコンテキストでは実行できません。すなわち、MULTI/EXEC ブロック内、または LUA もしくは FUNCTION スクリプト内では実行できません。

ベクトル検索が有効になっているクラスターを作成する

AWS マネジメントコンソール または AWS Command Line Interface を使用して、ベクトル検索が有効になっているクラスターを作成できます。アプローチによっては、ベクトル検索を有効にするための考慮事項を有効にする必要があります。

の使用AWS マネジメントコンソール

コンソール内でベクトル検索が有効なクラスターを作成するには、クラスターの設定でベクトル検索を有効にする必要があります。ベクトル検索は、MemoryDB バージョン 7.1 および単一シャード設定で使用できます。



AWS マネジメントコンソール でのベクトル検索の使用の詳細については、「クラスターの作成 (コンソール)」を参照してください。

の使用AWS Command Line Interface

ベクトル検索が有効な MemoryDB クラスターを作成するには、MemoryDB create-cluster コマンドを使用してイミュータブルなパラメータグループ `default.memorydb-redis7.search` を渡し、ベクトル検索機能を有効にします。

```
aws memorydb create-cluster \
  --cluster-name <value> \
  --node-type <value> \
  --engine redis \
  --engine-version 7.1 \
  --num-shards 1 \
  --acl-name <value> \
  --parameter-group-name default.memorydb-redis7.search
```

必要に応じて、次の例に示すように、ベクトル検索を有効にする新しいパラメータグループを作成することもできます。パラメータグループの詳細については、こちらを参照してください。

```
aws memorydb create-parameter-group \
  --parameter-group-name my-search-parameter-group \
  --family memorydb_redis7
```

次に、新しく作成したパラメータグループで、パラメータ search-enabled を yes に更新します。

```
aws memorydb update-parameter-group \
  --parameter-group-name my-search-parameter-group \
  --parameter-name-values "ParameterName=search-enabled,ParameterValue=yes"
```

デフォルトのパラメータグループの代わりにこのカスタムパラメータグループを使用して、MemoryDB クラスターでベクトル検索を有効にできるようになりました。

ベクトル検索コマンド

ベクトル検索でサポートされているコマンドのリストを次に示します。

トピック

- FT.CREATE
- FT.SEARCH
- FT.AGGREGATE
- FT.DROPINDEX
- FT.INFO
- FT._LIST
- FT.ALIASADD
- FT.ALIASDEL
- FT.ALIASUPDATE
- FT._ALIASLIST
- FT.PROFILE
- FT.EXPLAIN
- FT.EXPLAINCLI

FT.CREATE

インデックスを作成し、そのインデックスのバックフィルを開始します。詳細については、「ベクトル検索の概要」でインデックス構築の詳細をご覧ください。

構文

```
FT.CREATE <index-name>
ON HASH | JSON
[PREFIX <count> <prefix1> [<prefix2>...]]
SCHEMA
(<field-identifier> [AS <alias>]
  NUMERIC
```

```
| TAG [SEPARATOR <sep>] [CASESENSITIVE]
| TEXT
| VECTOR [HNSW|FLAT] <attr_count> [<attribute_name> <attribute_value>])

)+
```

Schema

- フィールド識別子:
  - ハッシュキーの場合、フィールド識別子はフィールド名です。
  - JSON キーの場合、フィールド識別子は JSON パスです。

  詳細については、「インデックスフィールドの型」を参照してください。

- フィールドの型:
  - タグ: 詳細については、「タグ」を参照してください。
  - NUMERIC: フィールドには数値が含まれます。
  - TEXT: フィールドにはデータの BLOB が含まれます。
  - VECTOR: ベクトル検索をサポートするベクトルフィールド。
    - アルゴリズム – HNSW (Hierarchical Navigable Small World) または FLAT (総当たり) を使用できます。
    - `attr_count` – アルゴリズム設定として渡される属性の数。これには名前と値の両方が含まれます。
    - `{attribute_name} {attribute_value}` – インデックス設定を定義するアルゴリズム固有の key/value ペア。

      FLAT アルゴリズムの場合、属性は次のとおりです:

      必須:

      - DIM – ベクトルの次元の数。
      - DISTANCE_METRIC – [L2 | IP | COSINE] のいずれかを使用できます。
      - TYPE – ベクトルの型。サポートされている唯一の型は `FLOAT32` です。

      オプション:

      - INITIAL_CAP – インデックスのメモリ割り当てサイズに影響するインデックスの初期ベクトル容量。

HNSW アルゴリズムの場合、属性は次のとおりです:

必須:

- TYPE – ベクトルの型。サポートされている唯一の型は FLOAT32 です。
- DIM – ベクトルの次元。正の整数として指定します。最大: 32,768
- DISTANCE_METRIC – [L2 | IP | COSINE] のいずれかを使用できます。

オプション:

- INITIAL_CAP – インデックスのメモリ割り当てサイズに影響するインデックスの初期ベクトル容量。デフォルトは 1024 です。
- M – 各レイヤーのグラフ内の各ノードに許可される発信エッジの最大数。レイヤー 0 では、発信エッジの最大数は 2M です。デフォルトは 16、最大値は 512 です。
- EF_CONSTRUCTION – インデックスの構築中に検査されるベクトルの数を制御します。このパラメータの値を大きくすると、インデックスの作成時間が長くなりますが、再現率が向上します。デフォルト値は 200 です。最大値は 4096 です。
- EF_RUNTIME – クエリオペレーション中に検査されるベクトルの数を制御します。このパラメータの値を大きくすると、クエリ時間が長くなりますが、再現が改善されます。このパラメータの値はクエリごとに上書きできます。デフォルト値は 10 です。最大値は 4096 です。

戻る

シンプルな文字列の OK メッセージまたはエラー応答を返します。

例

> **Note**
>
> 次の例では、データを Valkey または Redis OSS に送信する前に、そのデータの引用符の削除やエスケープの削除など、valkey-cli にネイティブな引数を使用します。他のプログラミング言語クライアント (Python、Ruby、C# など) を使用するには、それらの環境の文字列およびバイナリデータの処理規則に従います。サポートされているクライアントの詳細については、「AWS で構築するためのツール」を参照してください。

Example 1: インデックスを作成する

サイズ 2 のベクトルのインデックスを作成する

```
FT.CREATE hash_idx1 ON HASH PREFIX 1 hash: SCHEMA vec AS VEC VECTOR HNSW 6 DIM 2 TYPE
 FLOAT32 DISTANCE_METRIC L2
OK
```

HNSW アルゴリズムを使用して 6 次元の JSON インデックスを作成します:

```
FT.CREATE json_idx1 ON JSON PREFIX 1 json: SCHEMA $.vec AS VEC VECTOR HNSW 6 DIM 6 TYPE
 FLOAT32 DISTANCE_METRIC L2
OK
```

Example 例 2: 一部のデータを入力する

次のコマンドは、redis-cli ターミナルプログラムの引数として実行できるようにフォーマットされています。プログラミング言語クライアント (Python、Ruby、C# など) を使用するデベロッパーは、文字列とバイナリデータを処理するための環境の処理ルールに従う必要があります。

ハッシュデータと JSON データの作成:

```
HSET hash:0 vec "\x00\x00\x00\x00\x00\x00\x00\x00"
HSET hash:1 vec "\x00\x00\x00\x00\x00\x00\x80\xbf"
JSON.SET json:0 . '{"vec":[1,2,3,4,5,6]}'
JSON.SET json:1 . '{"vec":[10,20,30,40,50,60]}'
JSON.SET json:2 . '{"vec":[1.1,1.2,1.3,1.4,1.5,1.6]}'
```

次の点に注意してください。

- ハッシュおよび JSON データのキーには、インデックス定義のプレフィックスが含まれています。
- ベクトルはインデックス定義の適切なパスにあります。
- ハッシュベクトルは 16 進データとして入力され、JSON データは数値として入力されます。
- ベクトルは適切な長さであり、2 次元のハッシュベクトルエントリには 2 つの浮動小数点相当の 16 進データが含まれており、6 次元の JSON ベクトルエントリには 6 つの数値が含まれています。

Example 例 3: インデックスを削除して再作成する

```
FT.DROPINDEX json_idx1
OK

FT.CREATE json_idx1 ON JSON PREFIX 1 json: SCHEMA $.vec AS VEC VECTOR FLAT 6 DIM 6 TYPE
 FLOAT32 DISTANCE_METRIC L2
OK
```

新しい JSON インデックスでは、HNSW アルゴリズムの代わりに FLAT アルゴリズムが使用されることに留意してください。また、既存の JSON データのインデックスが再作成されることにも留意してください。

```
FT.SEARCH json_idx1 "*=>[KNN 100 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
 DIALECT 2
1) (integer) 3
2) "json:2"
3) 1) "__VEC_score"
   2) "11.11"
   3) "$"
   4) "[{\"vec\":[1.1, 1.2, 1.3, 1.4, 1.5, 1.6]}]"
4) "json:0"
5) 1) "__VEC_score"
   2) "91"
   3) "$"
   4) "[{\"vec\":[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]}]"
6) "json:1"
7) 1) "__VEC_score"
   2) "9100"
   3) "$"
   4) "[{\"vec\":[10.0, 20.0, 30.0, 40.0, 50.0, 60.0]}]"
```

FT.SEARCH

インデックス内のキーを検索するために、指定されたクエリ式を使用します。見つかると、それらのキー内のインデックス付きフィールドのカウントおよび/または内容を返すことができます。詳細については、「ベクトル検索クエリ式」を参照してください。

これらの例で使用するデータを作成するには、FT.CREATE コマンドを参照してください。

構文

```
FT.SEARCH <index-name> <query>
[RETURN <token_count> (<field-identifier> [AS <alias>])+]
[TIMEOUT timeout]
[PARAMS <count> <name> <value> [<name> <value>]]
[LIMIT <offset> <count>]
[COUNT]
```

- RETURN: この句は、キーのどのフィールドが返されるかを指定します。各フィールドのオプションの AS 句は、結果内のフィールドの名前を上書きします。このインデックスのために宣言されているフィールドのみを指定できます。
- LIMIT: <offset> <count>: この句は、オフセットおよびカウントの値を満たすキーのみが返されるというページネーション機能を提供します。この句を省略すると、デフォルトで「LIMIT 0 10」になります。すなわち、最大 10 個のキーのみが返されます。
- PARAMS: key-value ペアの数の 2 倍。Param の key/value ペアはクエリ式内から参照できます。詳細については、「ベクトル検索クエリ式」を参照してください。
- COUNT: この句はキーの内容を返すことを抑制し、キーの数のみが返されます。これは「LIMIT 0 0」のエイリアスです。

戻る

配列またはエラー応答を返します。

- オペレーションが正常に完了すると、配列が返されます。最初の要素は、クエリに一致するキーの総数です。残りの要素は、キー名とフィールドリストのペアです。フィールドリストは、フィールド名と値のペアで構成される別の配列です。
- インデックスのバックフィルが進行中の場合、コマンドは直ちにエラー応答を返します。
- タイムアウトに達すると、コマンドはエラー応答を返します。

例: 検索を実行する

> **Note**
>
> 次の例では、データを Valkey または Redis OSS に送信する前に、そのデータの引用符の削除やエスケープの削除など、valkey-cli にネイティブな引数を使用します。他のプログラミング言語クライアント (Python、Ruby、C# など) を使用するには、それらの環境の文字列およ

びバイナリデータの処理規則に従います。サポートされているクライアントの詳細については、「AWS で構築するためのツール」を参照してください。

ハッシュ検索

```
FT.SEARCH hash_idx1 "*=>[KNN 2 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00" DIALECT 2
1) (integer) 2
2) "hash:0"
3) 1) "__VEC_score"
   2) "0"
   3) "vec"
   4) "\x00\x00\x00\x00\x00\x00\x00\x00"
4) "hash:1"
5) 1) "__VEC_score"
   2) "1"
   3) "vec"
   4) "\x00\x00\x00\x00\x00\x00\x80\xbf"
```

これにより、クエリベクトル (16 進数で入力) からの距離であるスコアによってソートされた 2 つの結果が生成されます。

JSON 検索

```
FT.SEARCH json_idx1 "*=>[KNN 2 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
 DIALECT 2
1) (integer) 2
2) "json:2"
3) 1) "__VEC_score"
   2) "11.11"
   3) "$"
   4) "[{\"vec\":[1.1, 1.2, 1.3, 1.4, 1.5, 1.6]}]"
4) "json:0"
5) 1) "__VEC_score"
   2) "91"
   3) "$"
   4) "[{\"vec\":[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]}]"
```

これにより、スコア順に並べ替えられた最も近い 2 つの結果が生成されます。JSON ベクトル値は浮動小数点数に変換され、クエリベクトルは依然としてベクトルデータであることに留意してくださ

い。KNN パラメータが 2 であるため、結果は 2 つだけであることにも留意してください。値が大きいほど、より多くの結果が返されます:

```
FT.SEARCH json_idx1 "*=>[KNN 100 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
 DIALECT 2
1) (integer) 3
2) "json:2"
3) 1) "__VEC_score"
   2) "11.11"
   3) "$"
   4) "[{\"vec\":[1.1, 1.2, 1.3, 1.4, 1.5, 1.6]}]"
4) "json:0"
5) 1) "__VEC_score"
   2) "91"
   3) "$"
   4) "[{\"vec\":[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]}]"
6) "json:1"
7) 1) "__VEC_score"
   2) "9100"
   3) "$"
   4) "[{\"vec\":[10.0, 20.0, 30.0, 40.0, 50.0, 60.0]}]"
```

FT.AGGREGATE

FT.SEARCH コマンドのスーパーセット。クエリ式によって選択されたキーの実質的な追加処理を許可します。

構文

```
FT.AGGREGATE index query
  [LOAD * | [count field [field ...]]]
  [TIMEOUT timeout]
  [PARAMS count name value [name value ...]]
  [FILTER expression]
  [LIMIT offset num]
  [GROUPBY count property [property ...] [REDUCE function count arg [arg ...] [AS name]
 [REDUCE function count arg [arg ...] [AS name] ...]] ...]]
  [SORTBY count [ property ASC | DESC [property ASC | DESC ...]] [MAX num]]
  [APPLY expression AS name]
```

- FILTER、LIMIT、GROUPBY、SORTBY、および APPLY 句は、任意の順序で複数回繰り返すことができ、自由に組み合わせることができます。これらは指定された順序で適用され、1 つの句の出力を次の句の入力に供給します。
- 上記の構文では、「プロパティ」は、このインデックスの FT.CREATE コマンドで宣言されたフィールド、または前の APPLY 句もしくは REDUCE 関数の出力のいずれかです。
- LOAD 句は、インデックスで宣言されたフィールドのロードに制限されます。「LOAD *」は、インデックスで宣言されたすべてのフィールドをロードします。
- 次のリデューサー関数がサポートされています: COUNT、COUNT_DISTINCTISH、SUM、MIN、MAX、AVG、STDDEV、QUANTILE、TOLIST、FIRST_および RANDOM_SAMPLE。詳細については、「集計」を参照してください。
- LIMIT <offset> <count>: <offset> で始まり <count> まで続くレコードを保持し、他のすべてのレコードは破棄されます。
- PARAMS: key-value ペアの数の 2 倍。Param の key/value ペアはクエリ式内から参照できます。

戻る

配列またはエラー応答を返します。

- オペレーションが正常に完了すると、配列が返されます。最初の要素は特定の意味を持たない整数です (無視する必要があります)。残りの要素は、最後のステージによって出力された結果です。各要素はフィールド名と値のペアの配列です。
- インデックスのバックフィルが進行中の場合、コマンドは直ちにエラー応答を返します。
- タイムアウトに達すると、コマンドはエラー応答を返します。

FT.DROPINDEX

インデックスをドロップします。インデックス定義と関連付けられたコンテンツが削除されます。キーは影響を受けません。

構文

```
FT.DROPINDEX <index-name>
```

戻る

シンプルな文字列の OK メッセージまたはエラー応答を返します。

FT.INFO

構文

```
FT.INFO <index-name>
```

FT.INFO ページからの出力は、次の表に示す key-value ペアの配列です:

キー	値のタイプ	説明
index_name	文字列	インデックスの名前
creation_timestamp	integer	作成時の Unix スタイルのタイムスタンプ
key_type	文字列	HASH または JSON
key_prefixes	文字列の配列	このインデックスのキープレフィックス
fields	フィールド情報の配列	このインデックスのフィールド
space_usage	integer	このインデックスによって使用されるメモリバイト
fullext_space_usage	integer	非ベクトルフィールドによって使用されるメモリバイト
vector_space_usage	integer	ベクトルフィールドによって使用されるメモリバイト
num_docs	integer	現在インデックスに含まれているキーの数
num_indexed_vectors	integer	現在インデックスに含まれているベクトルの数
current_lag	integer	最近の取り込み遅延 (ミリ秒)

キー	値のタイプ	説明
backfill_status	文字列	次のいずれか: Completed、InProgres、Paused、または Failed

次の表では、各フィールドの情報について説明します:

キー	値のタイプ	説明
識別子	文字列	フィールドの名前
field_name	文字列	ハッシュメンバー名または JSON パス
type	文字列	次のいずれか: Numeric、Tag、Text、または Vector
option	文字列	ignore

フィールドの型が Vector である場合、アルゴリズムに応じて追加情報が存在します。

HNSW アルゴリズムの場合:

キー	値のタイプ	説明
アルゴリズム	文字列	HNSW
data_type	文字列	FLOAT32
distance_metric	文字列	次のいずれか: L2、IP、または Cosine
initial_capacity	integer	ベクトルフィールドインデックスの初期サイズ
current_capacity	integer	ベクトルフィールドインデックスの現在のサイズ

キー	値のタイプ	説明
maximum_edges	integer	作成時の M パラメータ
ef_construction	integer	作成時の EF_CONSTRUCTION パラメータ
ef_runtime	integer	作成時の EF_RUNTIME パラメータ

FLAT アルゴリズムの場合:

キー	値のタイプ	説明
アルゴリズム	文字列	FLAT
data_type	文字列	FLOAT32
distance_metric	文字列	次のいずれか: L2、IP、または Cosine
initial_capacity	integer	ベクトルフィールドインデックスの初期サイズ
current_capacity	integer	ベクトルフィールドインデックスの現在のサイズ

FT._LIST

すべてのインデックスを一覧表示します。

構文

```
FT._LIST
```

戻る

インデックス名の配列を返します

FT.ALIASADD

インデックスのエイリアスを追加します。新しいエイリアスは、インデックス名が必要なあらゆる場所で使用できます。

構文

```
FT.ALIASADD <alias> <index-name>
```

戻る

シンプルな文字列の OK メッセージまたはエラー応答を返します。

FT.ALIASDEL

インデックスの既存のエイリアスを削除します。

構文

```
FT.ALIASDEL <alias>
```

戻る

シンプルな文字列の OK メッセージまたはエラー応答を返します。

FT.ALIASUPDATE

別の物理インデックスをポイントするように既存のエイリアスを更新します。このコマンドは、エイリアスへの今後の参照にのみ影響します。現在進行中のオペレーション (FT.SEARCH、FT.AGGREGATE) は、このコマンドの影響を受けません。

構文

```
FT.ALIASUPDATE <alias> <index>
```

戻る

シンプルな文字列の OK メッセージまたはエラー応答を返します。

FT._ALIASLIST

インデックスのエイリアスを一覧表示します。

構文

```
FT._ALIASLIST
```

戻る

現在のエイリアスの数のサイズの配列を返します。配列の各要素は、エイリアスとインデックスのペアです。

FT.PROFILE

クエリを実行し、そのクエリに関するプロファイル情報を返します。

構文

```
FT.PROFILE

<index>
SEARCH | AGGREGATE
[LIMITED]
QUERY <query ....>
```

戻る

2 つの要素の配列。1 つ目の要素は、プロファイリングされた FT.SEARCH または FT.AGGREGATE コマンドの結果です。2 つ目の要素は、パフォーマンスおよびプロファイリング情報の配列です。

FT.EXPLAIN

クエリを解析し、そのクエリがどのように解析されたかに関する情報を返します。

構文

```
FT.EXPLAIN <index> <query>
```

戻る

解析結果を含む文字列。

FT.EXPLAINCLI

結果が、redis-cli でより便利な別の形式で表示されることを除いて、FT.EXPLAIN コマンドと同じです。

構文

```
FT.EXPLAINCLI <index> <query>
```

戻る

解析結果を含む文字列。

MemoryDB マルチリージョン

MemoryDB マルチリージョンは、フルマネージド型のアクティブ/アクティブマルチリージョンデータベースです。これを使用することで、最大 99.999% の可用性を持ち、読み取りレイテンシーがマイクロ秒で書き込みレイテンシーが 1 桁ミリ秒のマルチリージョンアプリケーションを構築できます。ユーザーは、可用性と、リージョン劣化からの回復力の両方を向上させることができるだけでなく、マルチリージョンアプリケーションに対するローカルな読み取りと書き込みのレイテンシーが低いというメリットが得られます。

MemoryDB マルチリージョンを使用すると、可用性の高いマルチリージョンアプリケーションを構築して回復力を高めることができます。MemoryDB マルチリージョンはアクティブ/アクティブレプリケーションを提供します。そのため、読み取りと書き込みを、顧客に最も近いリージョンからマイクロ秒の読み取りレイテンシーと 1 桁ミリ秒の書き込みレイテンシーでローカルに処理できます。MemoryDB マルチリージョンはリージョン間でデータを非同期的にレプリケートし、データは通常 1 秒以内に伝播されます。更新の競合を自動的に解決し、データ相違の問題を修正します。これにより、ユーザーはアプリケーションに集中できるようになります。

MemoryDB マルチリージョンは現在、米国東部 (バージニア北部およびオハイオ)、米国西部 (オレゴン、北カリフォルニア)、欧州 (アイルランド、フランクフルト、ロンドン)、アジアパシフィック (東京、シドニー、ムンバイ、ソウル、シンガポール) の各 AWS リージョンでサポートされています。

MemoryDB マルチリージョンは、AWS マネジメントコンソールから数回クリックするか、最新の AWS SDK、または AWS CLI を使用することで、簡単に利用し始めることができます。

トピック

- 前提条件と制限
- 仕組み
- 整合性と競合の解決
- コンソールでの MemoryDB マルチリージョンの使用
- CLI での MemoryDB マルチリージョンの使用
- MemoryDB マルチリージョンのモニタリング
- MemoryDB マルチリージョンでのスケーリング
- サポートされているコマンドとサポートされていないコマンド

前提条件と制限

MemoryDB マルチリージョンを利用し始める前に、以下の点に注意してください。

- MemoryDB マルチリージョンはユーザーが選択したリージョン間でデータをレプリケートする – ユーザーは、マルチリージョンクラスターを作成することで、選択したリージョン間でデータが移動することを理解し、これに同意したものとみなされます。

  マルチリージョングループからリージョンを削除すると、そのリージョンのリージョンクラスターも削除されます。
- リージョンの可用性 – MemoryDB マルチリージョンは現在、米国東部 (バージニア北部およびオハイオ)、米国西部 (オレゴン、北カリフォルニア)、欧州 (アイルランド、フランクフルト、ロンドン)、アジアパシフィック (東京、シドニー、ムンバイ、ソウル、シンガポール) の各 AWS リージョンでサポートされています。
- 動作と設定 – すべてのマルチリージョンのリージョンクラスターは、シャード数、インスタンスタイプ、Valkey エンジンのバージョン、TLS、およびパラメータグループ設定が同一になります。ユーザーは、リージョンクラスターごとに異なる IAM 認証、ACL、スナップショットウィンドウ、タグ、カスタマーマネージドキー (CMK)、およびメンテナンスウィンドウを選択できます。

  MemoryDB マルチリージョンでは、さまざまなリージョンのクラスターにさまざまな数のレプリカを作成できます。
- サポートされるノードタイプ – MemoryDB マルチリージョンは、XL サイズ以上の R7g ノードでサポートされます。

  MemoryDB マルチリージョンは、Valkey エンジンバージョン 7.3 以降をサポートしています。
- サポートされているデータ型 – MemoryDB マルチリージョンは現在、ほとんどの Redis OSS または Valkey データ型をサポートしており、今後さらに多くのデータ型をサポートする予定です。サポートされているデータ型には、文字列、ハッシュ、セット、ソートされたセットが含まれますが、これらのデータ型を操作するコマンドがすべてサポートされているわけではありません。
- リージョンの総数 – MemoryDB マルチリージョンでは、最大 5 つの AWS リージョン間で MemoryDB クラスターデータを自動的にレプリケートできます。
- サポートされているオプション – MemoryDB マルチリージョンでは、水平/垂直スケーリング、IAM 統合、ACL、自動およびオンデマンドのスナップショット作成、自動ソフトウェアパッチ適用、およびモニタリングをサポートされています。

- バックアップと復元 – スナップショットを作成してマルチリージョンのリージョンクラスターのデータをバックアップできます。スナップショットは、手動で作成することも、MemoryDB の自動スナップショットスケジューラを使用して、毎日、リージョンクラスターごとに個別に指定した時刻に新しいスナップショットを作成することもできます。
- 移行 – MemoryDB または Redis OSS/Valkey RDB 形式のバックアップを復元できます。バックアップからデータを移行するには、新しい MemoryDB マルチリージョンのリージョンクラスターを作成し、Amazon S3 からスナップショットの場所を指定します。バックアップが MemoryDB スナップショットである場合は、名前を指定することもできます。MemoryDB マルチリージョンは、スナップショットのデータを使用してリージョンクラスターを作成します。MemoryDB マルチリージョンでは、文字列、ハッシュ、セット、ソートされたセットの各データ型がサポートされています。そのため、移行できるのはこれらのサポート対象のデータ型のスナップショットデータのみです。バックアップファイルにサポートされていない Redis OSS データ型が含まれている場合、MemoryDB マルチリージョンはデフォルトで移行オペレーションに失敗します。
- リソース予約 – MemoryDB マルチリージョンは、リージョンの可用性を保護するように設計されています。一部のリソースは、ローカルの読み取りおよび書き込みリクエストをピアリージョンのワークロードとは独立して処理できるようにするために、各ノードで永続的に予約されています。これらのリソースは、リージョン分離イベントとその復旧中など、ピアリージョンでイベント (Regionisolation イベントとそれに伴う復旧など) が発生しているときにローカルの可用性を保護する役割も果たします。これにより、単一リージョン MemoryDB とは異なるパフォーマンス特性が得られます。MemoryDB マルチリージョンでは、水平スケーリングおよび垂直スケーリングによる利用可能なリソースの拡張がサポートされます。
- RPO/RTO SLA なし – MemoryDB マルチリージョンでは、RPO/RTO SLA が定められていません。MemoryDB マルチリージョンは、他の AWS リージョンから分離された AWS リージョンでの書き込みを受け入れ続けます。その結果、クロスレプリケーション遅延が無限に増加する可能性があります。当社は、ユーザーが「MultiRegionClusterReplicationLag」メトリクスを使用して分離を検出し、必要な RPO に応じてアプリケーショントラフィックを別のリージョンにリダイレクトすることを想定しています。
- 単一のエンドポイントや自動フェイルオーバーなし: – リージョンの停止が発生した場合、ユーザーは顧客のトラフィックを別のリージョンにある顧客のアプリケーションスタックに手動でリダイレクトする必要があります。ユーザーは、顧客が MemoryDB クラスターへのマルチリージョンアクセスを適切に設定していることを確認する必要があります。
- TTL サポートなし – MemoryDB マルチリージョンでは、TTL (有効期限) がサポートされません。
- データ階層化またはベクトル検索のサポートなし – MemoryDB マルチリージョンでは、ベクトル検索およびデータ階層化機能がサポートされません。

- MemoryDB マルチリージョンでは、読み取り/変更/書き込みコマンド (APPEND、RENAMENX など) がサポートされません。
- MemoryDB マルチリージョンでは、Redis OSS トランザクションのアトミック性と一貫性は保証されません。
- 認証モデル – MemoryDB マルチリージョン API アクションは、サポートされている任意のリージョンから呼び出すことができます。アクセス許可の範囲は、IAM ポリシーでマルチリージョンクラスターの ARN を指定することで制限できます。マルチリージョンクラスター ARN の形式は `arn:aws:memorydb::<account-id>:multiregioncluster/multi-region-cluster-name` です。ARN にリージョン情報はありません。
- スループットの制限 – MemoryDB マルチリージョンは、リージョン内のノードあたり最大 1.3 GB/秒の読み取りスループットと、シャードあたり最大 50 MB/秒のグローバル集計書き込みスループットをサポートできます。
- AWS ポリシー – AWS ReadOnlyAccess ポリシーは、AWS のサービスとリソースへの読み取り専用アクセスを提供しますが、1 つまたは複数のマルチリージョンクラスターに関する詳細を自動的に取得することはありません。1 つまたは複数のマルチリージョンクラスターに関する詳細を取得するには、AmazonMemoryDBReadOnlyAccess ポリシーを使用するか、IAM カスタマー管理ポリシーを作成します。
- リージョンクラスターの削除 – リージョンクラスターを削除した場合、関連するカスタマーマネージドキー (CMK) は、リージョンクラスターの削除が完了するまで有効のままになります。これにより、残りのリージョンクラスターは一貫性のある状態に収束できます。

仕組み

MemoryDB マルチリージョンの仕組みは次のとおりです。

- 概念:

  マルチリージョンクラスターは、1 つ以上のリージョンクラスターのコレクションであり、すべて 1 つの AWS アカウントによって所有されます。

  リージョンクラスターは、マルチリージョンクラスターの一部である AWS リージョン内の単一のクラスターです。各リージョンクラスターには、同じデータセットが保存されます。特定のマルチリージョンクラスターは、AWS リージョンごとに 1 つのリージョンクラスターのみを持つことができます。

マルチリージョンクラスターを作成した場合、そのクラスターは、MemoryDB が単一のユニットとして扱う複数のリージョンクラスター (リージョンごとに 1 つ) で構成されます。アプリケーションがリージョンクラスターにデータを書き込むと、MemoryDB はそのデータをマルチリージョンクラスター内の他のすべてのリージョンクラスターに自動的かつ非同期的にレプリケートします。マルチリージョンクラスターにリージョンクラスターを追加して、そのクラスターを追加のリージョンで使用できるようにすることができます。MemoryDB クラスターデータは、最大 5 つのリージョン間で自動的にレプリケートできます。

- 可用性と耐久性

  リージョンの分離や機能低下が万一発生した場合は、グローバル DNS を更新して、データベースの再設定なしでアプリケーションへのトラフィックを他の正常なリージョンのいずれかにリダイレクトできます。これにより、アプリケーションの高可用性を維持するプロセスが簡素化されます。MemoryDB は、すべてのリージョンからのすべての書き込みをマルチ AZ トランザクションログに永続的に保存し、リージョン内でデータ損失が発生しないようにします。MemoryDB マルチリージョンは、リージョンで承認されたものの、まだすべてのメンバークラスターにレプリケートされていない書き込みをすべて追跡します。リージョンの分離または機能低下が発生しても、ローカルの書き込みを受け入れ続けます。分離されたリージョンがマルチリージョンクラスターに再度接続されると、承認されたものの、他のリージョンにまたレプリケートされていない書き込みは、マルチリージョンクラスター内のすべてのリージョンにレプリケートされます。また、MemoryDB マルチリージョンは、CRDT メカニズムを使用し、保留中の書き込みと、停止中に他のリージョンで発生した可能性がある更新を自動的に調整します。

- MemoryDB マルチリージョンクラスターへの接続

  リージョンクラスターとの間でデータの書き込み/読み取りを行うには、サポートされている Redis OSS/Valkey クライアント (Valkey GLIDE を含む) のいずれかを使用してリージョンクラスターに接続します。各リージョンクラスターには、Redis OSS/Valkey クライアントが接続できるエンドポイントがあります。リージョンクラスターのエンドポイントは、AWS コンソール、CLI、または API を使用して取得できます。その後、このエンドポイントをアプリケーションで使用 (または設定) して、リージョンクラスターからデータを読み書きできます。

整合性と競合の解決

いずれかのリージョンクラスターでキーが更新された場合、その更新はマルチリージョンクラスター内の他のリージョンクラスターに通常は 1 秒未満で非同期的に伝播されます。リージョンが分離または機能低下した場合、MemoryDB マルチリージョンは、実行されたものの、まだすべてのメンバークラスターに伝播されていない書き込みをすべて追跡します。リージョンがオンラインに戻る

と、MemoryDB マルチリージョンは、そのリージョンから他のリージョンのメンバークラスターへの保留中の書き込みの伝播を再開します。また、他のメンバークラスターから現在オンラインに戻っているリージョンへの書き込みの伝播も再開します。リージョンの分離期間がどれほど長くても、それまでに成功した書き込みはすべて最終的に反映されます。

アプリケーションが異なるリージョンにある同一キーをほぼ同時に更新すると、競合が発生する可能性があります。MemoryDB マルチリージョンは、競合のないレプリケートデータ型 (CRDT) を使用して、競合する同時書き込みを調整します。CRDT は、調整なしで個別かつ同時に更新できるデータ構造です。つまり、書き込みと書き込みの競合は各レプリカで個別にマージされ、最終的に整合性が確保されます。

具体的には、MemoryDB は 2 つのレベルの Last Writer Wins (LWW) を使用して競合を解決します。文字列データ型の場合、LWW はキーレベルで競合を解決します。それ以外のデータ型の場合、LWW はサブキーレベルで競合を解決します。競合の解決は全面的に管理され、アプリケーションの可用性に影響を与えることなくバックグラウンドで処理されます。ハッシュデータ型の例を次に示します。

リージョン A は「HSET K F1 V1」をタイムスタンプ T1 で実行します。リージョン B は「HSET K F2 V2」をタイムスタンプ T2 で実行します。レプリケーション後、リージョン A とリージョン B のどちらにも両方のフィールドを持つキー K があります。異なるリージョンが同じコレクション内の異なるサブキーを同時に更新している場合、MemoryDB は、ハッシュデータ型の場合はサブキーレベルで競合を解決するため、2 つの更新は互いに競合しません。したがって、最終データには両方の更新の効果が反映されます。

時間	リージョン A	リージョン B
T1	HSET K F1 V1	
T2		HSET K F2 V2
T3	sync	sync
T4	K: {F1:V1, F2:V2}	K: {F1:V1, F2:V2}

CRDT と例

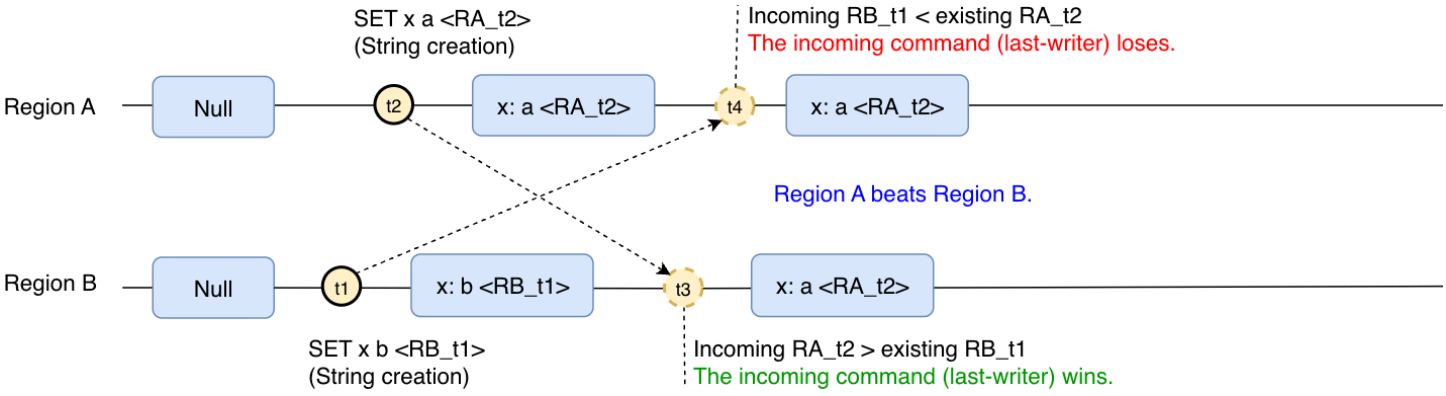
MemoryDB マルチリージョンは、競合のないレプリケートデータ型 (CRDT) を実装して、複数のリージョンから実行された同時書き込みの競合を解決します。CRDT により、複数の異なるリージョンが同じオペレーションセットを最終的に受信した後、それぞれのリージョンが受信順序に関係なく最終的な整合性を個別に達成することが可能になります。

1 つのキーが複数のリージョンで同時に更新された場合は、データ整合性を達成するために書き込みと書き込みの競合を解決する必要があります。MemoryDB マルチリージョンは、有効なオペレーションを決定するために Last Writer Wins (最後の書き込みが有効) (LWW) 戦略を使用します。この戦略では、「後に実行された」オペレーションの効果のみが最終的に観察されます。オペレーション op1 がオペレーション op2 の「前に実行された」と言えるのは、op2 が実行されたときに op1 が当初実行されたリージョンで op1 の効果が既に適用されていた場合です。

コレクション (ハッシュ、セット、およびソートされたセット) の場合、MemoryDB マルチリージョンは要素レベルで競合を解決します。これにより、MemoryDB マルチリージョンは、LWW を使用して各要素に対する書き込み/書き込みの競合を解決できます。例えば、複数のリージョンから異なる要素を同じコレクションに同時に追加すると、コレクションにはすべての要素が格納されます。

同時実行: Last Writer Wins

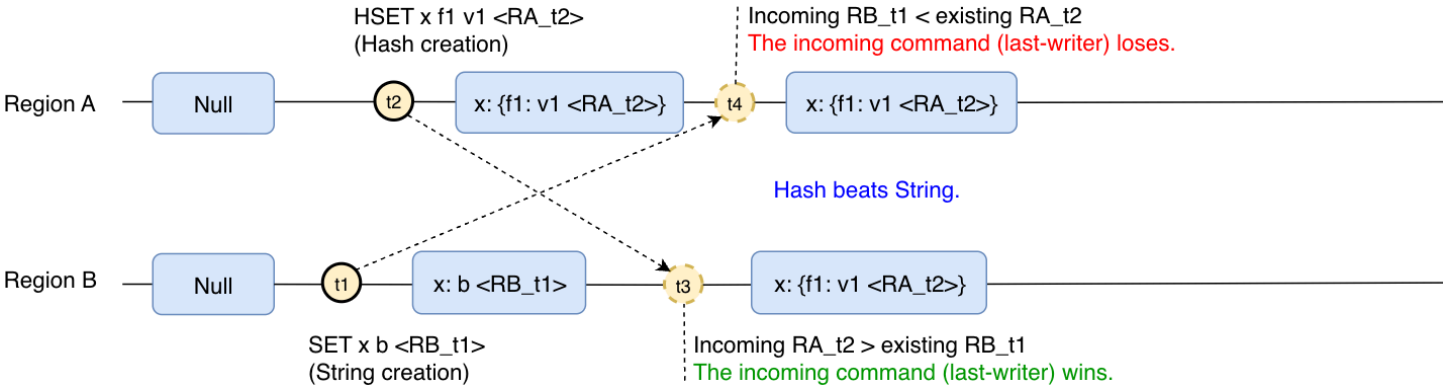
MemoryDB マルチリージョンでは、キーが同時に作成された場合、任意のリージョンで最後に実行されたオペレーションによってキーの結果が決まります。例:



リージョン B でキー x が値「b」で作成されましたが、その後、リージョン A で同じキーが値「a」で作成されました。リージョン A でのオペレーションが最後に実行されたオペレーションであるため、最終的にキーは値「a」に収束します。

競合するデータ型での同時実行: 最後の書き込みが有効

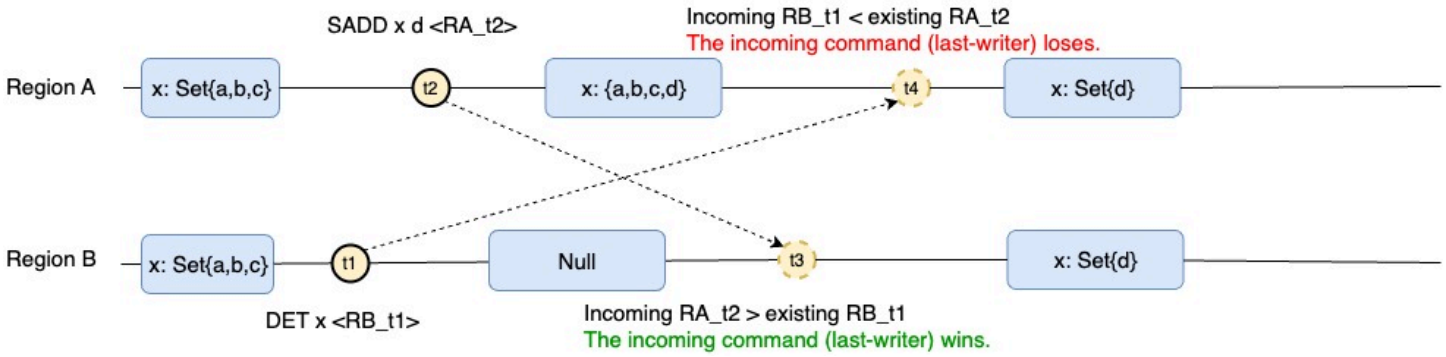
前の例では、両方のリージョンでキーが同じ型で作成されました。キーが異なるデータ型で作成された場合も、同様の動作が観察されます。



リージョン B でキー x が文字列として値「b」で作成されました。しかし、その後、そのオペレーションがリージョン A にレプリケートされる前に、リージョン A で同じキーがハッシュとして作成されます。リージョン A でのオペレーションが最後に実行されたオペレーションであったため、最終的にキーはリージョン A で作成されたハッシュに収束します。

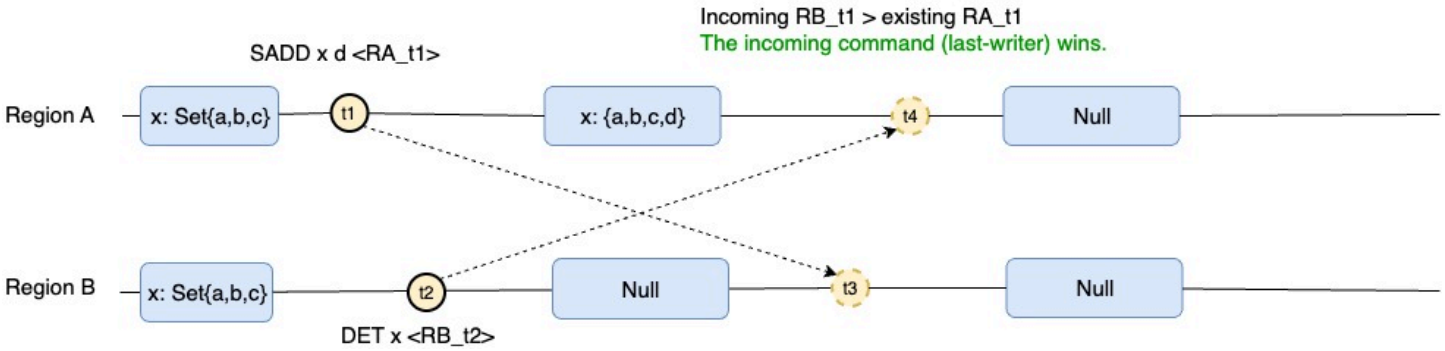
同時作成/削除: 最後の書き込みが有効

削除と「作成」(つまり、値の置換/追加) が同時に実行されるシナリオでは、最後に実行されたオペレーションが有効になります。最終結果は、削除オペレーションの順序によって決まります。削除が前に実行された場合:



リージョン B でセット型のキー x が削除されました。その後、リージョン A でそのキーに新しいメンバーが追加されました。リージョン A でのオペレーションが最後に実行されたオペレーションであるため、最終的にキーはリージョン A で追加された唯一の要素を持つセットに収束します。

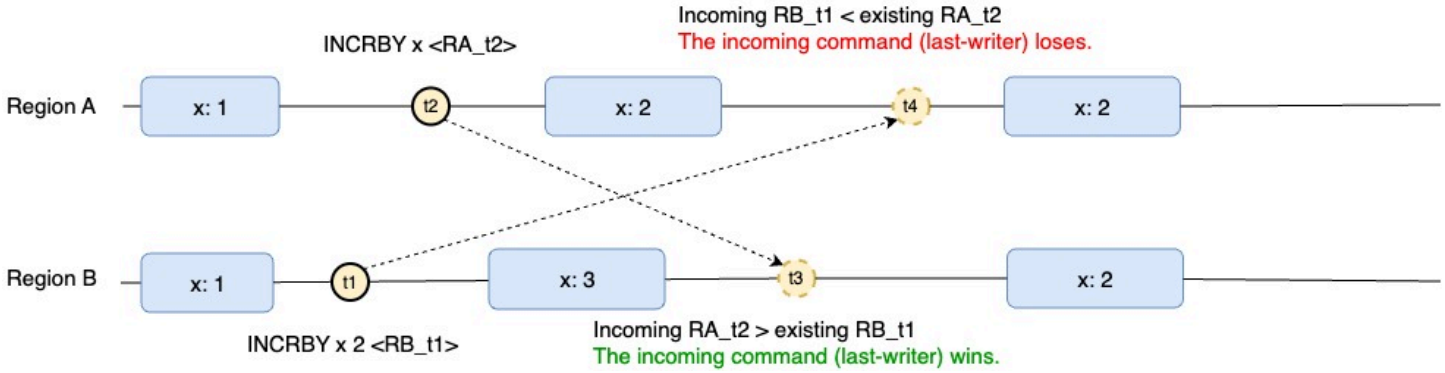
削除が後に実行された場合:



リージョン A でセット型のキー x に新しいメンバーが追加されました。その後、リージョン B でキーが削除されました。リージョン B でのオペレーションが最後に実行されたオペレーションであるため、最終的にキーは削除された状態に収束します。

カウンター、同時オペレーション: 最後の書き込みが有効になる完全値レプリケーション

MemoryDB マルチリージョンのカウンターは、完全値レプリケーションを実行し、最後の書き込み戦略を適用することで、非カウンタータイプと同様に動作します。同時オペレーションは結合されませんが、代わりに最後のオペレーションが有効になります。例:

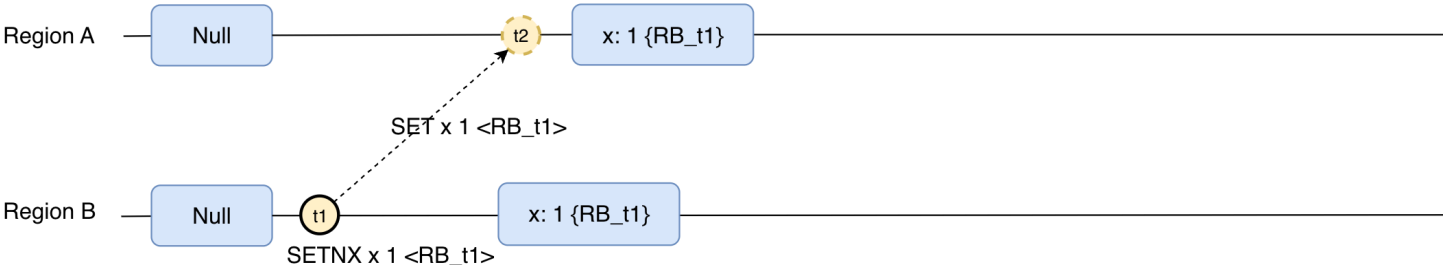


このシナリオでは、キー x の開始値は 1 です。その後、リージョン B がカウンター x を 2 増やし、その直後にリージョン A がカウンターを 1 増やしました。リージョン A が最後に実行されたオペレーションで、そのオペレーションが値を 1 増やすものであるため、キー x は最終的に値 2 に収束します。

非決定的コマンドは決定的としてレプリケートされる

MemoryDB マルチリージョンでは、異なるリージョン間で値の整合性を保証するため、非決定的コマンドは決定的としてレプリケートされます。非決定的コマンドは外部要因に依存するコマンドであり、SETNX などがこれに該当します。SETNX はキーが存在するかどうかに依存し、キーはリモートリージョンには存在するものの、コマンドを受信するローカルリージョンには存在しない可能性が

あります。このため、それ以外の場合には、非決定的コマンドは完全値レプリケーションとしてレプリケートされます。文字列の場合は、SET コマンドとしてレプリケートされます。



要約すると、文字列型に対するオペレーションはすべて SET または DEL としてレプリケートされ、ハッシュ型に対するオペレーションはすべて HSET または HDEL としてレプリケートされます。また、セット型に対するオペレーションはすべて SADD または SREM としてレプリケートされ、ソートされたセットに対するオペレーションはすべて ZADD または ZREM としてレプリケートされます。

コンソールでの MemoryDB マルチリージョンの使用

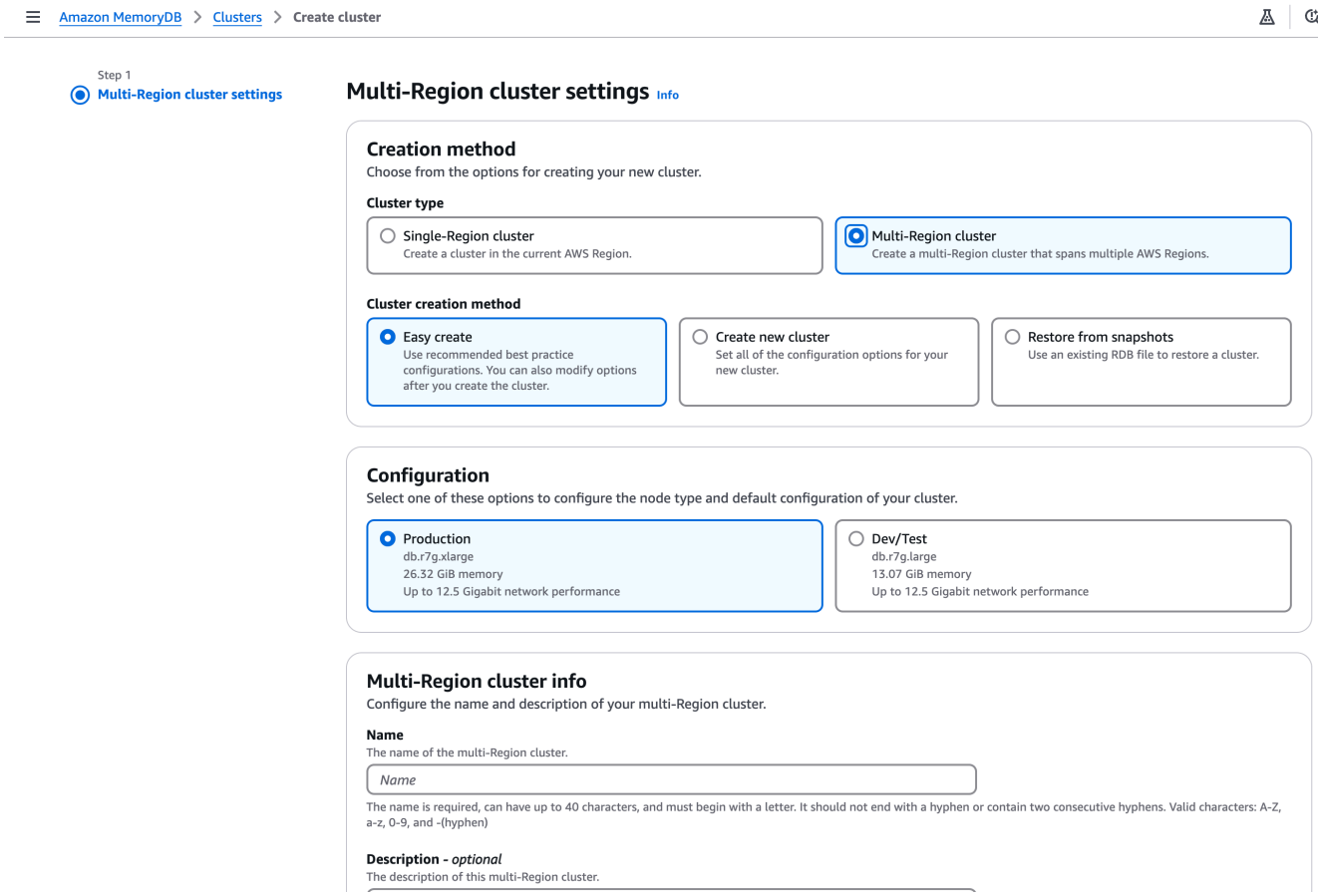
コンソールで MemoryDB マルチリージョンを使用する方法は以下のとおりです。

トピック

- MemoryDB マルチリージョンで新しいクラスターを作成する
- スナップショットをマルチリージョンクラスター内の新規または既存のクラスターに復元する
- MemoryDB マルチリージョンでクラスターを変更する
- MemoryDB マルチリージョンでクラスターを削除する

MemoryDB マルチリージョンで新しいクラスターを作成する

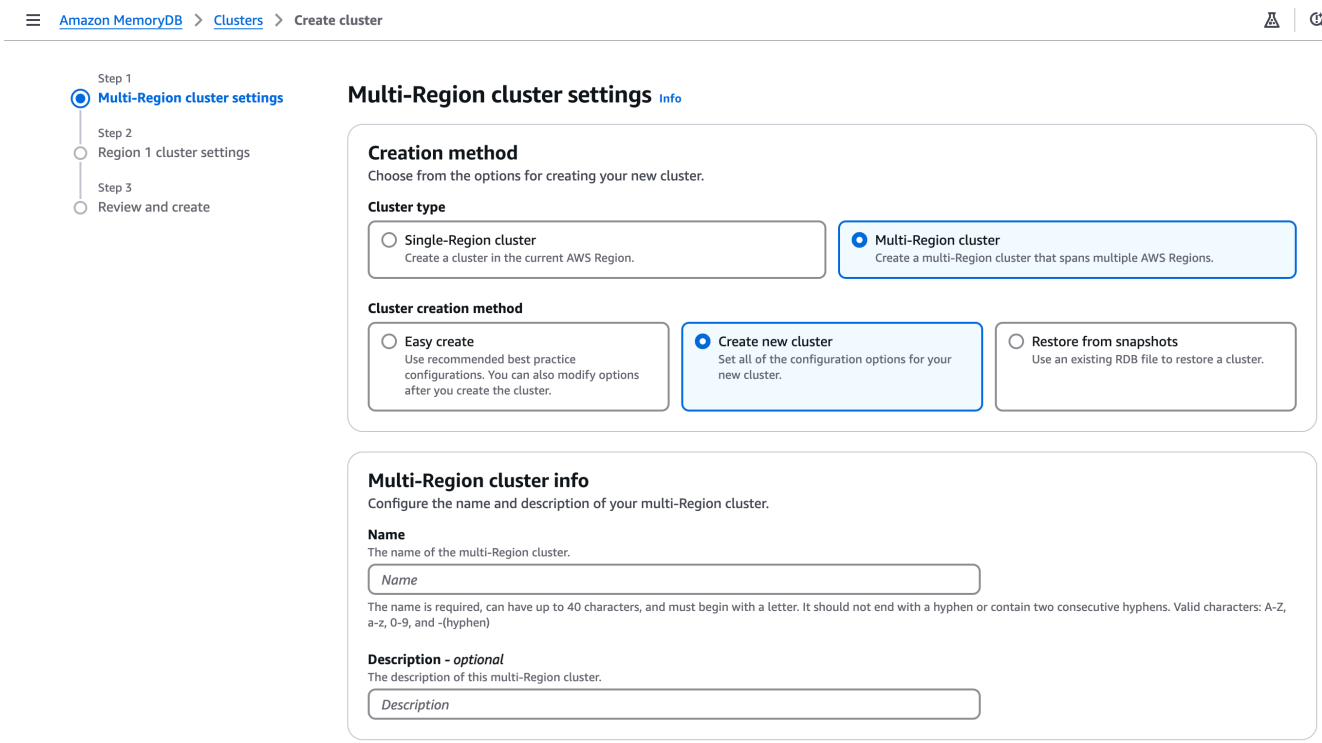
1. クラスターリストまたはダッシュボードから [クラスターを作成] セクションに移動します。



2. [クラスタータイプ]フィールドで、[マルチリージョンクラスター] を選択します。
3. [クラスターの作成方法] フィールドで、[簡易作成] を選択します。
4. [名前] と [説明] を入力し、デフォルト値を確認して、[作成] を選択します。

クラスターを作成および設定する

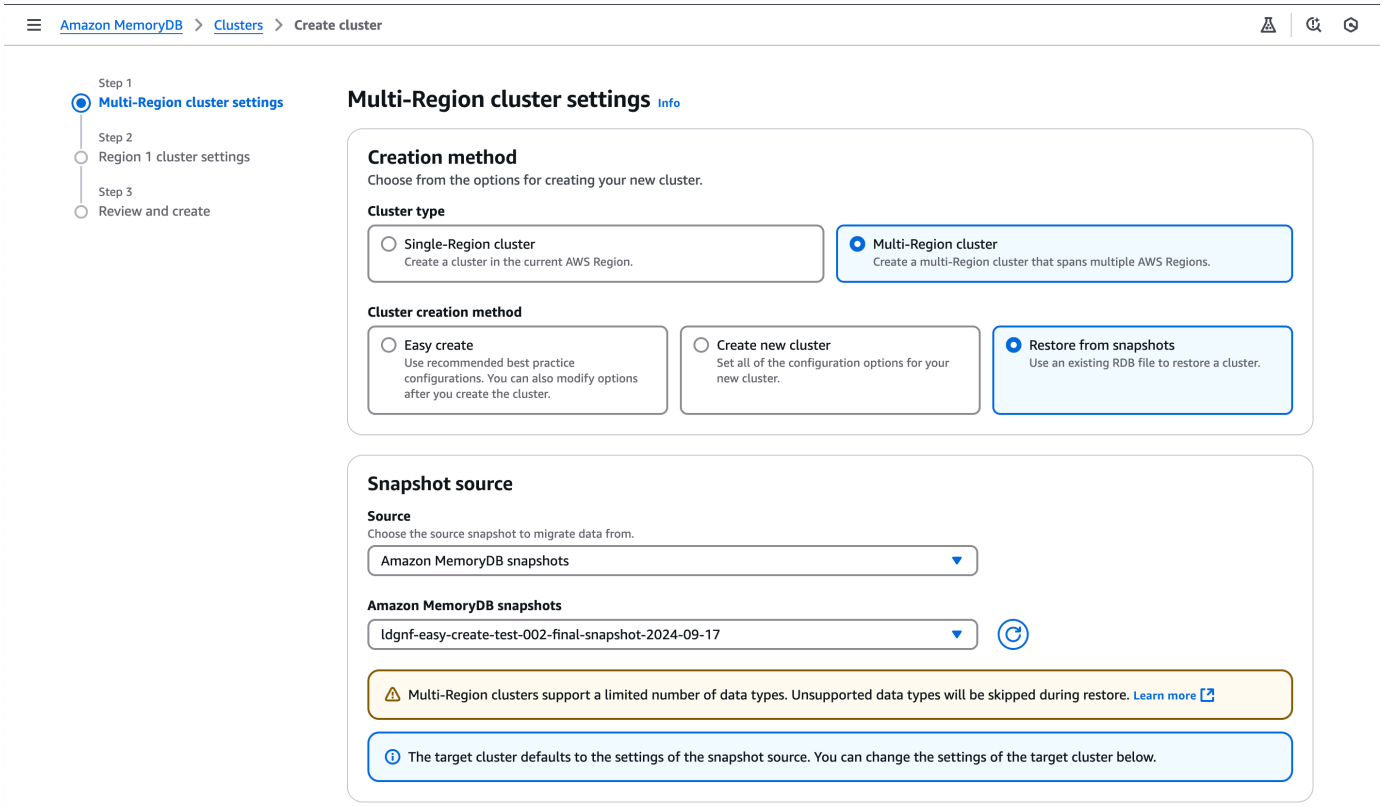
1. クラスターリストまたはダッシュボードから [クラスターを作成] セクションに移動します。



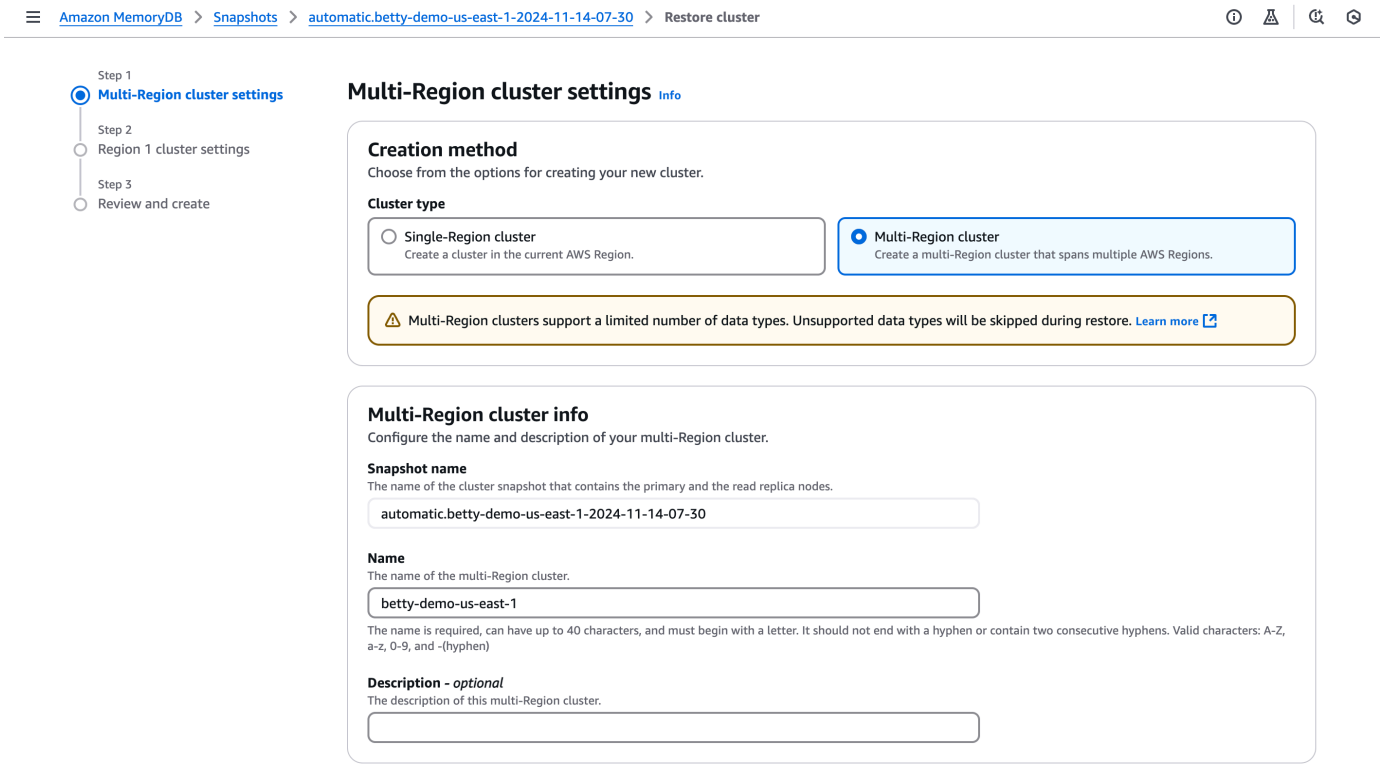
2. [クラスタータイプ]フィールドで、[マルチリージョンクラスター] を選択します。
3. [クラスターの作成方法] フィールドで、[新しいクラスターを作成] を選択します。
4. [名前] と [説明] を入力し、デフォルト値を確認して、[作成] を選択します。

スナップショットをマルチリージョンクラスター内の新規または既存のクラスターに復元する

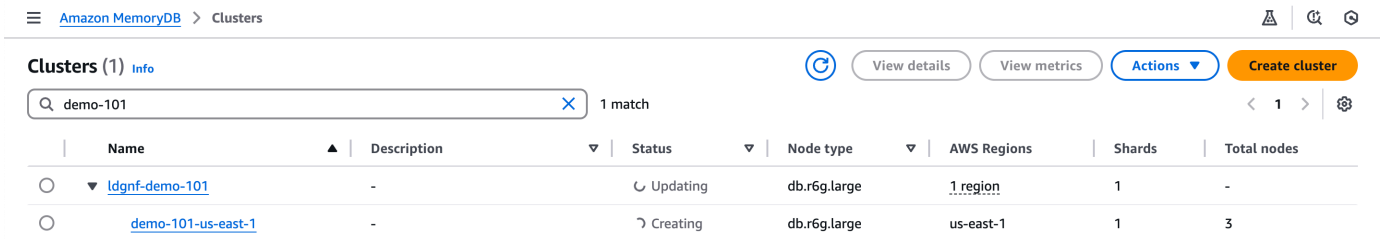
1. クラスターリストまたはダッシュボードから [クラスターを作成] セクションに移動します。



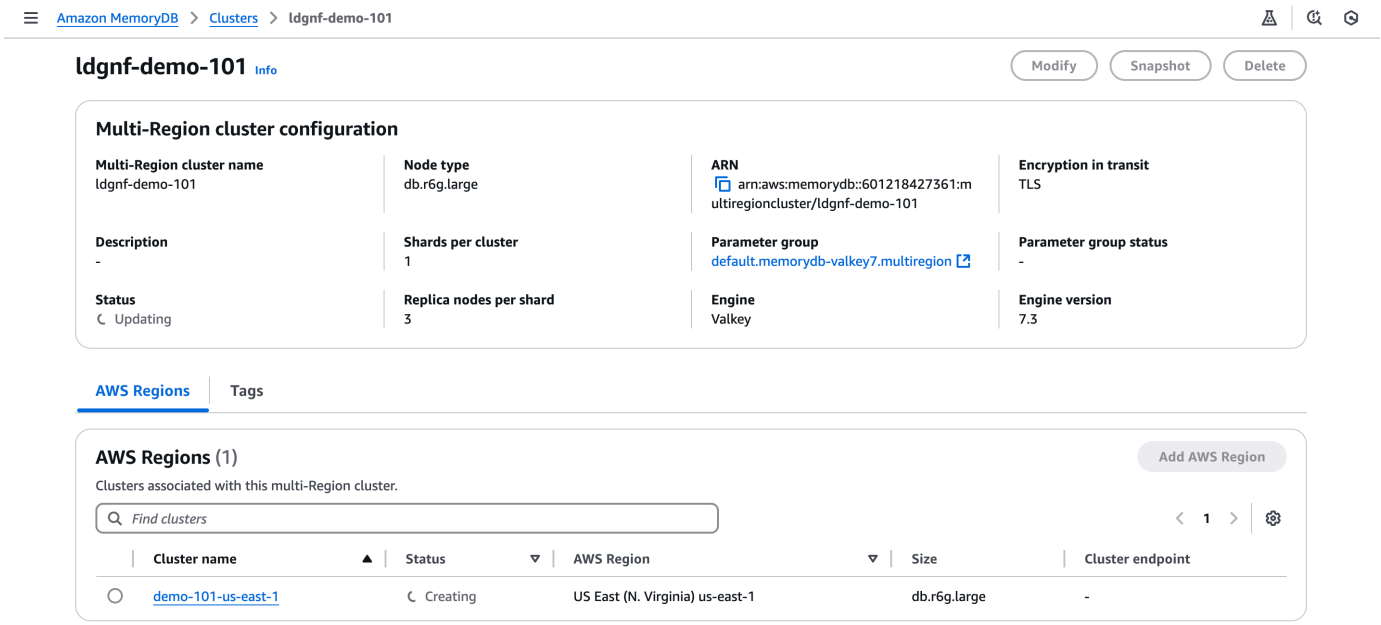
2. [クラスタータイプ] フィールドで、[マルチリージョンクラスター] を選択します。
3. [クラスターの作成方法] フィールドで、[スナップショットから復元] を選択します。
4. ソーススナップショットを選択し、必須フィールドに入力します。選択内容を確認し、[復元] を選択します。



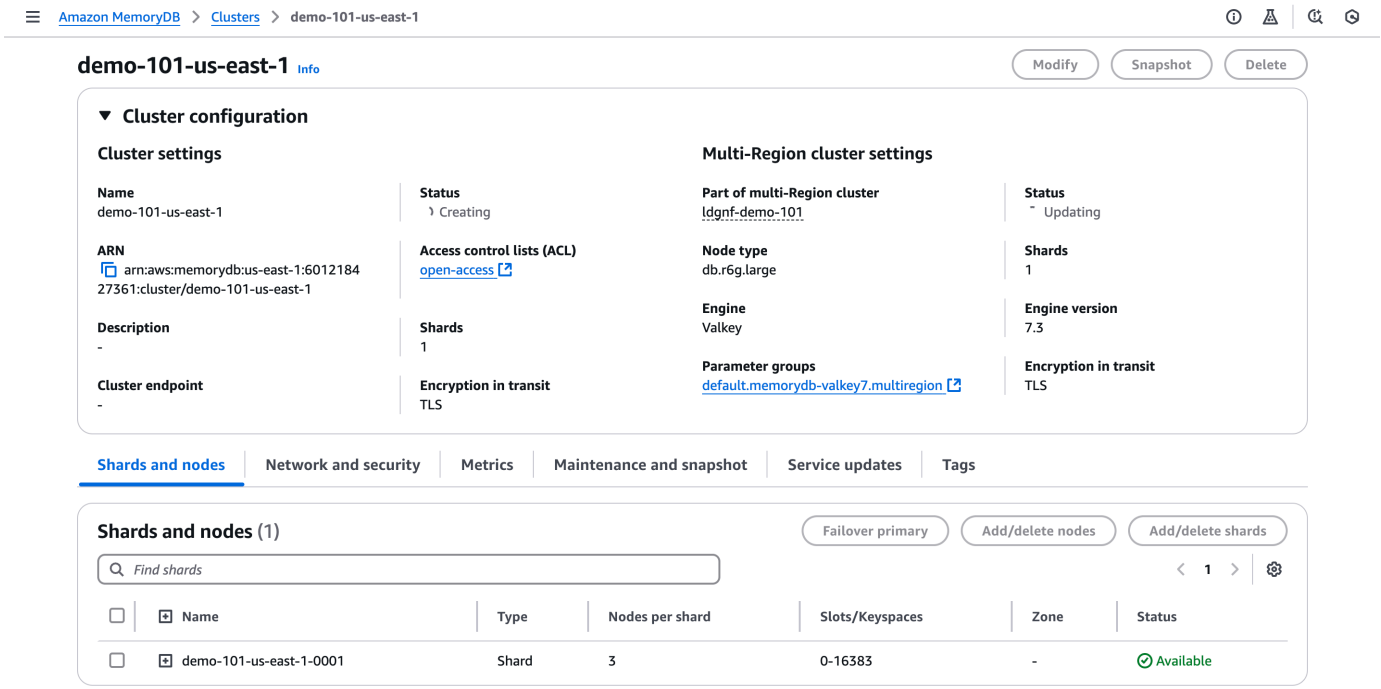
5. マルチリージョンクラスターを表示するには、[クラスター] セクションに移動します。



6. 次に、ターゲットのマルチリージョンクラスター名を選択します。

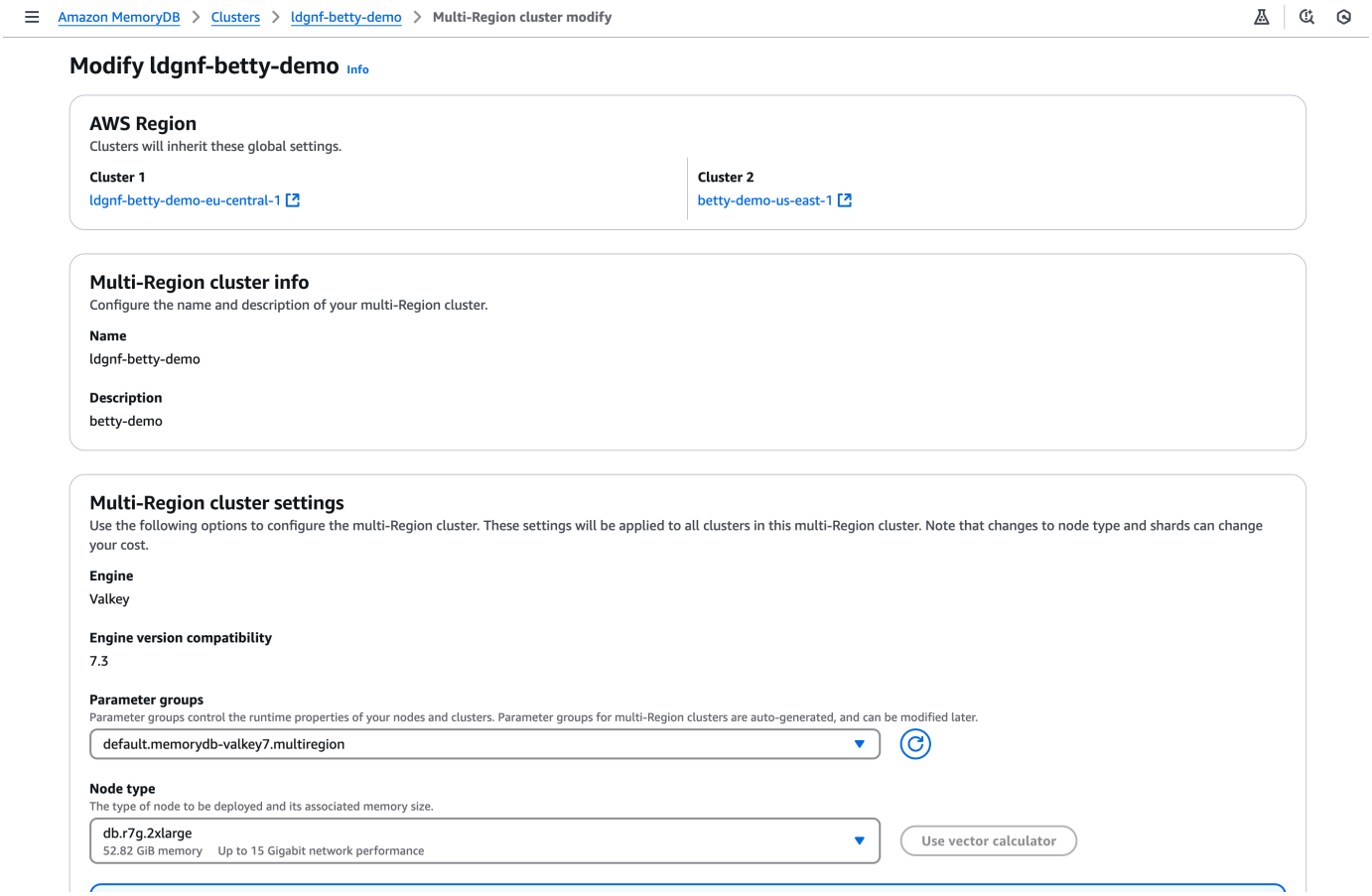


7. 次に、ターゲットのリージョンクラスター名を選択します。



MemoryDB マルチリージョンでクラスターを変更する

1. [クラスター] セクションに移動します。現在のクラスターがすべて表示されます。

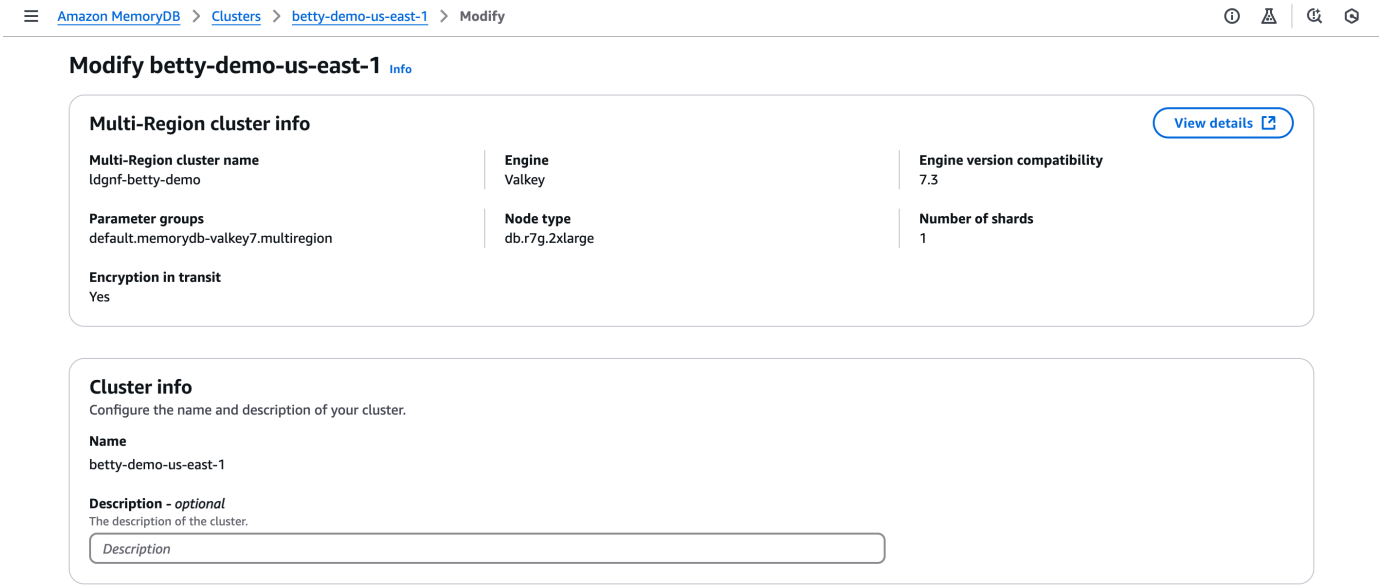


次に、変更するクラスターのタイプに応じて次のいずれかの手順を選択します。

2. マルチリージョンクラスターを含む単一クラスターを変更するには、まずそのクラスターが属するマルチリージョンを選択します。次に、[アクション] (右上) の [編集] ボタンを選択します。次に、ターゲットの単一クラスターを選択します。このクラスターは、[詳細] ページから変更することもできます。

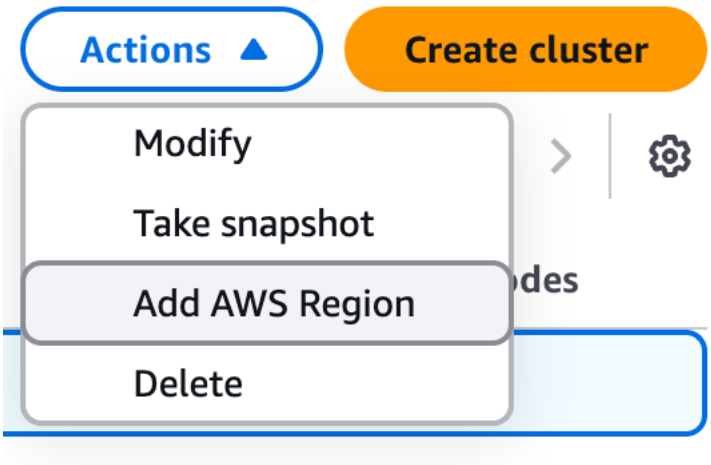
リージョンクラスターを変更する

1. マルチリージョンクラスターを変更するには、ターゲットのマルチリージョンクラスター名を選択します。

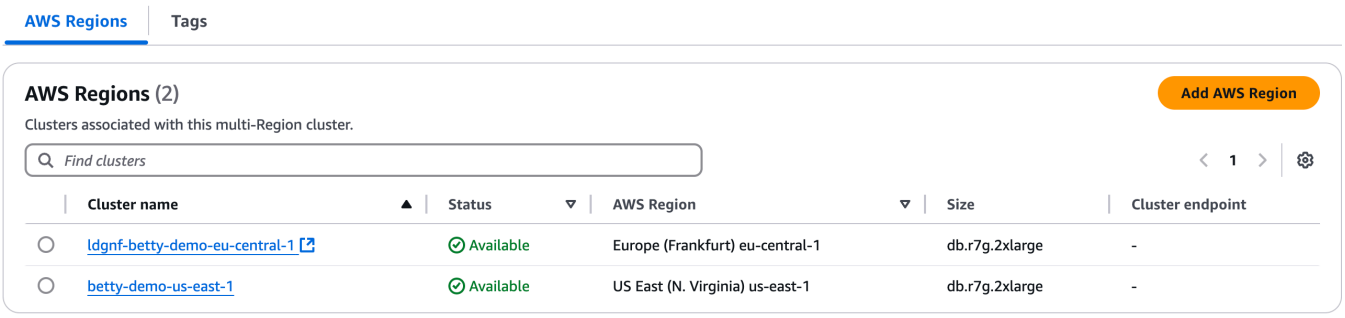


次に、クラスターを選択し、[アクション] (右上) または [詳細] ページの [編集] ボタンを選択します。

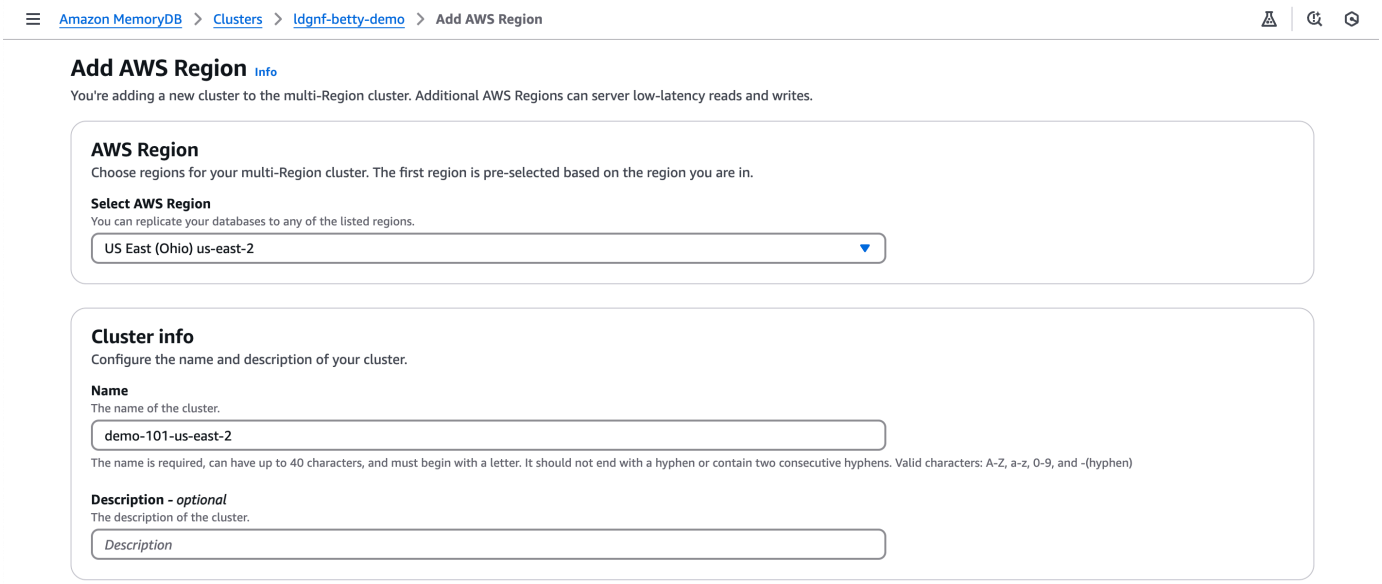
2. リージョンクラスターを追加するには、選択したターゲットマルチリージョンクラスターを選択し、[アクション] ドロップダウンに移動して [AWS リージョンを追加] を選択します。AWS リージョンの [詳細] ページに移動し、ターゲットのマルチリージョンクラスターを選択して、そこから追加することもできます。



3. リージョンを追加するには、ターゲットのリージョンを選択します。次に、必要な情報を入力し、[AWS リージョンを追加] を選択します。

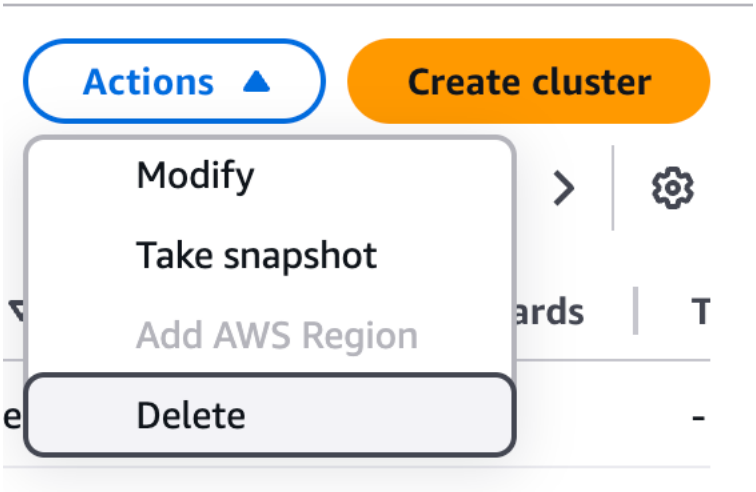


4. 空のマルチリージョンクラスターに新しいリージョンクラスターを追加する場合は、マルチリージョンクラスターの作成と同じオプションが表示されます。唯一の違いは、マルチリージョンクラスター情報が既に入力されていることです。

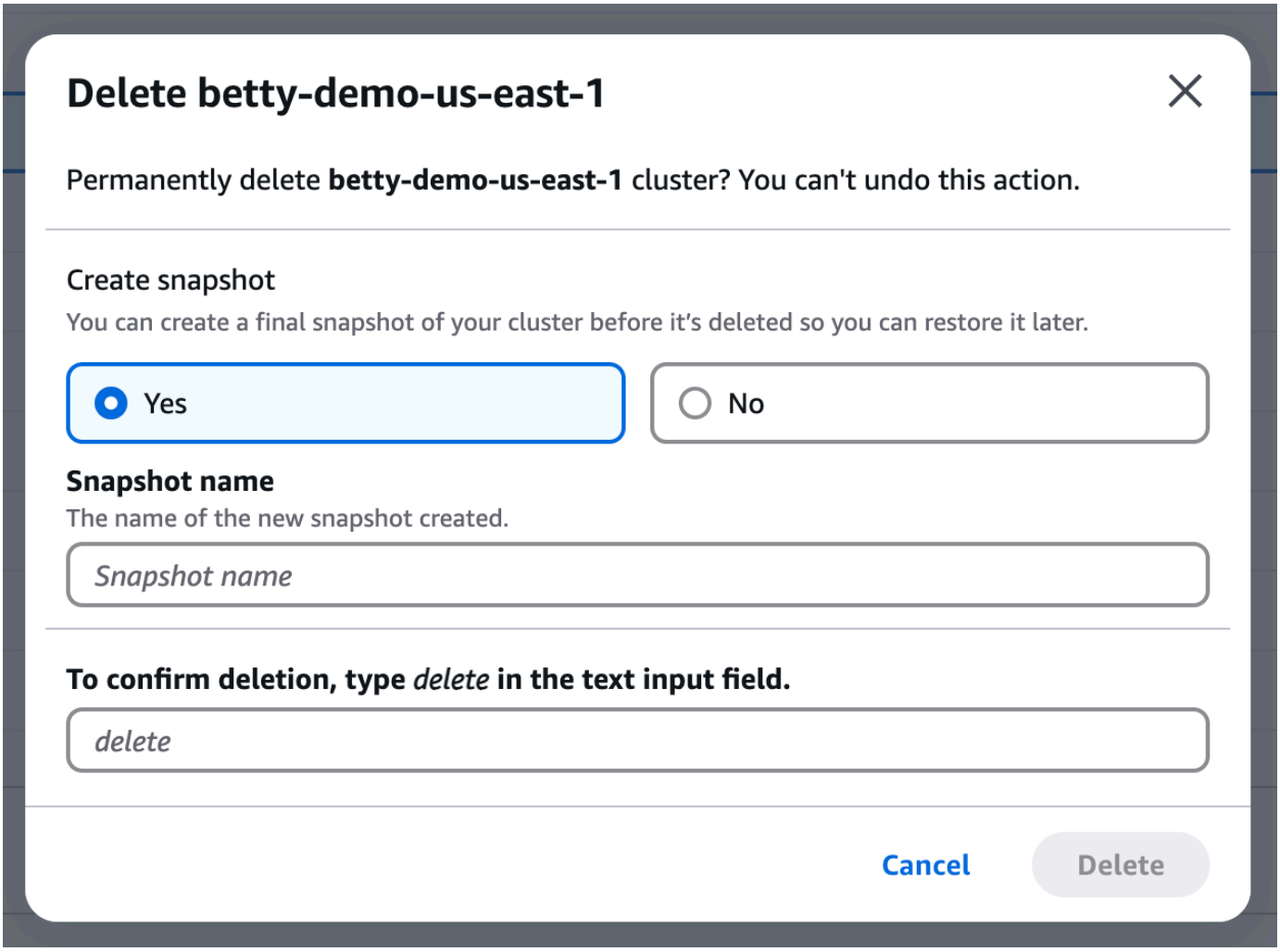


MemoryDB マルチリージョンでクラスターを削除する

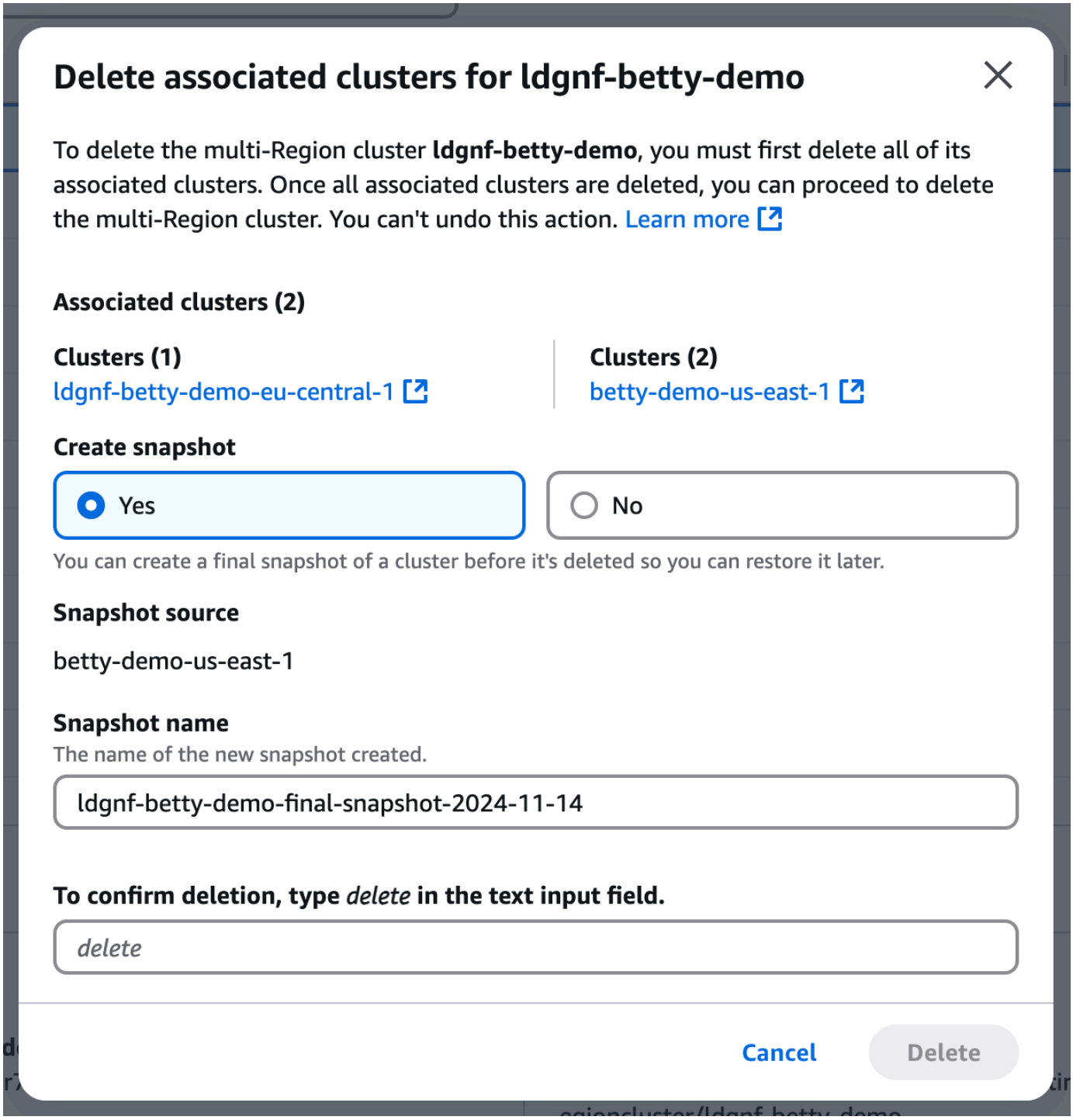
1. リージョン内の単一クラスターを削除するには、ターゲットのリージョンクラスターを選択します。次に、[アクション] メニュードロップダウンに移動し、個々のクラスターを選択して、[削除] を選択します。



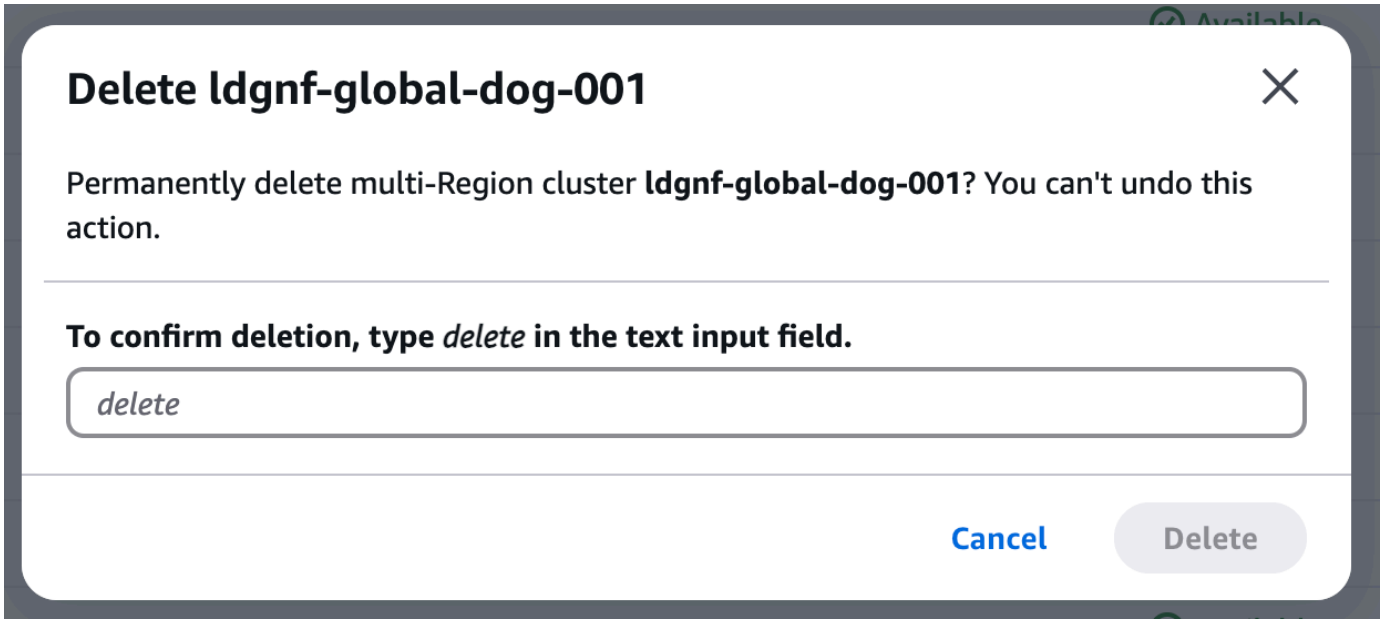
削除する前に確認ウィンドウが表示されます。このウィンドウには、スナップショットの作成オプションが表示されます。削除する場合は、テキストフィールドに「削除」と入力し、[削除] を選択します。



2. マルチリージョンクラスターに関連付けられているリージョンクラスターをすべて削除するには、1 つ以上のクラスターを含むターゲットマルチリージョンクラスターを選択します。次に、ターゲットのマルチリージョンクラスターが選択されている状態で [アクション] メニュードロップダウンに移動し、[削除] を選択します。



3. マルチリージョンクラスター全体を削除するには、ターゲットの空のマルチリージョンクラスターを選択します。次に、[アクション] メニュードロップダウンに移動し、[削除] を選択します。



CLI での MemoryDB マルチリージョンの使用

CLI で MemoryDB マルチリージョンを使用する方法は以下のとおりです。

> **Note**
>
> MemoryDB マルチリージョンでは、db.r7g.xlarge 以降のノードタイプのみがサポートされます。

MemoryDBMulti リージョンでのクラスターの作成

マルチリージョンクラスターを作成する

```
aws memorydb create-multi-region-cluster \
 --multi-region-cluster-name-suffix my-multi-region-cluster \
 --node-type db.r7g.xlarge \
 --engine valkey \
 --region us-east-1
```

米国東部 (バージニア北部) リージョンでリージョンクラスターを作成する

```
aws memorydb create-cluster \
 --cluster-name my-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --node-type db.r7g.xlarge \
 --acl-name open-access \
 --region us-east-1 \
```

欧州 (アイルランド) リージョンでリージョンクラスターを作成する

```
aws memorydb create-cluster \
 --cluster-name my-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --node-type db.r7g.xlarge \
 --acl-name open-access \
 --region eu-west-1 \
```

任意のリージョンのマルチリージョンクラスターについて説明する

```
aws memorydb describe-multi-region-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --region eu-west-1
```

マルチリージョンクラスターを更新する

ノードタイプの変更

```
aws memorydb update-multi-region-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --node-type db.r7g.4xlarge \
 --region us-east-1
```

シャード数の変更

```
aws memorydb update-multi-region-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --shard-configuration \
 ShardCount=3 \
 --update-strategy COORDINATED \
 --region us-east-1
```

MemoryDB クラスターのスケーリング

まず、`list-allowed-node-type-updates` コマンドを使用してスケールアップまたはスケールダウンできるノードを一覧表示します。

```
aws memorydb list-allowed-node-type-updates \
 --cluster-name my-cluster-name
```

これにより、スケールアップまたはスケールダウンできるノードのリストが表示されます。次に、それらのノードを更新するには、`update-cluster` コマンドを使用します。

```
aws memorydb update-cluster  \
 --cluster-name my-cluster \
 --node-type db.r6g.2xlarge
```

マルチリージョンでのスケーリングの詳細については、「MemoryDB マルチリージョンでのスケーリング」を参照してください。

MemoryDB マルチリージョンでのクラスターの削除

リージョンクラスターを削除する

```
aws memorydb delete-cluster \
 --cluster-name my-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --region us-east-1
```

マルチリージョンクラスターを削除する

```
aws memorydb delete-multi-region-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --region us-east-1
```

MemoryDB マルチリージョンのモニタリング

Amazon CloudWatch を使用して、マルチリージョンクラスターの動作とパフォーマンスをモニタリングできます。MemoryDB は、マルチリージョンクラスター内の各リージョンクラスターの `MultiRegionClusterReplicationLag` メトリクスを公開します。

MultiRegionClusterReplicationLag は、更新がリモートのマルチリージョンのリージョンクラスターにあるマルチ AZ トランザクションログに書き込まれてから、その更新がローカルのマルチリージョンのリージョンクラスターにあるプライマリノードに書き込まれるまでの経過時間を示します。このメトリクスは、ミリ秒単位で表され、すべてのレプリケート元リージョンとレプリケート先リージョンのペアについてシャードレベルで出力されます。

通常オペレーション中、MultiRegionClusterReplicationLag は完全に一定にする必要があります。MultiRegionClusterReplicationLag の値が増加している場合、1 つのリージョンクラスターからの更新が他のリージョンクラスターにタイムリーに伝播されていないことを示している可能性があります。これにより、時間の経過とともに、他のリージョンクラスターが遅れを取る可能性があります。これらのリージョンは更新を一貫して受け取ることができないからです。

MultiRegionClusterReplicationLag は、AWS リージョンの分離または機能低下が発生し、そのリージョンにリージョンクラスターが含まれている場合に増加することがあります。この場合、アプリケーションの読み取りおよび書き込みアクティビティを別の AWS リージョンに一時的にリダイレクトすることができます。

MemoryDB マルチリージョンでのスケーリング

クラスターの需要の変化に応じて MemoryDB のクラスター内のノードタイプまたはシャード数を変更することで、パフォーマンスを向上させたりコストを削減したりできます。MemoryDB マルチリージョンクラスターをスケーリングすると、そのクラスターに含まれているすべてのリージョンクラスターがスケーリングされます。MemoryDB マルチリージョンクラスターでは、オンラインリシャーディングがサポートされています。MemoryDB マルチリージョンクラスターでは、オフラインリシャーディングはサポートされていません。

クラスターを再スケーリングするかどうかの判断条件には、次のようなものがあります。

- メモリプレッシャー:

  リージョンクラスター内のノードがメモリプレッシャーを受けている場合、より多くのリソースがより効率よくデータを保存してリクエストを処理するようにスケールアウトまたはスケールアップできます。

  ノードがメモリプレッシャーを受けているかどうかは、FreeableMemory、SwapUsage、BytesUsedForMemoryDB、MultiRegionClusterReplicationLag といったメトリクスをモニタリングすることで判断できます。
- CPU やネットワークボトルネック:

レイテンシーやスループットがクラスターの問題となっている場合、問題解決のためにスケールアウトまたはスケールアップが必要な場合があります。

レイテンシーとスループットのレベルは、CPUUtilization、NetworkBytesIn、NetworkBytesOut、CurrConnections、 NewConnections、 and MultiRegionClusterReplicationLag の各メトリクスを監視することでモニタリングできます。

- クラスターのサイズが大きすぎます:

  現在のクラスターの需要からすると、スケールインまたはスケールダウンを行ってもパフォーマンスに影響せず、コストも削減できます。

クラスターの使用状況をモニタリング
し、FreeableMemory、SwapUsage、BytesUsedForMemoryDB、CPUUtilization、NetworkBytesIn、Networ
および MultiRegionClusterReplicationLag といったメトリクスを使用して安全にスケールインまたはスケールダウンできるかどうかを判断できます。

MemoryDB マルチリージョンクラスターをスケールするには、水平スケーリングと垂直スケーリングの 2 つの方法があります。

- 水平スケーリングでは、シャードを追加または削除することで、MemoryDB マルチリージョンクラスター内のシャードの数を変更できます。オンラインのリシャーディングプロセスでは、リージョンクラスターによる受信リクエストの処理を継続しながら、スケールイン/スケールアウトすることができます。
- 垂直スケーリングでは、ノードタイプを変更することで、MemoryDB マルチリージョンクラスターのサイズを変更します。オンラインの垂直スケーリングでは、リージョンクラスターによる受信リクエストの処理を継続しながら、スケールアップ/スケールダウンすることができます。

スケーリングでは、デフォルトで「調整された」更新戦略が使用されます。つまり、すべてのリージョンクラスターが正常にスケールされるか、どのリージョンクラスターもスケールされないかのいずれかです。

スケールアウトオペレーションでは、「調整されない」更新戦略もサポートされます。つまり、正常にスケールアウトされるリージョンクラスターもあれば、スケールアウト試行に失敗するリージョンクラスターもあるということです。1 つのリージョンクラスターのスケールアウトが成功した場合、それ以外のすべてのリージョンクラスターは、それぞれのスケールアウトも成功するまでスケールアウトを再試行し続けます。

すべてのリージョンクラスターがスケールアウトに失敗した場合、マルチリージョンクラスターは「調整されない」スケールアウトに失敗します。

> **Note**
>
> 「調整されない」スケールアウトでは、各リージョンクラスターのスケールアウトが異なる時間に行われた場合、リージョンクラスター間での容量の不均衡が長期間続く可能性があります。これにより、MultiRegionClusterReplicationLag メトリクスが増加し、リージョンクラスターのデータが長時間にわたって異なるものになる可能性があります。

MemoryDB マルチリージョンクラスターのリージョンクラスターはレプリカノードの数の設定を異なるものにすることができますが、リージョンクラスター内のシャードはすべて同じ数のレプリカノードを持ちます。

スケールインまたはスケールダウンのいずれかを実行して MemoryDB マルチリージョンクラスターのサイズとメモリ容量を減らす場合は、新しい設定にデータ用の十分なメモリおよび空き IP と十分なエンジンオーバーヘッドがあること、およびリージョンクラスターの MultiRegionClusterReplicationLag メトリクスが数秒以内または 1 分以内であることを確認してください。

AWS マネジメントコンソール、AWS CLI、および MemoryDB API を使用して、MemoryDB マルチリージョンクラスターを水平および垂直にスケールできます。

サポートされているコマンドとサポートされていないコマンド

サポートされているコマンド

> **Note**
>
> - SET コマンドでは現在、EX、PX、EXAT、PXAT、および KEEPTTL オプションがサポートされていません。
> - RESTORE コマンドでは、TTL をゼロ以外の値に設定することができません。ABSTTL、IDLETIME、および FREQ オプションもサポートされていません。

データ型	commands

データ型	commands
文字列	SET*、DECR、DECRBY、GET、GETRANGE、SUBSTR、GETDEL、GETSET、INCR、INCRBY、INCRBYFLOAT、MGET、MSET、MSETNX、SETNX、STRLEN、LCS
ハッシュ	HINCRBY、HINCRBYFLOAT、HDEL、HSET、HMSET、HGET、HEXISTS、HLEN、HKEYS、HVALS、HGETALL、HMGET、HSTRLEN、HSETNX、HRANDFIELD、HSCAN
設定	SADD、SREM、SISMEMBER、SMISMEMBER、SCARD、SMEMBERS、SRANDMEMBER、SSCAN、SUNION、SINTERCARD、SINTER、SDIFF、SPOP
ソートされたセット	ZADD、ZINCRBY、ZSCORE、ZMSCORE、ZCARD、ZRANK、ZREVRANK、ZRANGE、ZRANGEBYSCORE、ZRANGEBYLEX、ZREVRANGE、ZREVRANGEBYLEX、ZREVRANGEBYSCORE、ZREMRANGEBYLEX、ZREMRANGEBYSCORE、ZREMRANGEBYRANK、ZUNION、ZINTER、ZINTERCARD、ZDIFF、ZLEXCOUNT、ZCOUNT、ZREM、ZMPOP、ZPOPMIN、ZPOPMAX、ZSCAN、ZRANDMEMBER
ジェネリック	SCAN、DEL、UNLINK、DUMP、RESTORE**、EXISTS、KEYS、RANDOMKEY、TYPE

サポートされていないコマンド

サポートされていないコマンドの一般的なカテゴリは、サポートされていないデータ型 (ビットマップ、Hyperloglog、リスト、地理空間、ストリーム)、TTL 関連のコマンド、ブロックコマンド、および関数関連のコマンドです。完全なリストを次に示します。

データ型	commands
文字列	APPEND、GETEX、SETEX、SETRANGE
Bitmap	BITCOUNT、BITFIELD、BITFIELD_RO、BITOP、BITPOS、GETBIT、SETBIT
Hyperloglog	PFADD、PFCOUNT、PFDEBUG、PFMERGE、PFSELFTEST
リスト	BLMOVE、BLMPOP、BLPOP、BRPOP、BRPOPLPUSH、LINDEX、LINSERT、LLEN、LMOVE、LMPOP、LPOP、LPOS、PUSH、LPUSHX、LRANGE、LREM、LSET、LTRIM、RPOP、RPOPLPUSH、RPUSH、RPUSHX
設定	SMOVE、SUNIONSTORE、SDIFFSTORE、SINTERSTORE
ソートされたセット	BZMPOP、BZPOPMAX、BZPOPMIN、ZDIFFSTORE、ZINTERSTORE、ZRANGESTORE、ZUNIONSTORE
地理空間	GEOADD、GEODIST、GEOHASH、GEOPOS、GEORADIUS、GEORADIUS_RO、GEORADIUSBYMEMBER、GEORADIUSBYMEMBER_RO、GEOSEARCH、GEOSEARCHSTORE
ストリーム	XACK、XADD、XAUTOCLAIM、XCLAIM、XDEL、XLEN、XPENDING、XRANGE、XREAD、XREADGROUP、XREVRANGE、XSETID、XTRIM、XGROUP、XINFO

データ型	commands
ジェネリック	COPY、FLUSHDB、FLUSHALL、MOVE、RENAME、RENAMENX、SORT、SORT_RO、SWAPDB、OBJECT、FUNCTION、FCALL、FCALL_RO、EXPIRE、EXPIREAT、EXPIRETIME、PERSIST、PEXPIRE、PEXPIREAT、PEXPIRETIME、PSETEX、PTTL、TTL

MemoryDB 内のセキュリティ

「AWS」 ではクラウドセキュリティが最優先事項です。セキュリティを最も重視する組織の要件を満たすために構築された 「AWS」 のデータセンターとネットワークアーキテクチャは、お客様に大きく貢献します。

セキュリティは、AWS とお客様とが共有する責務です。責任共有モデルではこれをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ - AWS はAWS Cloud で AWS の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を負います。また、AWS は使用するサービスを安全に提供します。AWS コンプライアンスプログラムの一環として、サードパーティー監査者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ます。MemoryDB に適用するコンプライアンスプログラムの詳細については、コンプライアンスプログラムによる AWS 対象範囲内のサービスを参照してください。
- クラウド内のセキュリティ – お客様の責任範囲は、ご使用の AWS のサービスに応じて異なります。また、お客様は、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、MemoryDB を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。ここでは、セキュリティとコンプライアンスの目標を満たすように MemoryDB を設定する方法を説明します。MemoryDB リソースのモニタリングやセキュリティ確保に役立つ他の AWS のサービスの使用方法についても説明します。

内容

- MemoryDB 内のデータ保護
- MemoryDB でのアイデンティティとアクセス権の管理
- ログ記録とモニタリング
- MemoryDB のコンプライアンス検証
- MemoryDB のインフラストラクチャセキュリティ
- ネットワーク間のトラフィックのプライバシー
- MemoryDB のサービスの更新

MemoryDB 内のデータ保護

AWS 責任共有モデル は、データ保護に適用されます。このモデルで説明されているように、AWS は、AWS クラウド のすべてを実行するグローバルインフラストラクチャを保護するがあります。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「AWS のサービス」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された AWS 責任共有モデルおよび GDPR のブログ記事を参照してください。

データを保護するため、「AWS アカウント」 認証情報を保護し、「AWS IAM Identity Center」 または 「AWS Identity and Access Management」 (IAM) を使用して個々のユーザーをセットアップすることをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して 「AWS」 リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- AWS CloudTrail で API とユーザーアクティビティロギングを設定します。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「AWS CloudTrail ユーザーガイド」の「CloudTrail 証跡の使用」を参照してください。
- AWS のサービス 内のすべてのデフォルトセキュリティコントロールに加え、AWS 暗号化ソリューションを使用します。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API を使用して 「AWS」 にアクセスする際に FIPS 140-3 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、AWS CLI、または AWS SDK を使用して他の AWS のサービス 操作する場合も同様です。タグ、または名前に使用される自由形式のテキストフィールドに入力されるデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

MemoryDB 内のデータのセキュリティ

データを安全に保つために、MemoryDB および Amazon EC2 は、サーバーのデータへの不正アクセスに対する防御メカニズムを提供します。

MemoryDB には、クラスター上のデータの暗号化機能も用意されています。

- 転送時の暗号化では、ある場所から別の場所に移動するデータ (クラスターのノード間、クラスターとアプリケーション間など) に対して必ず暗号化が行なわれます。
- 保管時の暗号化では、スナップショット操作中にトランザクションログとオンディスクデータが暗号化されます。

アクセスコントロールリスト (ACL) によるユーザー認証 を使用してクラスターへのユーザーアクセス制御も可能です。

トピック

- MemoryDB に保存時の暗号化
- MemoryDBの転送時の暗号化 (TLS)
- アクセスコントロールリスト (ACL) によるユーザー認証
- IAM を使用した認証

MemoryDB に保存時の暗号化

データを安全に保つために、MemoryDB と Amazon S3 は、クラスター内のデータへのアクセスを制限するさまざまな方法を用意しています。詳細については、「MemoryDB と Amazon VPC」および「MemoryDB でのアイデンティティとアクセス権の管理」を参照してください。

MemoryDB の保管時の暗号化は常に有効になっており、永続データを暗号化することでデータのセキュリティを強化します。以下の項目を暗号化します。

- トランザクションログ内のデータ
- 同期、スナップショット、およびスワップオペレーション中のディスク
- Amazon S3 に保存されたバックアップ

MemoryDB は、保管時のデフォルト (サービス管理) の暗号化だけでなく、AWS Key Management Service (KMS) で独自の対称カスタマー管理カスタマールートキーを使用する機能を提供します。

データ階層化が有効なクラスター内の SSD (ソリッドステートドライブ) に保存されたデータは、デフォルトで常時暗号化されます。

転送時の暗号化については、「MemoryDBの転送時の暗号化 (TLS)」を参照してください。

トピック

- AWS KMS のカスタマー管理のキーの使用
- 以下の資料も参照してください。

AWS KMS のカスタマー管理のキーの使用

MemoryDB は、保管時の暗号化用の対称カスタマー管理の KMS キー (KMS キー) をサポートしています。カスタマー管理の KMS キーは、AWS アカウントで作成、所有、管理される暗号化キーです。詳細については、「AWSKey Management Service デベロッパーガイド」の「カスタマールートキー」を参照してください。キーは、MemoryDB で使用する前に AWS KMS で作成する必要があります。

AWS KMS ルートキーを作成する方法の詳細については、AWS Key Management Service デベロッパーガイドの「キーの作成」を参照してください。

MemoryDB を使用すると、AWS KMS と統合できます。詳細については、AWS Key Management Service デベロッパーガイドの「付与の使用」を参照してください。MemoryDB と AWS KMS の統合を有効にするために、お客様の作業は必要ありません。

kms:ViaService 条件キーは、AWS KMS キーの使用を、指定された AWS サービスからのリクエストに制限します。MemoryDB で kms:ViaService を使用するには、両方のViaService 名を条件キーの値 memorydb.amazon_region.amazonaws.com に含めます。 詳細については、「kms:ViaService」を参照してください。

AWSCloudTrail を使用して、MemoryDB によってお客様に代わって AWS Key Management Service に送信されるリクエストを追跡できます。カスタマー管理のキーに関連する AWS Key Management Service へのすべての API コールには、対応する CloudTrail ログがあります。ListGrants KMS API コールを行うことで、MemoryDB によって作成される許可を表示することもできます。

カスタマー管理のキーを使用してクラスターが暗号化されると、クラスターのすべてのスナップショットは以下のように暗号化されます。

- 毎日の自動スナップショットは、クラスターに関連付けられたカスタマー管理のキーを使用して暗号化されます。
- クラスターが削除されたときに作成される最終スナップショットも、クラスターに関連付けられたカスタマー管理のキーを使用して暗号化されます。
- 手動で作成されたスナップショットは、デフォルトで、クラスターに関連付けられた KMS キーを使用して暗号化されます。この動作は、別のカスタマー管理のキーを選択して上書きできます。
- スナップショットをコピーするとき、デフォルトでは、ソーススナップショットに関連付けられたカスタマー管理のキーが使用されます。この動作は、別のカスタマー管理のキーを選択して上書きできます。

> **Note**
>
> - 選択した Amazon S3 バケットにスナップショットをエクスポートするとき、カスタマー管理のキーは使用できません。ただし、Amazon S3 にエクスポートされたすべてのスナップショットは、サーバー側の暗号化を使用して暗号化されます。 スナップショットファイルを新しい S3 オブジェクトにコピーし、カスタマー管理の KMS キーを使用して暗号化するか、KMS キーを使用してデフォルトの暗号化が設定された別の S3 バケットにコピーするか、ファイル自体の暗号化オプションを変更するかを選択できます。
> - カスタマー管理のキーを使用して、暗号化にカスタマー管理のキーを使用しない手動で作成されたスナップショットを暗号化することもできます。このオプションを使用すると、

データが元のクラスターで暗号化されていない場合でも、Amazon S3 に保存されているスナップショットファイルは KMS キーを使用して暗号化されます。

スナップショットから復元するときは、新しいクラスターの作成時に使用できる暗号化オプションと同様に、使用可能な暗号化オプションから選択できます。

- キーを削除するか、キーを無効化して、クラスターの暗号化に使用したキーの許可を取り消すと、クラスターは回復不可能になります。 つまり、ハードウェア障害の後に変更も回復もできなくなります。AWSルートキーは 7 日間以上の待機期間後にのみ KMS によって削除されます。キーが削除された後、別のカスタマー管理のキーを使用して、アーカイブ目的のスナップショットを作成できます。
- 自動キー更新は AWS KMS ルートキーのプロパティを保持するため、お客様が MemoryDB データにアクセスできるかどうかには影響しません。暗号化された MemoryDB クラスターは、新しいルートキーの作成と古いキーへの参照の更新を伴う手動キーローテーションをサポートしません。詳細については、「AWS Key Management Service デベロッパーガイド」の「 Rotating Customer root Keys」を参照してください。
- KMS キーを使用して MemoryDB クラスターを暗号化するには、クラスターごとに 1 つの許可が必要です。この許可はクラスターの有効期間を通じて使用されます。さらに、スナップショットの作成時には、スナップショットごとに 1 つの権限が使用されます。この許可はスナップショットの作成後に無効になります。
- AWS KMS の付与と制限の詳細については、「AWS Key Management Service デベロッパーガイド」の「クォータ」を参照してください。

以下の資料も参照してください。

- MemoryDBの転送時の暗号化 (TLS)
- MemoryDB と Amazon VPC
- MemoryDB でのアイデンティティとアクセス権の管理

MemoryDBの転送時の暗号化 (TLS)

データを安全に保つために、MemoryDB および Amazon EC2 は、サーバーのデータへの不正アクセスに対する防御メカニズムを提供します。MemoryDB では転送時の暗号化機能を提供されるため、ある場所から別の場所に移動しているデータの保護ツールとして使用できます。例えば、クラスター

内、またはクラスターとアプリケーションの間でプライマリノードからリードレプリカノードにデータを移動するとします。

トピック

- 転送時の暗号化の概要
- 関連情報

転送時の暗号化の概要

MemoryDB の転送時の暗号化は、データがある場所から別の場所に転送されるときに、最も脆弱なポイントでのデータのセキュリティを強化できる機能です。

MemoryDB 転送時の暗号化では、次の機能が実装されます。

- 暗号化接続-サーバー接続もクライアント接続もTransport Layer Security (TLS)で暗号化されている。
- 暗号化レプリケーション — プライマリノードとレプリカ ノード間を移動するデータが暗号化されます。
- サーバー認証 — クライアントは、適切なサーバーに接続していることを認証できます。

2023 年 7 月 20 日以降、新規および既存のクラスターでサポートされる最小バージョンは TLS 1.2 です。AWS のTLS 1.2 の詳細については、こちらの「リンク」を参照してください。

MemoryDB クラスターとの接続の詳細については、「redis-cli を使用して MemoryDB ノードに接続する」を参照してください。

関連情報

- MemoryDB に保存時の暗号化
- アクセスコントロールリスト (ACL) によるユーザー認証
- MemoryDB と Amazon VPC
- MemoryDB でのアイデンティティとアクセス権の管理

アクセスコントロールリスト (ACL) によるユーザー認証

アクセスコントロールリスト (ACL) を使用してユーザーを認証できます。

ACL を使用すると、ユーザーをグループ化してクラスターアクセスを制御できます。これらのアクセスコントロールリストは、クラスターへのアクセスを分類する方法として設計されています。

ACL では、以下で説明されているように、アクセス文字列を使用してユーザーを作成し、ユーザーに特定のアクセス許可を割り当てます。特定のロール (管理者、人事) と連携したアクセスコントロールリストにユーザーを割り当てます。その後、それらは 1 つ以上の MemoryDB クラスターにデプロイされます。これにより、同じMemoryDBクラスターまたはクラスターを使用するクライアント間にセキュリティ境界を設定し、クライアントが互いのデータにアクセスできないようにすることができます。

ACL は、Redis OSS 6 の ACL の導入をサポートするように設計されています。MemoryDB クラスターで ACL を使用する場合は、いくつかの制約があります。

- アクセス文字列にパスワードを指定することはできません。パスワードは CreateUser または UpdateUser コールで設定します。
- ユーザー権限については、on および `off` をアクセス文字列の一部としてパスします。アクセス文字列にどちらも指定されていない場合、ユーザーには `off` が割り当てられ、クラスターへのアクセス権はありません。
- 禁止されたコマンドは使用できません。禁止されているコマンドを指定すると、例外がスローされます。これらのコマンドの一覧については、「制限されるコマンド」を参照してください。
- `reset` コマンドを、アクセス文字列の一部として使用することはできません。API パラメータを用いてパスワードを指定すると、MemoryDB がパスワードを管理します。したがって、`reset` を使用することはできません。それによりユーザーのすべてのパスワードが削除されるからです。
- Redis OSS 6 は、ACL LIST コマンドを導入します。このコマンドは、ユーザーのリストと、各ユーザーに適用される ACL ルールを返します。MemoryDB は `ACL LIST` コマンドをサポートしますが、Redis OSS のようにパスワードハッシュのサポートは含まれていません。MemoryDB を使用すると、DescribeUsers オペレーションを使用して、アクセス文字列に含まれるルールなど、同様の情報を取得できます。ただし、DescribeUsers は、ユーザーパスワードを取得しません。

  MemoryDB でサポートされているその他の読み取り専用コマンドには、ACL WHOAMI、ACL USERS、ACL CATなどがあります。MemoryDB は、他の書き込みベースの ACL コマンドをサポートしていません。

MemoryDB での ACL の使用については、以下で詳しく説明します。

トピック

- アクセス文字列を使用したアクセス許可の指定

- ベクトル検索機能
- MemoryDB のクラスターに ACL を適用します

アクセス文字列を使用したアクセス許可の指定

MemoryDB クラスターへのアクセス許可を指定するには、アクセス文字列を作成し、AWS CLI または AWS マネジメントコンソール を使用して、それをユーザーに割り当てます。

アクセス文字列は、ユーザーに適用されるスペース区切りルールのリストとして定義されます。それらは、ユーザーが実行できるコマンドと、ユーザーが操作できるキーを定義します。コマンドを実行するには、ユーザーは、実行されているコマンドと、そのコマンドによってアクセスされているすべてのキーにアクセスできる必要があります。ルールは左から右に累積的に適用され、提供された文字列に冗長性がある場合は、提供された文字列の代わりに、より単純な文字列を使用できます。

ACL ルールの構文の詳細については、「ACL」を参照してください。

次の例では、アクセス文字列は、使用可能なすべてのキーおよびコマンドにアクセスできるアクティブなユーザーを表します。

```
on ~* &* +@all
```

アクセス文字列の構文は、次のように分類されます。

- on — ユーザーはアクティブなユーザーです。
- ~* — アクセス権はすべての使用可能なキーに与えられます。
- &* - アクセス権はすべての pubsub チャネルに与えられます。
- +@all — アクセス権はすべての使用可能なコマンドに与えられます。

上記の設定は、最も制限が緩い設定です。これらの設定を変更して、セキュリティを強化できます。

次の例では、アクセス文字列は「app::」キースペースで始まるキーに対する読み取りアクセスに制限されたアクセス権を持つユーザーを表します。

```
on ~app::* -@all +@read
```

ユーザーがアクセス権を持つコマンドを一覧表示することで、これらのアクセス許可をさらに絞り込むことができます。

+*command1* — ユーザーのコマンドへのアクセスは *command1* に制限されます。

+@category — ユーザーのアクセスは、コマンドのカテゴリーに制限されます。

アクセス文字列をユーザーに割り当てる方法については、「コンソールと CLI を使用したユーザーおよびアクセスコントロールリストの作成」を参照してください。

既存のワークロードを MemoryDB に移行する場合は、ACL LIST を呼び出すことでアクセス権を取得して、ユーザーおよびパスワードハッシュを除外できます。

ベクトル検索機能

ベクトル検索 では、すべての検索コマンドは @search カテゴリに属しており、検索コマンドを含むために既存のカテゴリ @read、@write、@fast、および @slow が更新されます。ユーザーがあるカテゴリにアクセスできない場合、そのユーザーは、そのカテゴリ内のいかなるコマンドにもアクセスできません。例えば、ユーザーが @search にアクセスできない場合、そのユーザーは、検索関連のいかなるコマンドも実行できません。

次の表は、適切なカテゴリへの検索コマンドのマッピングを示しています。

VSS コマンド	@read	@write	@fast	@slow
FT.CREATE		Y	Y	
FT.DROPINDEX		Y	Y	
FT.LIST	Y			Y
FT.INFO	Y		Y	
FT.SEARCH	Y			Y
FT.AGGREGATE	Y			Y
FT.PROFILE	Y			Y

VSS コマンド	@read	@write	@fast	@slow
FT.ALIASADD		Y	Y	
FT.ALIASDEL		Y	Y	
FT.ALIASUPDATE		Y	Y	
FT._ALIASLIST	Y			Y
FT.EXPLAIN	Y		Y	
FT.EXPLAINCLI	Y		Y	
FT.CONFIG	Y		Y	

MemoryDB のクラスターに ACL を適用します

MemoryDB ACL を使用するには、次のステップに従います。

1. 1 つ以上のユーザーを作成します。
2. ACL を作成し、ユーザーをリストに追加します。
3. ACL をクラスターに割り当てます。

これらのステップは、以下に詳細が説明されます。

トピック

- コンソールと CLI を使用したユーザーおよびアクセスコントロールリストの作成
- コンソールおよび CLI を使用したアクセスコントロールリストの管理

- アクセスコントロールリストのクラスターへの割り当て

コンソールと CLI を使用したユーザーおよびアクセスコントロールリストの作成

ACLユーザーのユーザー情報は、ユーザー名、およびオプションのパスワードとアクセス文字列です。アクセス文字列は、キーとコマンドでのアクセス許可レベルを提供します。この名前 はユーザーに対して一意であり、エンジンに渡されるものです。

指定するユーザー許可が、ACLの意図した目的に合っていることを確認してください。例えば、`Administrators` というACLを作成した場合、そのグループに追加するユーザーは、アクセス文字列をキーおよびコマンドへのフルアクセスに設定する必要があります。 `e-commerce` ACL のユーザーの場合、アクセス文字列を読み取り専用アクセスに設定できます。

MemoryDB は、アカウントごとにユーザー名を使用してデフォルトユーザー `"default"` を自動的に設定します。ACL に明示的に追加しない限り、どのクラスターにも関連付けられません。このユーザーを変更または削除することはできません。このユーザーは、以前の Redis OSS バージョンのデフォルト動作との互換性を意図して作成されており、すべてのコマンドを呼び出してすべてのキーにアクセスできるようにするアクセス文字列を持っています。

デフォルトユーザーを含むすべてのアカウントに対して、不変の「オープンアクセス」ACLが作成されます。これは、デフォルトユーザーがメンバーになれる唯一の ACL です。クラスターを作成するときに、クラスター関連付けるACLを選択する必要があります。デフォルトユーザーで「オープンアクセス」ACL を適用することもできますが、ビジネスニーズに合わせて権限が制限されているユーザーを含む ACL を作成することを強くお勧めします。

TLS が有効になっていないクラスターでは、「オープンアクセス」ACL を使用してオープン認証を行う必要があります。

ACL はユーザーなしで作成できます。空の ACL はクラスターにアクセスできず、TLS 有効なクラスターにのみ関連付けることができます。

ユーザーを作成するときは、最大 2 つのパスワードを設定できます。パスワードを変更しても、クラスターへの既存の接続はすべて維持されます。

特に、MemoryDBでACLを使用する場合は、ユーザーパスワードの制約に注意してください：

- パスワードは、印刷可能な 16～128 文字にする必要があります。
- 次の英数字以外の文字は使用できません: , `""` / @。

コンソールおよび CLI を使用したユーザーの管理

ユーザーの作成 (コンソール)

コンソールでユーザーを作成するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 左のナビゲーションペインで、[ユーザー] を選択します。
3. [ユーザーの作成] を選択します。
4. [ユーザーの作成] ページで [名前] を入力します。

   クラスターの命名に関する制約は次のとおりです。

   - 1 ~ 40 個の英数字またはハイフンを使用する必要があります。
   - 先頭は文字を使用する必要があります。
   - 連続する 2 つのハイフンを含めることはできません。
   - ハイフンで終わることはできません。
5. [パスワード] には、最大 2 つのパスワードを入力できます。
6. [アクセス文字列] にアクセス文字列を入力します。アクセス文字列は、ユーザーが許可されたキーとコマンドのアクセス許可レベルを設定します。
7. [タグ] の場合、オプションでタグを適用してユーザーを検索して絞り込んだり、AWS コストを追跡したりできます。
8. 作成を選択します。

AWS CLI を使用してユーザーを作成します

CLI を使用してユーザーを作成するには

- ユーザーを作成するには、create-userコマンドを使用します。

  Linux、macOS、Unix の場合:

```
aws memorydb create-user \
  --user-name user-name-1 \
  --access-string "~objects:* ~items:* ~public:*" \
  --authentication-mode \
        Passwords="abc",Type=password
```

Windows の場合:

```
aws memorydb create-user ^
  --user-name user-name-1 ^
  --access-string "~objects:* ~items:* ~public:*" ^
  --authentication-mode \
        Passwords="abc",Type=password
```

ユーザーの変更 (コンソール)

コンソールでユーザーに変更を加えるには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 左のナビゲーションペインで、[ユーザー] を選択します。
3. 変更するユーザーの横にあるラジオボタンを選択し、[アクション] ->[変更] を選択します。
4. パスワードを変更する場合は、[パスワードの変更] ラジオボタンを選択します。パスワードが 2 つある場合は、どちらか一方を変更するときに両方を入力する必要があることに注意してください。
5. アクセス文字列を更新する場合は、新しい文字列を入力します。
6. 変更を選択します。

AWS CLI を使用してユーザーを変更します

CLI を使用してユーザーを変更するには

1. 「ユーザーの更新」コマンドを使用してユーザーを変更します。
2. ユーザーが変更されると、そのユーザーに関連付けられたアクセスコントロールリストが、ACL に関連付けられたクラスターとともに更新されます。既存の接続はすべて維持されます。以下は例です。

   Linux、macOS、Unix の場合:

```
aws memorydb update-user \
```

```
  --user-name user-name-1 \
  --access-string "~objects:* ~items:* ~public:*"
```

Windows の場合:

```
aws memorydb update-user ^
  --user-name user-name-1 ^
  --access-string "~objects:* ~items:* ~public:*"
```

ユーザーの詳細を表示する (コンソール)

コンソールでユーザーの詳細を表示するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 左のナビゲーションペインで、[ユーザー] を選択します。
3. [ユーザー名] でユーザーを選択するか、検索ボックスを使用してユーザーを検索します。
4. [ユーザー設定] で、ユーザーのアクセス文字列、パスワード数、ステータス、Amazon リソースネーム (ARN) を確認できます。
5. [アクセスコントロールリスト (ACL)] では、ユーザーが所属する ACL を確認できます。
6. [タグ] では、ユーザーに関連付けられているすべてのタグを確認できます。

AWS CLI を使用してユーザーの詳細を表示します

「ユーザーの詳細」コマンドを使用して、ユーザーの詳細を表示します。

```
aws memorydb describe-users \
  --user-name my-user-name
```

ユーザーの削除 (コンソール)

コンソールでユーザーを削除するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。

2. 左のナビゲーションペインで、[ユーザー] を選択します。
3. 変更するユーザーの横にあるラジオボタンを選択し、[アクション]->[削除] を選択します。
4. 確認テキストボックスで、delete を入力し、確認 を選択します。
5. キャンセルするには、キャンセル をクリックします。

AWS CLI を使用してユーザーを削除する

CLI を使用してサービスロールを削除するには

- ユーザーを削除するには、delete-user コマンドを使用します。

  アカウントが削除され、そのアカウントが属するアクセス制御リストから削除されます。以下に例を示します。

  Linux、macOS、Unix の場合:

```
aws memorydb delete-user \
  --user-name user-name-2
```

  Windows の場合:

```
aws memorydb delete-user ^
  --user-name user-name-2
```

コンソールおよび CLI を使用したアクセスコントロールリストの管理

次に示すように、アクセスコントロールリストを作成して、1 つ以上のクラスターに対するユーザーのアクセスを分類および制御できます。

次の手順に従って、コンソールを使用してアクセス制御リストを管理します。

アクセスコントロールリスト (ACL) の作成 (コンソール)

コンソールを使用してアクセスコントロールリストを作成するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 左側のナビゲーションペインで、[アクセスコントロールリスト (ACL)] を選択します。

3. [ACL を作成] を選択します。
4. [アクセスコントロールリスト (ACL) の作成] ページで、ACL 名を入力します。

   クラスターの命名に関する制約は次のとおりです。

   - 1～40 個の英数字またはハイフンを使用する必要があります。
   - 先頭は文字を使用する必要があります。
   - 連続する 2 つのハイフンを含めることはできません。
   - ハイフンで終わることはできません。
5. [ユースケースを選択する] で、次のいずれかを実行します。

   a. [ユーザーの作成] を選択して新規ユーザーを作成します。

   b. ユーザーを追加するには、[管理] を選択し、[ユーザーの管理] ダイアログからユーザーを選択し、[選択] を選択します。
6. [タグ] については、オプションでタグを適用して ACL を検索およびフィルタリングしたり、AWS コストを追跡したりできます。
7. 作成 を選択します。

AWS CLI を使用してアクセスコントロールリスト (ACL) を作成します

次の手順で、CLI を使用してアクセスコントロールリストを作成します。

CLI を使用して新しいACLを作成し、ユーザーを追加するには

- create-acl コマンドを使用して ACL を作成します。

  Linux、macOS、Unix の場合:

```
aws memorydb create-acl \
  --acl-name "new-acl-1" \
  --user-names "user-name-1" "user-name-2"
```

  Windows の場合:

```
aws memorydb create-acl ^
  --acl-name "new-acl-1" ^
  --user-names "user-name-1" "user-name-2"
```

アクセスコントロールリスト (ACL) の変更 (コンソール)

コンソールを使用してアクセスコントロールリストを変更するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 左側のナビゲーションペインで、[アクセスコントロールリスト (ACL)] を選択します。
3. 変更する ACL を選択し、[変更] をクリックします。
4. [変更] ページの [選択したユーザー] で、次のいずれかを実行します。

   a. [ユーザーの作成] を選択して新しいユーザーを作成し、ACL に追加します。

   b. ユーザーを追加または削除するには、[管理] を選択し、[ユーザーの管理] ダイアログでユーザーを選択または選択解除し、[選択] を選択します。
5. [アクセスコントロールリスト (ACL) の作成] ページで、ACL 名を入力します。

   クラスターの命名に関する制約は次のとおりです。

   - 1 ~ 40 個の英数字またはハイフンを使用する必要があります。
   - 先頭は文字を使用する必要があります。
   - 連続する 2 つのハイフンを含めることはできません。
   - ハイフンで終わることはできません。
6. [ユースケースを選択する] で、次のいずれかを実行します。

   a. [ユーザーの作成] を選択して新規ユーザーを作成します。

   b. ユーザーを追加するには、[管理] を選択し、[ユーザーの管理] ダイアログからユーザーを選択し、[選択] を選択します。
7. [変更] を選択して変更を保存するか、[キャンセル] を選択して変更を破棄します。

AWS CLI を使用してアクセスコントロールリスト (ACL) を変更します

CLI を使用して新しいユーザーを追加するか、現在のメンバーを削除してACLを変更するには

- 「ACL の更新」コマンドを使用して ACL を変更します。

  Linux、macOS、Unix の場合:

```
aws memorydb update-acl --acl-name new-acl-1 \
```

```
--user-names-to-add user-name-3 \
--user-names-to-remove user-name-2
```

Windows の場合:

```
aws memorydb update-acl --acl-name new-acl-1 ^
--user-names-to-add user-name-3 ^
--user-names-to-remove user-name-2
```

> **Note**
>
> ACLから削除されたユーザーに属するオープン接続はすべて、このコマンドによって終了されます。

アクセスコントロールリスト (ACL) の詳細の表示 (コンソール)

ACL の詳細をコンソールに表示するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 左のナビゲーションペインで、[アクセスコントロールリスト (ACL)] をクリックします。
3. [ACL 名] の下にある ACL を選択するか、検索ボックスを使用して ACL を検索します。
4. [ユーザー] で、ACL に関連付けられているユーザーのリストを確認できます。
5. [関連クラスター] で、ACL が属するクラスターを確認できます。
6. [タグ] では、ACL に関連付けられたすべてのタグを確認できます。

AWS CLI を使用してアクセスコントロールリスト (ACL) を表示します

「ACL の詳細」コマンドを使用して ACL の詳細を表示します。

```
aws memorydb describe-acls \
  --acl-name test-group
```

アクセスコントロールリスト (ACL) の削除 (コンソール)

コンソールを使用してアクセスコントロールリストを削除するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 左側のナビゲーションペインで、[アクセスコントロールリスト (ACL)] を選択します。
3. 変更する ACL を選択し、[削除] を選択します
4. ACL が削除されないようにするには、[削除] ページで確認ボックスに `delete` を入力し、[削除] または [キャンセル] を選択します。

グループに属するユーザーではなく、ACL自体が削除されます。

AWS CLI を使用してアクセスコントロールリスト (ACL) を削除します

CLI を使用してACLを削除するには

- delete- acl コマンドを使用してACLを削除します。

  Linux、macOS、Unix の場合:

  ```
  aws memorydb delete-acl /
     --acl-name
  ```

  Windows の場合:

  ```
  aws memorydb delete-acl ^
     --acl-name
  ```

  上記の例では、次の応答を返します。

  ```
  aws memorydb delete-acl --acl-name "new-acl-1"
  {
      "ACLName": "new-acl-1",
      "Status": "deleting",
      "EngineVersion": "6.2",
      "UserNames": [
          "user-name-1",
          "user-name-3"
      ],
  ```

```
    "clusters": [],
    "ARN":"arn:aws:memorydb:us-east-1:493071037918:acl/new-acl-1"
}
```

アクセスコントロールリストのクラスターへの割り当て

ACL を作成してユーザーを追加した後、ACL を実装する最後の手順では、ACL をクラスターに割り当てます。

コンソールを使用してクラスターにアクセスコントロールリストを割り当てます

AWS マネジメントコンソール を使用してクラスターに ACL を追加する方法については、「MemoryDB クラスターの作成」を参照してください。

AWS CLI を使用してクラスターへのアクセスコントロールリストを割り当てます

以下の AWS CLI オペレーションでは、転送時の暗号化 (TLS) が有効で、値 *my-acl-name* を持つ acl-name パラメータを使用してクラスターを作成します。 サブネット グループ subnet-group を、実存のサブネットグループに置き換えます。

主要パラメータ

- **`--engine-version`** -「6.2」を指定してください
- **`--tls-enabled`** – 認証と ACL の関連付けに使用されます。
- **`--acl-name`** — この値は、クラスターに対して指定されたアクセス権限を持つユーザーで構成されるアクセス制御リストを提供します。

Linux、macOS、Unix の場合:

```
aws memorydb create-cluster \
    --cluster-name "new-cluster" \
    --description "new-cluster" \
    --engine-version "6.2" \
    --node-type db.r6g.large \
    --tls-enabled \
    --acl-name "new-acl-1" \
    --subnet-group-name "subnet-group"
```

Windows の場合:

```
aws memorydb create-cluster ^
    --cluster-name "new-cluster" ^
    --cluster-description "new-cluster" ^
    --engine-version "6.2" ^
    --node-type db.r6g.large ^
    --tls-enabled ^
    --acl-name "new-acl-1" ^
    --subnet-group-name "subnet-group"
```

以下の AWS CLI オペレーションでは、転送時の暗号化 (TLS) が有効で、値 new-acl-2 を持つ acl-name パラメータを使用してクラスターを変更します。

Linux、macOS、Unix の場合:

```
aws memorydb update-cluster \
    --cluster-name cluster-1 \
    --acl-name "new-acl-2"
```

Windows の場合:

```
aws memorydb update-cluster ^
    --cluster-name cluster-1 ^
    --acl-name "new-acl-2"
```

IAM を使用した認証

トピック

- 概要
- 制限
- セットアップ
- 接続中

概要

クラスターが Valkey または Redis OSS バージョン 7 以上を使用するように設定されている場合、IAM 認証では、AWS IAM アイデンティティを使用して MemoryDB への接続を認証できます。これにより、セキュリティモデルを強化し、多くの管理セキュリティタスクを簡素化できます。IAM

認証では、個々の MemoryDB クラスターと MemoryDB ユーザーごとにきめ細かいアクセス制御を設定し、最小特権の権限の原則に従うことができます。MemoryDB の IAM 認証は、有効期間の長い MemoryDB ユーザーパスワードの代わりに、有効期間が短い IAM 認証トークンを AUTH または HELLO コマンドで提供することにより機能します。IAM 認証トークンの詳細については、AWS 全般リファレンスガイドの「Signature Version 4 signing process」(署名バージョン 4 の署名プロセス)と、以下のコード例を参照してください。

IAM アイデンティティとそれに関連するポリシーを使用して、Valkey または Redis OSS アクセスをさらに制限できます。また、フェデレーテッド ID プロバイダーのユーザーに MemoryDB クラスターへのアクセス権を直接付与することもできます。

MemoryDB for Redis でAWS IAM を使用するには、まず認証モードを IAM に設定して MemoryDB ユーザーを作成する必要があり、その後で IAM アイデンティティを作成または再利用できます。IAM アイデンティティには、MemoryDB クラスターと MemoryDB ユーザーに `memorydb:Connect` アクションを許可するための関連ポリシーが必要です。設定が完了すると、IAM ユーザーまたはロールの AWS 認証情報を使用して IAM 認証トークンを作成できます。最後に、MemoryDB クラスターノードに接続するときに、有効期間が短い IAM 認証トークンを Valkey または Redis OSS クライアントのパスワードとして指定する必要があります。認証情報プロバイダーをサポートしているクライアントは、新しい接続ごとに一時的な認証情報を自動的に生成できます。MemoryDB for Redis は、IAM が有効な MemoryDB ユーザーの接続リクエストに対して IAM 認証を実行し、その接続リクエストを IAM で検証します。

制限

IAM 認証を使用する場合、以下の制限が適用されます。

- Valkey または Redis OSS エンジンバージョン 7.0 以上を使用している場合、IAM 認証が利用できます。
- IAM 認証トークンは 15 分間有効です。長時間接続する場合は、認証情報プロバイダーインターフェイスをサポートする Redis OSS クライアントを使用することをお勧めします。
- MemoryDB for Redis への IAM 認証された接続は、12 時間後に自動的に切断されます。新しい IAM 認証トークンを使用してAUTH または HELLO コマンドを送信することで、接続を 12 時間延長できます。
- IAM 認証は `MULTI EXEC` コマンドではサポートされていません。
- 現在、IAM 認証はすべてのグローバル条件コンテキストキーをサポートしていません。グローバル条件コンテキストキーの詳細については、「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。

セットアップ

IAM 認証をセットアップするには:

1. クラスターを作成する

```
aws memorydb create-cluster \
    --cluster-name cluster-01 \
    --description "MemoryDB IAM auth application"
    --node-type db.r6g.large \
    --engine-version 7.0 \
    --acl-name open-access
```

2. アカウントが新しいロールを引き継ぐことを許可するロール用の IAM 信頼ポリシードキュメントを以下に示すように作成します。ポリシーを trust-policy.json というファイルに保存します。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
        "Action": "sts:AssumeRole"
    }
}
```

3. 以下に示すように、IAM ポリシードキュメントを作成します。ポリシーを policy.json というファイルに保存します。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect" : "Allow",
      "Action" : [
        "memorydb:connect"
      ],
      "Resource" : [
```

```
        "arn:aws:memorydb:us-east-1:123456789012:cluster/cluster-01",
        "arn:aws:memorydb:us-east-1:123456789012:user/iam-user-01"
      ]
    }
  ]
}
```

4. IAM ロールを作成します。

```
aws iam create-role \
  --role-name "memorydb-iam-auth-app" \
  --assume-role-policy-document file://trust-policy.json
```

5. IAM ポリシーを作成します。

```
aws iam create-policy \
  --policy-name "memorydb-allow-all" \
  --policy-document file://policy.json
```

6. IAM ポリシーをロールにアタッチします。

```
aws iam attach-role-policy \
 --role-name "memorydb-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"
```

7. IAM を有効にしている新しいユーザーを作成します。

```
aws memorydb create-user \
  --user-name iam-user-01 \
  --authentication-mode Type=iam \
  --access-string "on ~* +@all"
```

8. ACL を作成し、ユーザーをアタッチします。

```
aws memorydb create-acl \
  --acl-name iam-acl-01 \
  --user-names iam-user-01

aws memorydb update-cluster \
  --cluster-name cluster-01 \
  --acl-name iam-acl-01
```

接続中

トークンをパスワードとして接続

最初に、AWS SigV4 の署名済みリクエストを使用して、有効期間が短い IAM 認証トークンを生成する必要があります。その後、以下の例に示すように、MemoryDB クラスターに接続するときに IAM 認証トークンをパスワードとして指定します。

```
String userName = "insert user name"
String clusterName = "insert cluster name"
String region = "insert region"

// Create a default AWS Credentials provider.
// This will look for AWS credentials defined in environment variables or system
 properties.
AWSCredentialsProvider awsCredentialsProvider = new
 DefaultAWSCredentialsProviderChain();

// Create an IAM authentication token request and signed it using the AWS credentials.
// The pre-signed request URL is used as an IAM authentication token for MemoryDB.
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userName,
 clusterName, region);
String iamAuthToken =
 iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());

// Construct URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(userName, iamAuthToken)
    .build();

// Create a new Lettuce client
RedisClusterClient client = RedisClusterClient.create(redisURI);
client.connect();
```

以下は IAMAuthTokenRequest の定義です。

```
public class IAMAuthTokenRequest {
    private static final HttpMethodName REQUEST_METHOD = HttpMethodName.GET;
    private static final String REQUEST_PROTOCOL = "http://";
    private static final String PARAM_ACTION = "Action";
```

```
    private static final String PARAM_USER = "User";
    private static final String ACTION_NAME = "connect";
    private static final String SERVICE_NAME = "memorydb";
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final String userName;
    private final String clusterName;
    private final String region;

    public IAMAuthTokenRequest(String userName, String clusterName, String region) {
        this.userName = userName;
        this.clusterName = clusterName;
        this.region = region;
    }

    public String toSignedRequestUri(AWSCredentials credentials) throws
URISyntaxException {
        Request<Void> request = getSignableRequest();
        sign(request, credentials);
        return new URIBuilder(request.getEndpoint())
            .addParameters(toNamedValuePair(request.getParameters()))
            .build()
            .toString()
            .replace(REQUEST_PROTOCOL, "");
    }

    private <T> Request<T> getSignableRequest() {
        Request<T> request  = new DefaultRequest<>(SERVICE_NAME);
        request.setHttpMethod(REQUEST_METHOD);
        request.setEndpoint(getRequestUri());
        request.addParameters(PARAM_ACTION, Collections.singletonList(ACTION_NAME));
        request.addParameters(PARAM_USER, Collections.singletonList(userName));
        return request;
    }

    private URI getRequestUri() {
        return URI.create(String.format("%s%s/", REQUEST_PROTOCOL, clusterName));
    }

    private <T> void sign(SignableRequest<T> request, AWSCredentials credentials) {
        AWS4Signer signer = new AWS4Signer();
        signer.setRegionName(region);
        signer.setServiceName(SERVICE_NAME);
```

```
        DateTime dateTime = DateTime.now();
        dateTime = dateTime.plus(Duration.standardSeconds(TOKEN_EXPIRY_SECONDS));

        signer.presignRequest(request, credentials, dateTime.toDate());
    }

    private static List<NameValuePair> toNamedValuePair(Map<String, List<String>> in) {
        return in.entrySet().stream()
            .map(e -> new BasicNameValuePair(e.getKey(), e.getValue().get(0)))
            .collect(Collectors.toList());
    }
}
```

認証情報プロバイダーに接続

以下のコードは、IAM 認証情報プロバイダーを使用して MemoryDB for Redis で認証する方法を示しています。

```
String userName = "insert user name"
String clusterName = "insert cluster name"
String region = "insert region"

// Create a default AWS Credentials provider.
// This will look for AWS credentials defined in environment variables or system
 properties.
AWSCredentialsProvider awsCredentialsProvider = new
 DefaultAWSCredentialsProviderChain();

// Create an IAM authentication token request. Once this request is signed it can be
 used as an
// IAM authentication token for MemoryDB.
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userName,
 clusterName, region);

// Create a credentials provider using IAM credentials.
RedisCredentialsProvider redisCredentialsProvider = new
 RedisIAMAuthCredentialsProvider(
    userName, iamAuthTokenRequest, awsCredentialsProvider);

// Construct URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
```

```
    .withSsl(ssl)
    .withAuthentication(redisCredentialsProvider)
    .build();

// Create a new Lettuce cluster client
RedisClusterClient client = RedisClusterClient.create(redisURI);
client.connect();
```

以下は、IAMAuthTokenRequest を認証情報プロバイダーにラップして、必要に応じて一時的な認証情報を自動生成する Lettuce クラスタークライアントの例です。

```
public class RedisIAMAuthCredentialsProvider implements RedisCredentialsProvider {
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final AWSCredentialsProvider awsCredentialsProvider;
    private final String userName;
    private final IAMAuthTokenRequest iamAuthTokenRequest;
    private final Supplier<String> iamAuthTokenSupplier;

    public RedisIAMAuthCredentialsProvider(String userName,
        IAMAuthTokenRequest iamAuthTokenRequest,
        AWSCredentialsProvider awsCredentialsProvider) {
        this.userName = userName;
        this.awsCredentialsProvider = awsCredentialsProvider;
        this.iamAuthTokenRequest = iamAuthTokenRequest;
        this.iamAuthTokenSupplier =
 Suppliers.memoizeWithExpiration(this::getIamAuthToken, TOKEN_EXPIRY_SECONDS,
 TimeUnit.SECONDS);
    }

    @Override
    public Mono<RedisCredentials> resolveCredentials() {
        return Mono.just(RedisCredentials.just(userName, iamAuthTokenSupplier.get()));
    }

    private String getIamAuthToken() {
        return
 iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());
    }
```

MemoryDB でのアイデンティティとアクセス権の管理

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するために役立つ AWS のサービスです。IAM 管理者は、どのユーザーが MemoryDB リソースの使用を認証 (サインイン) および承認 (権限を持たせる) されるかを制御します。IAM は、追加費用なしで使用できる AWS のサービス です。

トピック

- 対象者
- アイデンティティによる認証
- ポリシーを使用したアクセス権の管理
- MemoryDB と IAM の連携の仕組み
- MemoryDB のアイデンティティベースのポリシーの例
- MemoryDB のアイデンティティとアクセスの問題のトラブルシューティング
- アクセスコントロール
- MemoryDB リソースに対する許可の管理の概要

対象者

AWS Identity and Access Management (IAM) の使用方法は、ロールによって異なります。

- サービスユーザー - 機能にアクセスできない場合は、管理者に許可をリクエストします (「MemoryDB のアイデンティティとアクセスの問題のトラブルシューティング」を参照)
- サービス管理者 - ユーザーアクセスを決定し、アクセス許可リクエストを送信します (「MemoryDB と IAM の連携の仕組み」を参照)
- IAM 管理者 - アクセスを管理するポリシーを記述します (「MemoryDB のアイデンティティベースのポリシーの例」を参照)

アイデンティティによる認証

認証とは、アイデンティティ認証情報を使用して AWS にサインインする方法です。ユーザーは、AWS アカウントのルートユーザー、IAM ユーザーとして、または IAM ロールを引き受けることによって、認証される必要があります。

AWS IAM Identity Center (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーテッドアイデンティティとしてサインイ

ンできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「How to sign in to your AWS アカウント」を参照してください。

プログラムによるアクセスの場合、AWS はリクエストに暗号で署名するための SDK と CLI を提供します。詳細については、「IAM ユーザーガイド」の「API リクエストに対する AWS Signature Version 4」を参照してください。

AWS アカウント のルートユーザー

AWS アカウントを作成すると、すべての AWS のサービスとリソースに対する完全なアクセス権を持つ AWS アカウントルートユーザーと呼ばれる 1 つのサインイン ID を使用して開始します。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。ルートユーザー認証情報を必要とするタスクについては、「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。

フェデレーテッドアイデンティティ

ベストプラクティスでは、人間のユーザーが一時的な認証情報を使用して AWS のサービス にアクセスする際、アイデンティティプロバイダーとのフェデレーションを使用することが求められます。

フェデレーテッドアイデンティティは、エンタープライズディレクトリ、ウェブ ID プロバイダー、Directory Service のユーザーであり、ID ソースからの認証情報を使用して AWS のサービスにアクセスするユーザーです。フェデレーティッドアイデンティティは、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、AWS IAM Identity Center 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「What is IAM Identity Center?」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、1 人のユーザーまたは 1 つのアプリケーションに対して特定の許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用することをお勧めします。詳細は「IAM ユーザーガイド」の「人間のユーザーが一時的な認証情報を使用して AWS にアクセスするには ID プロバイダーとのフェデレーションの使用が必要です」を参照してください。

IAM グループは IAM ユーザーのコレクションを指定し、大量のユーザーのアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、一時的な認証情報を提供する特定のアクセス許可を持つ ID です。ユーザーから IAM ロールに切り替える (コンソール)、または AWS CLI や AWS API オペレーションを呼び出すことで、ロールを引き受けることができます。詳細については、「IAM ユーザーガイド」の「ロールを引き受けることができない」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行されているアプリケーションに役立ちます。詳細については、「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。

ポリシーを使用したアクセス権の管理

AWS でアクセスを制御するには、ポリシーを作成して 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やリソースに関連付けて、これらのアクセス許可を定義します。AWS は、プリンシパルがリクエストを行う際に、それらのポリシーを評価します。大半のポリシーは JSON ドキュメントとして AWS に保存されます。JSON ポリシードキュメントの詳細については、「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。

ポリシーを使用して、管理者は、どのプリンシパルがどのリソースに対してどの条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

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 IAM ポリシーを作成し、ユーザーが引き受けることができるロールに追加します。IAM ポリシーは、オペレーションの実行方法を問わずアクセス許可を定義します。

アイデンティティベースポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、グループ、ロール) にアタッチする JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティが実行できるアクション、リソース、および条件を制御します。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

ID ベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーおよび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「管理ポリシーとインラインポリシーのいずれかを選択する」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、リソースにアタッチする JSON ポリシードキュメントです。例としては、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーなどがあります。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。リソースベースのポリシーで、プリンシパルを指定する必要があります。

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。

その他のポリシータイプ

AWS 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る、追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできる許可の上限を設定します。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 (SCP) – AWS Organizations において組織または組織単位のアクセス許可の上限を指定します。詳細については、AWS Organizations ユーザーガイドの「サービスコントロールポリシー」を参照してください。
- リソースコントロールポリシー (RCP) – アカウント内のリソースで利用できるアクセス許可の上限を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを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。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連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、「IAM ユーザーガイド」の「ポリシーの評価ロジック」を参照してください。

MemoryDB と IAM の連携の仕組み

IAM を使用して MemoryDB へのアクセスを管理する前に、MemoryDB で利用できる IAM の機能について学びます。

MemoryDB で使用できる IAM の機能

IAM の機能	MemoryDB サポート
アイデンティティベースポリシー	あり
リソースベースのポリシー	なし
ポリシーアクション	あり
ポリシーリソース	あり
ポリシー条件キー	あり
ACL	はい
ABAC (ポリシー内のタグ)	あり
一時的な認証情報	あり
プリンシパルアクセス許可	あり
サービスロール	あり
サービスリンクロール	はい

MemoryDB およびその他の AWS サービスと多くの IAM 機能の連携についての概要は、「IAM ユーザーガイド」の「IAM と連携する AWS サービス」を参照してください。

MemoryDB の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のポリシーのサポート: あり

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 許可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ポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。JSON ポリシーで使用できるすべての要素について学ぶには、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

MemoryDB のアイデンティティベースのポリシーの例

MemoryDB 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、「MemoryDB のアイデンティティベースのポリシーの例」を参照してください。

MemoryDB 内のリソースベースのポリシー

リソースベースのポリシーのサポート: なし

リソースベースのポリシーは、リソースにアタッチする 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、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、フェデレーションユーザー、または AWS のサービス を含めることができます。

クロスアカウントアクセスを有効にするには、全体のアカウント、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。詳細については、「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。

MemoryDB のポリシーアクション

ポリシーアクションのサポート:あり

管理者は AWS JSON ポリシーを使用して、だれ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。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アクセス許可を付与するポリシーで使用されます。

MemoryDB アクションのリストを確認するには、「サービス認可リファレンス」の「Actions Defined by MemoryDB」を参照してください。

MemoryDB のポリシーアクションは、アクションの前に以下のプレフィックスを使用します。

```
MemoryDB
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "MemoryDB:action1",
      "MemoryDB:action2"
         ]
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、Describe という単語で始まるすべてのアクションを指定するには次のアクションを含めます。

```
"Action": "MemoryDB:Describe*"
```

MemoryDB 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、「MemoryDB のアイデンティティベースのポリシーの例」を参照してください。

MemoryDB のポリシーリソース

ポリシーリソースのサポート: あり

管理者は AWS JSON ポリシーを使用して、だれ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。

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。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。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": "*"
```

MemoryDB のリソースタイプとその ARN のリストを確認するには、「サービス認可リファレンス」の「Resources Defined by MemoryDB」。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、「Actions Defined by MemoryDB」を参照してください。

MemoryDB 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、「MemoryDB のアイデンティティベースのポリシーの例」を参照してください。

MemoryDB のポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり

管理者は AWS JSON ポリシーを使用して、だれ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` 要素は、定義された基準に基づいてステートメントが実行されるタイミングを指定します。イコールや未満などの 条件演算子 を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。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。

MemoryDB 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、「MemoryDB のアイデンティティベースのポリシーの例」を参照してください。

条件キーの使用

IAM ポリシーを有効にする方法を決める条件を指定できます。MemoryDB では、JSON ポリシーの `Condition` 要素を使用して、リクエストコンテキストのキーを、ポリシーで指定したキー値と比較できます。ポリシー要素の詳細については、IAM JSON policy elements: Condition を参照してください。

MemoryDB の条件キーのリストを確認するには、「サービス認可リファレンス」の「MemoryDB の条件キー」を参照してください。

グローバル条件キーのリストについては、「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。

条件の指定: 条件キーの使用

きめ細かなコントロールを実装するには、特定のリクエストに対して個別のパラメータセットを制御するための条件を指定した IAM アクセス許可ポリシーを作成します。次に、IAM コンソールを使用して作成する IAM ユーザー、グループ、またはロールにそのポリシーを適用できます。

条件を適用するには、条件情報を IAM ポリシーステートメントに追加します。例えば、TLS が無効になっている MemoryDB クラスターの作成を禁止するには、ポリシーステートメントで次の条件を指定できます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "memorydb:CreateCluster"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "Bool": {
          "memorydb:TLSEnabled": "false"
        }
      }
    }
  ]
}
```

タグ付けの詳細については、「MemoryDB リソースのタグ付け」を参照してください。

ポリシー条件演算子の使用に関する詳細については、「MemoryDB API の許可: アクション、リソース、条件リファレンス」を参照してください。

ポリシー例: きめ細かなパラーメータコントロールのための IAM ポリシー条件の使用

このセクションでは、前述の MemoryDB パラメータに対してきめ細かなアクセスコントロールを実装するためのポリシー例について説明します。

1. memorydb:TLSEnabled - TLS を有効にした場合にのみクラスターを作成するように指定します。

   JSON

   ```
   {
   ```

```
    "Version":"2012-10-17",
    "Statement": [
              {
            "Effect": "Allow",
            "Action": [
                "memorydb:CreateCluster"
            ],
            "Resource": [
                "arn:aws:memorydb:*:*:parametergroup/*",
                "arn:aws:memorydb:*:*:subnetgroup/*",
                "arn:aws:memorydb:*:*:acl/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "memorydb:CreateCluster"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "Bool": {
                    "memorydb:TLSEnabled": "true"
                }
            }
        }
    ]
}
```

2. memorydb:UserAuthenticationMode: - 特定のタイプ認証モード (IAM など) でユーザーを作成できるように指定します。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "memorydb:Createuser"
            ],
            "Resource": [
```

```
                "arn:aws:memorydb:*:*:user/*"
            ],
            "Condition": {
                "StringEquals": {
                    "memorydb:UserAuthenticationMode": "iam"
                }
            }
        }
    ]
}
```

「Deny」ベースのポリシーを設定する場合は、StringEqualsIgnoreCase 演算子を使用して、ケースに関係なく、特定のユーザー認証モードタイプのすべての呼び出しを回避することをお勧めします。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "memorydb:CreateUser"
      ],
      "Resource": "*",
      "Condition": {
        "StringEqualsIgnoreCase": {
          "memorydb:UserAuthenticationMode": "password"
        }
      }
    }
  ]
}
```

MemoryDB のアクセスコントロールリスト (ACL)

ACL のサポート: あり

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。

MemoryDB での属性ベースのアクセスコントロール (ABAC)

ABAC (ポリシー内のタグ) のサポート: あり

属性ベースのアクセス制御 (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。IAM エンティティと AWS リソースにタグを付けることで、プリンシパルのタグがリソースタグと一致するときに操作を許可する ABAC ポリシーを設計できます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

MemoryDB での一時的な認証情報の使用

一時的な認証情報のサポート: あり

一時的な認証情報は、AWS リソースへの短期的なアクセスを提供し、フェデレーションまたはスイッチロールの使用時に自動的に作成されます。AWS では、長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成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「IAM の一時的な認証情報」および「IAM と連携する AWS のサービス」を参照してください。

MemoryDB のクロスサービスプリンシパル許可

転送アクセスセッション (FAS) のサポート: あり

転送アクセスセッション (FAS) は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行いま

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。

MemoryDB のサービスロール

サービスロールのサポート: あり

サービスロールとは、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 に許可を委任するロールを作成する」を参照してください。

> **Warning**
>
> サービスロールの許可を変更すると、MemoryDB の機能が破損する可能性があります。MemoryDB が指示する場合以外は、サービスロールを編集しないでください。

MemoryDB 用のサービスリンクロール

サービスリンクロールのサポート: あり

サービスにリンクされたロールは、AWS のサービスにリンクされているサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスにリンクされたロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

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、「IAM と提携する AWS のサービス」を参照してください。表の「サービスリンクロール」列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

MemoryDB の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、ユーザーおよびロールには、MemoryDB または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。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。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーを作成する (コンソール)」を参照してください。

MemoryDB が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、「サービス認可リファレンス」の「MemoryDB のアクション、リソース、および条件キー」を参照してください。

トピック

- ポリシーに関するベストプラクティス
- MemoryDB コンソールの使用
- 自分の権限の表示をユーザーに許可する

ポリシーに関するベストプラクティス

ID ベースのポリシーは、ユーザーのアカウントで誰かが MemoryDB 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では、AWS アカウントに費用が発生する場合があります。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 マネージドポリシーを使用して開始し、最小特権の許可に移行する – ユーザーとワークロードへの許可の付与を開始するには、多くの一般的なユースケースのために許可を付与する AWS マネージドポリシーを使用します。これらは AWS アカウントで使用できます。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、アクセス許可を絞り込むことをお勧めします。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」または「ジョブ機能の AWS マネージドポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーとアクセス許可」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。また、CloudFormation などの特定の AWS のサービス を介して使用する場合、条件を使用して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。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素:条件」を参照してください。

- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer でポリシーを検証する」を参照してください。
- 多要素認証 (MFA) を要求する – AWS アカウントで IAM ユーザーまたはルートユーザーを要求するシナリオがある場合は、セキュリティを強化するために MFA をオンにします。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 を使用した安全な API アクセス」を参照してください。

IAM でのベストプラクティスの詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

MemoryDB コンソールの使用

MemoryDB コンソールにアクセスするには、最小限の許可セットが必要です。これらのアクセス許可により、AWS アカウントの MemoryDB リソースの詳細をリストおよび表示できます。最小限必要なアクセス許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) ではコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソール権限を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスを許可します。

ユーザーとロールが引き続き MemoryDB コンソールを使用できるようにするには、エンティティに MemoryDB `ConsoleAccess` または`ReadOnly` AWS マネージドポリシーもアタッチします。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

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 か AWS API を使用してプログラム的に、このアクションを完了する権限が含まれています。

```
{
```

```
    "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": "*"
        }
    ]
}
```

MemoryDB のアイデンティティとアクセスの問題のトラブルシューティング

次の情報は、MemoryDB と IAM を使用する際に発生する可能性がある一般的な問題の診断や修復に役立ちます。

トピック

- MemoryDB でアクションを実行する権限がない

- iam:PassRole を実行する権限がない
- 自分の AWS アカウント以外のユーザーに MemoryDB リソースへのアクセスを許可したい

MemoryDB でアクションを実行する権限がない

AWS マネジメントコンソール から、アクションを実行することが認可されていないと通知された場合、管理者に問い合わせ、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。

以下のエラー例は、mateojackson ユーザーがコンソールを使用して架空の *my-example-widget* リソースに関する詳細情報を表示しようとしているが、架空の MemoryDB:*GetWidget* 許可がないという場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 MemoryDB:GetWidget on resource: my-example-widget
```

この場合、Mateo は、MemoryDB:*GetWidget* アクションを使用して *my-example-widget* リソースにアクセスできるように、管理者にポリシーの更新を依頼します。

iam:PassRole を実行する権限がない

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 MemoryDB 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リンクロールを作成せずに、既存のロールをサービスに渡すことが許可されています。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 MemoryDB 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 管理者に問い合わせてください。サインイン認証情報を提供した担当者が管理者です。

自分の AWS アカウント以外のユーザーに MemoryDB 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。

- MemoryDB でこれらの機能がサポートされるかどうかを確認するには、「MemoryDB と IAM の連携の仕組み」を参照してください。
- 所有している AWS アカウント 全体のリソースへのアクセス権を提供する方法については、「IAM ユーザーガイド」の「所有している別の AWS アカウント へのアクセス権を IAM ユーザーに提供」を参照してください。
- サードパーティーの AWS アカウント にリソースへのアクセス権を提供する方法については、「IAM ユーザーガイド」の「サードパーティーが所有する AWS アカウント へのアクセス権を付与する」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「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

アクセスコントロール

有効な認証情報があればリクエストを認証できますが、アクセス許可が付与されている場合を除き、MemoryDB リソースの作成やアクセスはできません。例えば、MemoryDB クラスターを作成するためのアクセス権限が必要です。

次のセクションでは、MemoryDB の許可を管理する方法について説明します。最初に概要のセクションを読むことをお勧めします。

- MemoryDB リソースに対する許可の管理の概要

- MemoryDB でのアイデンティティベースのポリシー (IAM ポリシー) の使用

MemoryDB リソースに対する許可の管理の概要

すべての AWS リソースは AWS アカウントによって所有され、リソースの作成またはアクセスは、許可のポリシーによって管理されます。アカウント管理者は、IAM アイデンティティ (ユーザー、グループ、ロール) にアクセス許可ポリシーをアタッチできます。さらに、MemoryDB では、アクセス許可ポリシーをリソースにアタッチすることもできます。

> **Note**
>
> アカウント管理者 (または管理者ユーザー) は、管理者権限を持つユーザーです。詳細については、「IAM ユーザーガイド」の「IAM のベストプラクティス」を参照してください。

アクセス権限を付与するにはユーザー、グループ、またはロールにアクセス許可を追加します。

- AWS IAM Identity Center のユーザーとグループ:

  アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」の手順に従ってください。
- IAM 内で、ID プロバイダーによって管理されているユーザー:

  ID フェデレーションの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「サードパーティー ID プロバイダー (フェデレーション) 用のロールを作成する」を参照してください。
- IAM ユーザー:
  - ユーザーが担当できるロールを作成します。手順については「IAM ユーザーガイド」の「IAM ユーザーのロールの作成」を参照してください。
  - (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加します。詳細については「IAM ユーザーガイド」の「ユーザー (コンソール) へのアクセス権限の追加」を参照してください。

トピック

- MemoryDB リソースおよびオペレーション
- リソース所有権についての理解
- リソースへのアクセスの管理
- MemoryDB でのアイデンティティベースのポリシー (IAM ポリシー) の使用
- リソースレベルのアクセス許可

- MemoryDB 用のサービスリンクロール
- MemoryDB の AWS マネージドポリシー
- MemoryDB API の許可: アクション、リソース、条件リファレンス

MemoryDB リソースおよびオペレーション

MemoryDB では、プライマリリソースはクラスターです。

これらのリソースには、以下に示すとおり、一意の Amazon リソースネーム (ARN) が関連付けられています。

> **Note**
>
> リソースレベルのアクセス許可を有効にするには、ARN 文字列のリソース名を小文字にする必要があります。

リソースタイプ	ARN 形式
ユーザー	arn:aws:memorydb:*us-east-1:123456789012* :user/user1
アクセスコントロールリスト (ACL)	arn:aws:memorydb:*us-east-1:123456789012* :acl/myacl
クラスター	arn:aws:memorydb:*us-east-1:123456789012* :cluster/my-cluster
Snapshot	arn:aws:memorydb:*us-east-1:123456789012* :snapshot/my-snapshot
パラメータグループ	arn:aws:memorydb:*us-east-1:123456789012* :parametergroup/my-parameter-group
サブネットグループ	arn:aws:memorydb:*us-east-1:123456789012* :subnetgroup/my-subnet-group

MemoryDB では、MemoryDB リソースを操作する一連のオペレーションが用意されています。可能なオペレーションのリストについては、MemoryDB「アクション」を参照してください。

リソース所有権についての理解

リソース所有者は、リソースを作成した AWS アカウントです。つまり、リソース所有者は、そのリソースを作成するリクエストを認証するプリンシパルエンティティの AWS アカウントです。プリンシパルエンティティ はルートアカウント、IAM ユーザー、または IAM ロールです。次の例は、この仕組みを示しています。

- AWS アカウントのルートアカウント認証情報を使用してクラスターを作成するとします。この場合、AWS アカウントはリソースの所有者です。MemoryDB では、リソースはクラスターです。
- AWS アカウントに IAM ユーザーを作成し、クラスターを作成するためのアクセス許可をそのユーザーに付与するとします。この場合、ユーザーはクラスターを作成できます。ただし、ユーザーが属するアカウントが AWS クラスターリソースを所有します。
- クラスターを作成するためのアクセス許可のある AWS アカウントに IAM ロールを作成するとします。この場合、ロールを引き受けることができるいずれのユーザーもクラスターを作成できます。ロールが属するアカウントが AWS クラスターリソースを所有します。

リソースへのアクセスの管理

アクセス権限ポリシー では、誰が何にアクセスできるかを記述します。以下のセクションで、アクセス許可ポリシーを作成するために使用可能なオプションについて説明します。

> **Note**
>
> このセクションでは、MemoryDB のコンテキストでの IAM の使用について説明します。ここでは、IAM サービスに関する詳細情報を提供しません。完全な IAM ドキュメンテーションについては、「IAM ユーザーガイド」の「IAM とは」を参照してください。IAM ポリシー構文の詳細と説明については、IAM ユーザーガイドの AWS IAM ポリシーの参照を参照してください。

IAM アイデンティティにアタッチされているポリシーは、アイデンティティベースのポリシー (IAM ポリシー) と呼ばれます。リソースに添付されたポリシーは、[リソースベース]のポリシーと呼ばれます。

トピック

- アイデンティティベースのポリシー (IAM ポリシー)
- ポリシー要素の指定: アクション、効果、リソース、プリンシパル
- ポリシーでの条件の指定

アイデンティティベースのポリシー (IAM ポリシー)

ポリシーを IAM アイデンティティにアタッチできます。例えば、次のオペレーションを実行できます。

- アカウントのユーザーまたはグループにアクセス許可ポリシーをアタッチする – アカウント管理者は、特定のユーザーに関連付けられるアクセス許可ポリシーを使用して、アクセス許可を付与できます。この場合、アクセス許可は、そのユーザーがクラスター、パラメータグループ、セキュリティグループなどの MemoryDB リソースを作成するためのものです。
- 許可ポリシーをロールにアタッチする (クロスアカウントの許可を付与) - ID ベースのアクセス許可ポリシーを IAM ロールにアタッチして、クロスアカウントの許可を付与することができます。例えば、アカウント A の管理者は、次のように別の AWS アカウント (例えば、アカウント B) または AWS サービスにクロスアカウントアクセス許可を付与するロールを作成できます。
  1. アカウント A の管理者は、IAM ロールを作成して、アカウント A のリソースにアクセス許可を付与するロールにアクセス許可ポリシーをアタッチします。
  2. アカウント A の管理者は、アカウント B をそのロールを引き受けるプリンシパルとして識別するロールに、信頼ポリシーをアタッチします。
  3. アカウント B の管理者は、ロールを引き受けるためのアクセス許可をアカウント B のユーザーに委任できます。これにより、アカウント B のユーザーは、アカウント A のリソースの作成またはアクセスが許可されます。場合によっては、AWS のサービスにロールを引き受けるためのアクセス許可を付与する必要があります。このアプローチをサポートするために、信頼ポリシーのプリンシパルを AWS のサービスのプリンシパルにすることもできます。

  IAM を使用した許可の委任の詳細については、「IAM ユーザーガイド」の「アクセス管理」を参照してください。

以下に示しているのは、お客様の AWS アカウントに対する `DescribeClusters` アクションの実行をユーザーに許可するポリシーの例です。MemoryDB では、API アクションのリソース ARN を使用した特定のリソースの識別もサポートしています。(このアプローチは、リソースレベルのアクセス許可とも呼ばれます)。

でアイデンティティベースのポリシーを使用する場合の詳細については、MemoryDB「MemoryDB でのアイデンティティベースのポリシー (IAM ポリシー) の使用」を参照してください。ユーザー、グループ、ロール、アクセス許可の詳細については、IAM ユーザーガイドの「アイデンティティ (ユーザー、グループ、ロール)」を参照してください。

ポリシー要素の指定: アクション、効果、リソース、プリンシパル

サービスは、MemoryDB リソースごとに (MemoryDB リソースおよびオペレーションを参照)、一連の API オペレーションを定義します (アクションを参照)。こうした API オペレーションへの許可を付与するために、MemoryDB はポリシーに定義できる一連のアクションを定義します。例えば、MemoryDB クラスターリソースに対して、アクション `CreateCluster`、`DeleteCluster`、`DescribeClusters` を定義します。1 つの API オペレーションの実行で、複数のアクションのアクセス権限が必要になる場合があります。

最も基本的なポリシーの要素を次に示します。

- リソース– ポリシーで Amazon リソースネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。詳細については、「MemoryDB リソースおよびオペレーション」を参照してください。
- アクション – アクションキーワードを使用して、許可または拒否するリソース操作を特定します。例えば、指定した `Effect` に応じて、`memorydb:CreateCluster` アクセス権限では、MemoryDB `CreateCluster` オペレーションの実行をユーザーに許可または拒否します。
- 効果 – ユーザーが特定のアクションを要求する際の効果を指定します。許可または拒否のいずれかになります。リソースへのアクセスを明示的に付与 (許可) していない場合、アクセスは暗黙的に拒否されます。リソースへのアクセスを明示的に拒否することもできます。例えば、別のポリシーでリソースへのアクセスが許可されているユーザーに対して、そのリソースへのアクセスを禁止できます。
- プリンシパル - ID ベースのポリシー (IAM ポリシー) で、ポリシーがアタッチされているユーザーが黙示的なプリンシパルとなります。リソースベースのポリシーでは、アクセス許可 (リソースベースのポリシーにのみ適用) を受け取りたいユーザー、アカウント、サービス、またはその他のエンティティを指定します。

IAM ポリシーの構文と記述の詳細については、「IAM ユーザーガイド」の「AWS IAM ポリシーリファレンス」を参照してください。

すべての MemoryDB API アクションを示す表については、「MemoryDB API の許可: アクション、リソース、条件リファレンス」を参照してください。

ポリシーでの条件の指定

許可を付与するとき、IAM ポリシー言語を使用して、ポリシーが有効になる必要がある条件を指定できます。例えば、特定の日付の後にのみ適用されるポリシーが必要になる場合があります。ポリシー言語での条件の指定の詳細については、「IAM ユーザーガイド」の「条件」を参照してください。

MemoryDB でのアイデンティティベースのポリシー (IAM ポリシー) の使用

このトピックでは、アカウント管理者が IAM ID (ユーザー、グループ、ロール) へのアクセス許可ポリシーをアタッチする、ID ベースのポリシーの例を示します。

> **Important**
>
> 最初に、MemoryDB リソースへのアクセスを管理するための基本的な概念とオプションについて説明するトピックを読むことをお勧めします。詳細については、「MemoryDB リソースに対する許可の管理の概要」を参照してください。

このセクションでは、次のトピックを対象としています。

- MemoryDB コンソールの使用に必要なアクセス権限
- MemoryDB の AWS 管理 (事前定義) ポリシー
- カスタマーマネージドポリシーの例

以下に示しているのは、アクセス許可ポリシーの例です。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [{
       "Sid": "AllowClusterPermissions",
       "Effect": "Allow",
       "Action": [
          "memorydb:CreateCluster",
          "memorydb:DescribeClusters",
          "memorydb:UpdateCluster"],
       "Resource": "*"
       },
       {
         "Sid": "AllowUserToPassRole",
         "Effect": "Allow",
         "Action": [ "iam:PassRole" ],
         "Resource": "arn:aws:iam::123456789012:role/EC2-roles-for-cluster"
       }
```

```
    ]
}
```

このポリシーには以下の 2 つのステートメントがあります。

- 最初のステートメントは、アカウントが所有するクラスター上の MemoryDB アクション (`memorydb:CreateCluster`、`memorydb:DescribeClusters`、`memorydb:UpdateCluster`) のアクセス権限を付与します。
- 2 番目のステートメントは、`Resource` 値の最後に指定した IAM ロール名での IAM アクション`iam:PassRole`のアクセス許可を付与します。

ID ベースのポリシーでアクセス許可を得るプリンシパルを指定していないため、ポリシーでは `Principal` 要素を指定していません。ユーザーにポリシーをアタッチすると、そのユーザーが暗黙のプリンシパルになります。IAM ロールにアクセス権限ポリシーをアタッチすると、ロールの信頼ポリシーで識別されたプリンシパルがアクセス権限を得ることになります。

すべての MemoryDB API アクションとそれらが適用されるリソースの表については、「MemoryDB API の許可: アクション、リソース、条件リファレンス」を参照してください。

MemoryDB コンソールの使用に必要なアクセス権限

アクセス権限のリファレンス表では、MemoryDB API オペレーションとそれらの各オペレーションに必要なアクセス権限を示しています。MemoryDB API オペレーションの詳細については、「MemoryDB API の許可: アクション、リソース、条件リファレンス」を参照してください。

MemoryDB コンソールを使用するには、まず、以下のアクセス権限ポリシーに示しているように、追加のアクションのためのアクセス権限を付与します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "MinPermsForMemDBConsole",
        "Effect": "Allow",
        "Action": [
            "memorydb:Describe*",
            "memorydb:List*",
```

```
            "ec2:DescribeAvailabilityZones",
            "ec2:DescribeVpcs",
            "ec2:DescribeAccountAttributes",
            "ec2:DescribeSecurityGroups",
            "cloudwatch:GetMetricStatistics",
            "cloudwatch:DescribeAlarms",
            "s3:ListAllMyBuckets",
            "sns:ListTopics",
            "sns:ListSubscriptions" ],
        "Resource": "*"
        }
    ]
}
```

MemoryDB コンソールには、以下の理由でこれらの追加のアクセス権限が必要になります。

- MemoryDB アクションを実行するためのアクセス権限。コンソールで、アカウントの MemoryDB リソースを表示するために必要です。
- Amazon EC2 に対してクエリを行う `ec2` アクションを実行するためのアクセス権限。コンソールで、アベイラビリティーゾーン、VPC、セキュリティグループ、アカウント属性を表示するために必要です。
- `cloudwatch` アクションを実行するためのアクセス許可。コンソールで、Amazon CloudWatch メトリクスとアラームを取得し、表示するために必要です。
- `sns` アクションのアクセス許可を使用すると、Amazon Simple Notification Service (Amazon SNS) のトピックやサブスクリプションを取得し、コンソールにそれらを表示することができます。

カスタマーマネージドポリシーの例

デフォルトポリシーを使用せず、カスタムマネージドポリシーを使用することを選択した場合は、以下の 2 点のいずれかを確認してください。`iam:createServiceLinkedRole` を呼び出すためのアクセス許可があることが必要です (詳細については、「例 4: ユーザーが IAM CreateServiceLinkedRole API を呼び出すことを許可する」を参照)。または、MemoryDB サービスにリンクされたロールを作成済みであることが必要です。

MemoryDB コンソールを使用するために必要な最小限のアクセス権限と組み合わせて、このセクションでのポリシーの例は、追加のアクセス権限を付与します。この例は、AWS SDK と AWS CLI に関連しています。MemoryDB コンソールを使用するために必要なアクセス権限の詳細については、「MemoryDB コンソールの使用に必要なアクセス権限」を参照してください。

IAM ユーザーおよびグループのセットアップ手順については、IAM ユーザーガイドの「最初の IAM ユーザーおよび管理者グループの作成」を参照してください。

> **Important**
>
> IAM ポリシーは必ず、本稼働環境での使用前にテストしてください。MemoryDB のアクションによっては、シンプルに見えても、MemoryDB コンソールの使用時にそれらのアクションをサポートするために、他のアクションが必要になる場合があります。例えば、`memorydb:CreateCluster` は、 MemoryDBクラスターを作成するためのアクセス権限を付与します。ただし、このオペレーションを実行するために、 MemoryDB コンソールでは `Describe` と `List` の多数のアクションが使用されて、リストが事前設定されます。

例

- 例 1: MemoryDB リソースへの読み取り専用アクセスをユーザーに許可する
- 例 2: ユーザーに一般的な MemoryDB システム管理者タスクの実行を許可する
- 例 3: ユーザーにすべての MemoryDB API アクションへのアクセスを許可する
- 例 4: ユーザーが IAM CreateServiceLinkedRole API を呼び出すことを許可する

例 1: MemoryDB リソースへの読み取り専用アクセスをユーザーに許可する

以下のポリシーでは、リソースを一覧表示する MemoryDB アクションを実行するためのアクセス権限をユーザーに付与します。通常、このタイプのアクセス権限ポリシーは管理者グループにアタッチします。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[{
      "Sid": "MemDBUnrestricted",
      "Effect":"Allow",
      "Action": [
          "memorydb:Describe*",
          "memorydb:List*"],
      "Resource":"*"
      }
```

```
    ]
}
```

例 2: ユーザーに一般的な MemoryDB システム管理者タスクの実行を許可する

一般的なシステム管理者タスクには、クラスター、パラメータ、パラメータグループの変更が含まれます。システム管理者は MemoryDB イベントに関する情報を取得することが必要になる場合もあります。以下のポリシーでは、これらの一般的なシステム管理タスクの MemoryDB アクションを実行する権限をユーザーに付与します。通常、このタイプのアクセス権限ポリシーはシステム管理者グループにアタッチします。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "MDBAllowSpecific",
            "Effect": "Allow",
            "Action": [
                "memorydb:UpdateCluster",
                "memorydb:DescribeClusters",
                "memorydb:DescribeEvents",
                "memorydb:UpdateParameterGroup",
                "memorydb:DescribeParameterGroups",
                "memorydb:DescribeParameters",
                "memorydb:ResetParameterGroup"
            ],
            "Resource": "*"
        }
    ]
}
```

例 3: ユーザーにすべての MemoryDB API アクションへのアクセスを許可する

以下のポリシーでは、ユーザーにすべての MemoryDB アクションへのアクセスを許可します。このタイプのアクセス権限ポリシーは管理者ユーザーにのみ付与することをお勧めします。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[{
      "Sid": "MDBAllowAll",
      "Effect":"Allow",
      "Action":[
          "memorydb:*" ],
      "Resource":"*"
      }
   ]
}
```

例 4: ユーザーが IAM CreateServiceLinkedRole API を呼び出すことを許可する

次のポリシーでは、ユーザーが IAM `CreateServiceLinkedRole` API を呼び出すことを許可します。mutative MemoryDB オペレーションを実行するユーザーには、このタイプのアクセス許可ポリシーを与えることをお勧めします。

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"CreateSLRAllows",
      "Effect":"Allow",
      "Action":[
        "iam:CreateServiceLinkedRole"
      ],
      "Resource":"*",
      "Condition":{
        "StringLike":{
          "iam:AWSServiceName":"memorydb.amazonaws.com"
        }
      }
    }
  ]
}
```

リソースレベルのアクセス許可

IAM ポリシーでリソースを指定することで、アクセス許可の範囲を制限できます。AWS CLI API アクションの多くは、アクションの動作に応じて異なるリソースタイプをサポートしています。各 IAM ポリシーステートメントによって、リソースで実行されるアクションに対するアクセス許可が付与されます。アクションが名前の付いたリソースで動作しない場合、またはすべてのリソースに対してアクションを実行するアクセス許可を付与した場合、ポリシー内のリソースの値はワイルドカード (*) になります。多くの API アクションでは、リソースの Amazon リソースネーム (ARN)、または複数のリソースに一致する ARN パターンを指定することによって、ユーザーが変更できるリソースを制限できます。リソース別にアクセス許可を制限するには、ARN 別にリソースを指定します。

MemoryDB リソース ARN フォーマット

> **Note**
>
> リソースレベルのアクセス許可を有効にするには、ARN 文字列のリソース名を小文字にする必要があります。

- ユーザー – arn:aws:memorydb:*us-east-1:123456789012*:user/user1
- ACL – arn:aws:memorydb:*us-east-1:123456789012*:acl/my-acl
- クラスター – arn:aws:memorydb:*us-east-1:123456789012*:cluster/my-cluster
- スナップショット – arn:aws:memorydb:*us-east-1:123456789012*:snapshot/my-snapshot
- パラメータグループ – arn:aws:memorydb:*us-east-1:123456789012*:parametergroup/my-parameter-group
- サブネットグループ – arn:aws:memorydb:*us-east-1:123456789012*:subnetgroup/my-subnet-group

例

- 例 1: 特定の MemoryDB リソースタイプへのフルアクセスをユーザーに許可する
- 例 2: クラスターへのユーザーアクセスを拒否する。

例 1: 特定の MemoryDB リソースタイプへのフルアクセスをユーザーに許可する

次のポリシーでは、サブネットグループ、セキュリティグループ、クラスターのすべてのリソースへの指定された `account-id` フルアクセスを明示的に許可します。

```
{
        "Sid": "Example1",
        "Effect": "Allow",
        "Action": "memorydb:*",
        "Resource": [
             "arn:aws:memorydb:us-east-1:account-id:subnetgroup/*",
             "arn:aws:memorydb:us-east-1:account-id:securitygroup/*",
             "arn:aws:memorydb:us-east-1:account-id:cluster/*"
        ]
}
```

例 2: クラスターへのユーザーアクセスを拒否する。

次の例では、特定のクラスターへの指定された account-id アクセスを明示的に拒否します。

```
{
        "Sid": "Example2",
        "Effect": "Deny",
        "Action": "memorydb:*",
        "Resource": [
                "arn:aws:memorydb:us-east-1:account-id:cluster/name"
        ]
}
```

MemoryDB 用のサービスリンクロール

MemoryDB は AWS Identity and Access Management (IAM) サービスリンクロールを使用します。サービスリンクロールは、MemoryDB などの AWS サービスに直接リンクされる一意なタイプの IAM ロールです。MemoryDB サービスリンクロールは、MemoryDB によって事前定義されています。それらには、サービスがユーザーのクラスターに代わって AWS のサービスを呼び出すために必要なすべてのアクセス許可が含まれます。

必要な権限を手動で追加する必要がないため、サービスリンクロールは MemoryDB のセットアップを容易にします。ロールは AWS アカウント内にありますが、MemoryDB のユースケースにリンクされており、アクセス権限が事前に定義されています。これらのロールを引き受けることができるのは MemoryDB のみで、事前定義されたアクセス権限ポリシーを使用することができるのはこれらのロールのみです。ロールを削除するには、まず関連リソースを削除します。これは、リソースにアクセスするためのアクセス権限を誤って削除できないように、MemoryDB リソースを保護します。

サービスにリンクされたロールをサポートする他のサービスについては、「IAM と連携する AWS サービス」を参照して、サービスにリンクされたロール列がはいになっているサービスを見つけてく

ださい。サービスにリンクされた役割に関するドキュメントをサービスで表示するには[はい] リンクを選択してください。

目次

- MemoryDB のサービスにリンクされたロールのアクセス権限
- サービスにリンクされたロールの作成 (IAM)
  - サービスにリンクされたロールの作成 (IAM コンソール)
  - サービスにリンクされたロールの作成 (IAM CLI)
  - サービスにリンクされたロールの作成 (IAM API)
- MemoryDB のサービスにリンクされたロールの説明の編集
  - サービスにリンクされたロールの説明の編集 (IAMコンソール)
  - サービスにリンクされたロールの説明の編集 (IAM CLI)
  - サービスにリンクされたロールの説明の編集 (IAM API)
- MemoryDB 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のクリーンアップ
  - サービスにリンクされたロールの削除 (IAMコンソール)
  - サービスにリンクされたロールの削除 (IAM CLI)
  - サービスにリンクされたロールの削除 (IAM API)

MemoryDB のサービスにリンクされたロールのアクセス権限

MemoryDB では、[AWSServiceRoleForMemoryDB] という名前のサービスリンクロールを使用しています - このポリシーでは、ユーザーのクラスターを管理するために、MemoryDB が必要に応じてユーザーの代わりに AWS リソースを管理できます。

AWSServiceRoleForMemoryDB サービスリンクロールのアクセス権限ポリシーでは、MemoryDB は、指定されたリソースで次のアクションを完了できます。

JSON

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

```
    "Action": [
     "ec2:CreateTags"
    ],
    "Resource": "arn:aws-cn:ec2:*:*:network-interface/*",
    "Condition": {
     "StringEquals": {
      "ec2:CreateAction": "CreateNetworkInterface"
     },
     "ForAllValues:StringEquals": {
      "aws:TagKeys": [
       "AmazonMemoryDBManaged"
      ]
     }
    }
   },
   {
    "Effect": "Allow",
    "Action": [
     "ec2:CreateNetworkInterface"
    ],
    "Resource": [
     "arn:aws-cn:ec2:*:*:network-interface/*",
     "arn:aws-cn:ec2:*:*:subnet/*",
     "arn:aws-cn:ec2:*:*:security-group/*"
    ]
   },
   {
    "Effect": "Allow",
    "Action": [
     "ec2:DeleteNetworkInterface",
     "ec2:ModifyNetworkInterfaceAttribute"
    ],
    "Resource": "arn:aws-cn:ec2:*:*:network-interface/*",
    "Condition": {
     "StringEquals": {
      "ec2:ResourceTag/AmazonMemoryDBManaged": "true"
     }
    }
   },
   {
    "Effect": "Allow",
    "Action": [
     "ec2:DeleteNetworkInterface",
     "ec2:ModifyNetworkInterfaceAttribute"
```

```
    ],
    "Resource": "arn:aws-cn:ec2:*:*:security-group/*"
   },
   {
    "Effect": "Allow",
    "Action": [
     "ec2:DescribeSecurityGroups",
     "ec2:DescribeNetworkInterfaces",
     "ec2:DescribeAvailabilityZones",
     "ec2:DescribeSubnets",
     "ec2:DescribeVpcs"
    ],
    "Resource": "*"
   },
   {
    "Effect": "Allow",
    "Action": [
     "cloudwatch:PutMetricData"
    ],
    "Resource": "*",
    "Condition": {
     "StringEquals": {
      "cloudwatch:namespace": "AWS/MemoryDB"
     }
    }
   }
  ]
}
```

詳細については、「AWS マネージドポリシー:MemoryDBServiceRolePolicy」を参照してください。

IAM エンティティが AWSServiceRoleForMemoryDB サービスにリンクされたロールを作成するには以下のポリシーステートメントを IAM エンティティのアクセス許可に追加します。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB*",
```

```
    "Condition": {"StringLike": {"iam:AWSServiceName": "memorydb.amazonaws.com"}}
}
```

IAM エンティティが AWSServiceRoleForMemoryDB サービスにリンクされたロールを削除するには

以下のポリシーステートメントを IAM エンティティのアクセス許可に追加します。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "memorydb.amazonaws.com"}}
}
```

または、AWS マネージドポリシーを使用して、MemoryDB へのフルアクセスを付与することもできます。

サービスにリンクされたロールの作成 (IAM)

IAM コンソール、CLI または API を使用して、サービスにリンクされたロールを作成できます。

サービスにリンクされたロールの作成 (IAM コンソール)

IAM コンソールを使用して、サービスにリンクされたロールを作成できます。

サービスにリンクされたロールを作成するには (コンソール)

1. AWS マネジメントコンソール 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. IAM コンソールのナビゲーションペインで [ロール] をクリックします。次に、新しいロールの作成を選択します。
3. 信頼されたエンティティの種類を選択 の下で、AWS Service (サービス) を選択します。
4. [またはサービスを選択してそのユースケースを表示する] で、[MemoryDB] を選択します。
5. 次: 許可 を選択します。
6. ポリシー名 の下で、MemoryDBServiceRolePolicy はこのロールに必要であることに注意してください。次: タグ を選択します。

7. タグは、サービスにリンクされたロールではサポートされないことに注意してください。次: レビュー を選択します。
8. 「オプショナル」ロールの説明 で、サービスにリンクされた新しいロールの説明を編集します。
9. ロール情報を確認し、ロールの作成 を選択します。

サービスにリンクされたロールの作成 (IAM CLI)

AWS Command Line Interface から IAM オペレーションを使用して、サービスにリンクされたロールを作成できます。このロールには、ロールを引き受けるためにサービスで必要な信頼ポリシーやインラインポリシーを含めることができます。

サービスにリンクされたロールを作成するには (CLI)

次のオペレーションを使用してください。

```
$ aws iam create-service-linked-role --aws-service-name memorydb.amazonaws.com
```

サービスにリンクされたロールの作成 (IAM API)

IAM API を使用して、サービスにリンクされたロールを作成できます。このロールには、ロールを引き受けるためにサービスで必要な信頼ポリシーやインラインポリシーを含めることができます。

サービスにリンクされたロールを作成するには (API)

CreateServiceLinkedRole API コールを使用します。リクエストで、サービス名`memorydb.amazonaws.com`を指定します。

MemoryDB のサービスにリンクされたロールの説明の編集

MemoryDB では、AWSServiceRoleForMemoryDB のサービスにリンクされたロールを編集できません。サービスリンクロールの作成後は、さまざまなエンティティがロールを参照する可能性があるため、ロール名を変更することはできません。ただし、IAM を使用してロールの説明を編集することはできます。

サービスにリンクされたロールの説明の編集 (IAMコンソール)

サービスにリンクされたロールの説明は、IAM コンソールを使用して編集できます。

サービスにリンクされたロールの説明を編集するには (コンソール)

1. IAM コンソールのナビゲーションペインで [ロール] をクリックします。
2. 変更するロールの名前を選択します。
3. ロールの説明の右端にある編集を選択します。
4. ボックスに新しい説明を入力し、保存を選択します。

サービスにリンクされたロールの説明の編集 (IAM CLI)

サービスにリンクされたロールの説明は、AWS Command Line Interface から IAM オペレーションを使用して編集できます。

サービスにリンクされたロールの説明を変更するには (CLI)

1. 「オプショナル」現在のロールの説明を表示するには、IAM オペレーション get-role の AWS CLI を使用します。

   Example

   ```
   $ aws iam get-role --role-name AWSServiceRoleForMemoryDB
   ```

   CLI オペレーションでは、ARN ではなくロール名を使用してロールを参照します。例えば、ロールの ARN が `arn:aws:iam::123456789012:role/myrole` である場合、そのロールを **myrole** と参照します。
2. サービスにリンクされたロールの説明を更新するには、IAM オペレーション update-role-description の AWS CLI を使用します。

   Linux、macOS、Unix の場合:

   ```
   $ aws iam update-role-description \
       --role-name AWSServiceRoleForMemoryDB \
       --description "new description"
   ```

   Windows の場合:

   ```
   $ aws iam update-role-description ^
       --role-name AWSServiceRoleForMemoryDB ^
       --description "new description"
   ```

サービスにリンクされたロールの説明の編集 (IAM API)

サービスにリンクされたロールの説明は、IAM API を使用して編集できます。

サービスにリンクされたロールの説明を変更するには (API)

1. 「オプショナル」現在のロールの説明を表示するには、IAM API オペレーション GetRole を使用します。

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=GetRole
      &RoleName=AWSServiceRoleForMemoryDB
      &Version=2010-05-08
      &AUTHPARAMS
   ```

2. ロールの説明を更新するには、IAM API オペレーション UpdateRoleDescription を使用します。

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=UpdateRoleDescription
      &RoleName=AWSServiceRoleForMemoryDB
      &Version=2010-05-08
      &Description="New description"
   ```

MemoryDB 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を必要とする機能やサービスが不要になった場合は、ロールを削除することをお勧めします。そうすることで、モニタリングや保守が積極的に行われていない未使用のエンティティを排除できます。ただし、削除する前に、サービスにリンクされた役割をクリーンアップする必要があります。

MemoryDB はサービスにリンクされたロールを削除しません。

サービスにリンクされたロールのクリーンアップ

IAM を使用してサービスにリンクされたロールを削除するには、まずそれに関連付けられているリソース (クラスター) がないことを確認する必要があります。

サービスにリンクされたロールにアクティブなセッションがあるかどうかを、IAM コンソールで確認するには

1. AWS マネジメントコンソール 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. IAM コンソールのナビゲーションペインで [ロール] をクリックします。次に、AWSServiceRoleForMemoryDB ロールの名前 (チェックボックスではありません) を選択します。
3. 選択したロールの 概要 ページで、アクセスアドバイザー タブを選択します。
4. アクセスアドバイザー タブで、サービスにリンクされたロールの最新のアクティビティを確認します。

AWSServiceRoleForMemoryDB を必要とする MemoryDB リソースを削除するには (コンソール)

- クラスターを削除するには、以下を参照してください。
  - の使用AWS マネジメントコンソール
  - の使用AWS CLI
  - MemoryDB API の使用

サービスにリンクされたロールの削除 (IAMコンソール)

IAM コンソールを使用して、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削除するには (コンソール)

1. AWS マネジメントコンソール 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. IAM コンソールのナビゲーションペインで [ロール] をクリックします。ロール名または行そのものではなく、削除するロール名の横にあるチェックボックスをオンにします。
3. ページ上部にある ロールのアクション で ロールの削除 を選択します。
4. 確認ページで、サービスの最終アクセス時間データを確認します。これは、選択したそれぞれのロールの AWS サービスへの最終アクセス時間を示します。これは、そのロールが現在アクティブであるかどうかを確認するのに役立ちます。先に進む場合は、Yes, Delete] (はい、削除する) を選択し、削除するサービスにリンクされたロールを送信します。
5. IAM コンソール通知を見て、サービスにリンクされたロールの削除の進行状況をモニタリングします。IAM サービスにリンクされたロールの削除は非同期であるため、削除するロールを送

信すると、削除タスクは成功または失敗する可能性があります。タスクが失敗した場合は、通知から 詳細を表示または リソースを表示を選択して、削除が失敗した理由を知ることができます。

サービスにリンクされたロールの削除 (IAM CLI)

AWS Command Line Interface から IAM オペレーションを使用して、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削除するには (CLI)

1. 削除するサービスにリンクされたロールの名前が分からない場合、以下のコマンドを入力します。このコマンドでは、アカウントにあるロールとその Amazon リソースネーム (ARN) を一覧表示します。

   ```
   $ aws iam get-role --role-name role-name
   ```

   CLI オペレーションでは、ARN ではなくロール名を使用してロールを参照します。例えば、ロールに ARN `arn:aws:iam::123456789012:role/myrole` がある場合、そのロールを **myrole** と参照します。

2. サービスにリンクされているロールは、使用されている、または関連するリソースがある場合は削除できないため、削除リクエストを送信する必要があります。これらの条件が満たされない場合、そのリクエストは拒否される可能性があります。レスポンスから `deletion-task-id` を取得して、削除タスクのステータスを確認する必要があります。サービスにリンクされたロールの削除リクエストを送信するには、以下を入力します。

   ```
   $ aws iam delete-service-linked-role --role-name role-name
   ```

3. 削除タスクのステータスを確認するには、以下を入力します。

   ```
   $ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
   ```

   削除タスクのステータスは、 `NOT_STARTED`、`IN_PROGRESS`, `SUCCEEDED`、または `FAILED` となります。削除が失敗した場合は、失敗した理由がコールによって返され、トラブルシューティングが可能になります。

サービスにリンクされたロールの削除 (IAM API)

IAM API を使用して、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削除するには (API)

1. サービスにリンクされたロールの削除リクエストを送信するには、DeleteServiceLinkedRole を呼び出します。リクエストで、ロール名を指定します。

   サービスにリンクされているロールは、使用されている、または関連するリソースがある場合は削除できないため、削除リクエストを送信する必要があります。これらの条件が満たされない場合、そのリクエストは拒否される可能性があります。レスポンスから `DeletionTaskId` を取得して、削除タスクのステータスを確認する必要があります。
2. 削除タスクのステータスを確認するには、GetServiceLinkedRoleDeletionStatus を呼び出します。リクエストで `DeletionTaskId` を指定します。

   削除タスクのステータスは、 `NOT_STARTED`、`IN_PROGRESS`, `SUCCEEDED`、または `FAILED` となります。削除が失敗した場合は、失敗した理由がコールによって返され、トラブルシューティングが可能になります。

MemoryDB の AWS マネージドポリシー

ユーザー、グループ、ロールにアクセス許可を追加するには、自分でポリシーを作成するよりも、AWS マネージドポリシーを使用する方が簡単です。チームに必要な権限のみを提供する IAM カスタマーマネージドポリシーを作成するには時間と専門知識が必要です。すぐに使用を開始するために、AWS マネージドポリシーを使用できます。これらのポリシーは一般的なユースケースを対象範囲に含めており、AWS アカウントで利用できます。AWS マネージドポリシーの詳細については「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWS のサービスはAWS マネージドポリシーを維持および更新します。AWS マネージドポリシーの許可を変更することはできません。サービスでは新しい機能を利用できるようにするために、AWS マネージドポリシーに権限が追加されることがあります。この種類の更新はポリシーがアタッチされている、すべてのアイデンティティ (ユーザー、グループおよびロール) に影響を与えます。新しい機能が立ち上げられた場合や、新しいオペレーションが使用可能になった場合に、各サービスが AWS マネージドポリシーを更新する可能性が最も高くなります。サービスはAWS マネージドポリシーから権限を削除しないため、ポリシーの更新によって既存の権限が破棄されることはありません。

さらに、AWS は複数のサービスにまたがるジョブ機能の特徴に対するマネージドポリシーもサポートしています。例えば、ReadOnlyAccess AWS マネージドポリシーではすべての AWS のサービスおよびリソースへの読み取り専用アクセスを許可します。サービスが新しい機能を起動する場合、AWS は新たなオペレーションとリソース用に、読み取り専用の許可を追加します。ジョブ機能ポリシーのリストと説明については、「IAM ユーザーガイド」の「AWS ジョブ機能の管理ポリシー」を参照してください。

AWS マネージドポリシー:MemoryDBServiceRolePolicy

MemoryDBServiceRolePolicy AWS マネージドポリシーをアカウントのアイデンティティにアタッチすることはできません。このポリシーは AWS MemoryDB サービスにリンクされたロールの一部です。このロールにより、サービスはアカウント内のネットワークインターフェイスとセキュリティグループを管理できます。

MemoryDB は、このポリシーの権限を使用して EC2 セキュリティグループとネットワークインターフェイスを管理します。これは MemoryDB クラスターを管理するために必要です。

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。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateTags"
   ],
   "Resource": "arn:aws-cn:ec2:*:*:network-interface/*",
   "Condition": {
    "StringEquals": {
     "ec2:CreateAction": "CreateNetworkInterface"
```

```
    },
    "ForAllValues:StringEquals": {
     "aws:TagKeys": [
      "AmazonMemoryDBManaged"
     ]
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterface"
   ],
   "Resource": [
    "arn:aws-cn:ec2:*:*:network-interface/*",
    "arn:aws-cn:ec2:*:*:subnet/*",
    "arn:aws-cn:ec2:*:*:security-group/*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DeleteNetworkInterface",
    "ec2:ModifyNetworkInterfaceAttribute"
   ],
   "Resource": "arn:aws-cn:ec2:*:*:network-interface/*",
   "Condition": {
    "StringEquals": {
     "ec2:ResourceTag/AmazonMemoryDBManaged": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DeleteNetworkInterface",
    "ec2:ModifyNetworkInterfaceAttribute"
   ],
   "Resource": "arn:aws-cn:ec2:*:*:security-group/*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeSecurityGroups",
```

```
    "ec2:DescribeNetworkInterfaces",
    "ec2:DescribeAvailabilityZones",
    "ec2:DescribeSubnets",
    "ec2:DescribeVpcs"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "cloudwatch:PutMetricData"
   ],
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "cloudwatch:namespace": "AWS/MemoryDB"
    }
   }
  }
 ]
}
```

MemoryDB の AWS 管理 (事前定義) ポリシー

AWS は、によって作成され管理されるスタンドアロンの IAM ポリシーを提供することで、多くの一般的ユースケースに対応します。AWSマネージドポリシーは、一般的ユースケースに必要な許可を付与することで、どの許可が必要なのかをユーザーが調査する必要をなく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「AWS マネージドポリシー」を参照してください。

以下の AWS マネージドポリシーはアカウントのユーザーにアタッチできます。これらは MemoryDB に固有です。

AmazonMemoryDBReadOnlyAccess

`AmazonMemoryDBReadOnlyAccess` ポリシーを IAM アイデンティティにアタッチできます。このポリシーは、すべての読み取り専用アクセスを許可する管理者権限を付与します。

AmazonMemoryDBReadOnlyAccess - MemoryDB リソースへの読み取り専用のアクセス権を付与します。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "memorydb:Describe*",
   "memorydb:List*"
  ],
  "Resource": "*"
 }]
}
```

AmazonMemoryDBFullAccess

AmazonMemoryDBFullAccess ポリシーを IAM アイデンティティにアタッチできます。このポリシーは、すべてのMemoryDBリソースへのフルアクセスを許可する管理者権限を付与します。

AmazonMemoryDBFullAccess - MemoryDB リソースへの完全なアクセス権を付与します。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
   "Effect": "Allow",
   "Action": "memorydb:*",
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": "iam:CreateServiceLinkedRole",
   "Resource": "arn:aws:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB",
   "Condition": {
    "StringLike": {
     "iam:AWSServiceName": "memorydb.amazonaws.com"
    }
   }
```

```
  }
 ]
}
```

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
   "Effect": "Allow",
   "Action": "memorydb:*",
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": "iam:CreateServiceLinkedRole",
   "Resource": "arn:aws-cn:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB",
   "Condition": {
    "StringLike": {
     "iam:AWSServiceName": "memorydb.amazonaws.com"
    }
   }
  }
 ]
}
```

独自のカスタム IAM ポリシーを作成して、MemoryDB API アクションにアクセス権限を付与することもできます。これらのカスタムポリシーは、それらのアクセス許可が必要な IAM ユーザーまたはグループにアタッチできます。

MemoryDB AWS マネージドポリシーへの更新

MemoryDB の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。このページの変更に関する自動通知については、ドキュメントの履歴ページの RSS フィードをサブスクライブしてください。

変更	説明	日付
AWS マネージドポリシー :MemoryDBServiceRolePolicy - ポリシーの追加	MemoryDBServiceRolePolicy によって memorydb:ReplicateMultiRegionClusterData のアクセス許可が追加されました。このアクセス許可により、サービスにリンクされたロールは MemoryDB マルチリージョンクラスターのデータをレプリケートできるようになります。	12/01/2024
AmazonMemoryDBFullAccess - ポリシーの追加	MemoryDB では、サポートされているリソースを記述して一覧表示するための新しいアクセス許可が追加されました。これらのアクセス許可は、MemoryDB がアカウント内のサポートされているすべてのリソースをクエリするために必要です。	10/07/2021
AmazonMemoryDBReadOnlyAccess - ポリシーの追加	MemoryDB では、サポートされているリソースを記述して一覧表示するための新しいアクセス許可が追加されました。これらのアクセス許可は、MemoryDB がアカウント内のサポートされているすべてのリソースをクエリしてアカウントベースのアプリケーションを作成するために必要です。	10/07/2021
MemoryDB が変更の追跡を開始しました	サービスの起動	8/19/2021

MemoryDB API の許可: アクション、リソース、条件リファレンス

アクセスコントロール を設定し、IAM ポリシーにアタッチするアクセス許可ポリシー (アイデンティティベースまたはリソースベース) を作成するときは、以下の表をリファレンスとして使用できます。この表には、各 MemoryDB API オペレーション、およびその実行のためのアクセス権限を付与できる対応するアクションを示しています。ポリシーの `Action` フィールドでアクションを指定し、ポリシーの `Resource` フィールドでリソースの値を指定します。特に明記されていない限り、リソースは必須です。一部のフィールドには、必須リソースとオプションリソースの両方が含まれます。リソース ARN がない場合、ポリシー内のリソースはワイルドカード (*) になります。

> **Note**
>
> アクションを指定するには、API オペレーション名 (`memorydb:DescribeClusters` など) の前に `memorydb:` プレフィックスを使用します。

ログ記録とモニタリング

モニタリングは、MemoryDB および他の AWS ソリューションの信頼性、可用性、パフォーマンスの維持に重要な要素です。AWS は、MemoryDB をモニタリングし、問題が発生した場合には報告を行い、必要に応じて自動アクションを実行するために以下のモニタリングツールを提供しています。

- Amazon CloudWatch は、AWS のリソースおよび AWS で実行しているアプリケーションをリアルタイムでモニタリングします。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、および指定したメトリクスが指定したしきい値に達したときに通知またはアクションを実行するアラームの設定を行うことができます。例えば、CloudWatch で Amazon EC2 インスタンスの CPU 使用率などのメトリクスを追跡し、必要に応じて新しいインスタンスを自動的に起動できます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。
- Amazon CloudWatch Logs では、Amazon EC2 インスタンス、CloudTrail、その他ソースから得たログファイルのモニタリング、保存、およびアクセスが可能です。CloudWatch Logs は、ログファイル内の情報をモニタリングし、特定のしきい値が満たされたときに通知します。高い耐久性を備えたストレージにログデータをアーカイブすることも可能です。詳細については、「Amazon CloudWatch Logs ユーザーガイド」を参照してください。
- AWS CloudTrail は、AWS アカウントにより、またはそのアカウントに代わって行われた API よルや関連イベントを取得し、指定した Amazon S3 バケットにログファイルを配信します。AWS を呼び出したユーザーとアカウント、呼び出し元の IP アドレス、および呼び出しの発生日時を特定できます。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。

Amazon CloudWatch による MemoryDB のモニタリング

CloudWatch を使用して MemoryDB を監視できます。CloudWatch は raw データを収集し、それを読み取り可能なほぼリアルタイムのメトリクスに処理します。これらの統計は 15 か月間保持されるため、履歴情報にアクセスし、ウェブアプリケーションまたはサービスの動作をより的確に把握できます。また、特定のしきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

以下のセクションでは、MemoryDB のメトリクスとディメンションを一覧表示しています。

トピック

- ホストレベルのメトリクス
- MemoryDB のメトリック
- モニタリングすべきメトリクス
- メトリクスの統計と期間の選択
- CloudWatch メトリクスを使用したモニタリング

ホストレベルのメトリクス

AWS/MemoryDB 名前空間は、各ノードに対する以下のホストレベルのメトリクスが含まれます。

以下の資料も参照してください

- MemoryDB のメトリック

メトリクス	説明	単位
CPUUtilization	ホスト全体の CPU 使用率の割合 (%)。Valkey および Redis OSS はシングルスレットであるため、4 個以上の vCPU を持つノードで EngineCPUUtilization メトリクスをモニタリングすることをお勧めします。	割合 (%)
FreeableMemory	ホストで使用可能な空きメモリの量。この数字は、OS によって解放できる可能性があるとレ	バイト

メトリクス	説明	単位
	ポートされる RAM のメモリとバッファから算出されます。	
NetworkBytesIn	ホストがネットワークから読み取ったバイト数。	バイト
NetworkBytesOut	すべてのネットワークインターフェイスでの、このインスタンスから送信されたバイトの数。	バイト
NetworkPacketsIn	すべてのネットワークインターフェイスでの、このインスタンスによって受信されたパケットの数。このメトリクスは受信トラフィックのボリュームを単一インスタンスでのパケット数として識別します。	カウント
NetworkPacketsOut	すべてのネットワークインターフェイスでの、このインスタンスから送信されたパケットの数。このメトリクスは送信トラフィックのボリュームを単一インスタンスでのパケット数として識別します。	カウント
NetworkBandwidthInAllowanceExceeded	インバウンドの集計帯域幅がインスタンスの最大値を超えたために形成されたパケットの数。	カウント
NetworkConntrackAllowanceExceeded	接続トラッキングがインスタンスの最大数を超え、新しい接続を確立できなかったために形成されたパケットの数。これにより、インスタンスとの間で送受信されるトラフィックのパケット損失が発生する可能性があります。	カウント
NetworkBandwidthOutAllowanceExceeded	アウトバウンド集計帯域幅がインスタンスの最大値を超えたために形成されたパケットの数。	カウント
NetworkPacketsPerSecondAllowanceExceeded	1 秒あたりの双方向パケットがインスタンスの最大値を超えたために形成されたパケットの数。	カウント

メトリクス	説明	単位
NetworkMaxBytesIn	1 分間の受信バイトの最大毎秒バースト量。	バイト
NetworkMaxBytesOut	1 分間の送信バイトの最大毎秒バースト量。	バイト
NetworkMaxPacketsIn	1 分間の受信パケットの最大毎秒バースト量。	カウント
NetworkMaxPacketsOut	1 分間の送信パケットの最大毎秒バースト量。	カウント
SwapUsage	ホストで使用されるスワップの量。	バイト

MemoryDB のメトリック

AWS/MemoryDB 名前空間には、次のメトリクスが含まれます。

ReplicationLag、EngineCPUUtilization、SuccessfulWriteRequestLatency、および SuccessfulReadRequestLatency を除き、これらのメトリクスは、Valkey および Redis OSS の info コマンドから算出されます。各メトリクスは、ノードレベルで算出されます。

INFO コマンドの詳細なドキュメントについては、「INFO」を参照してください。

「」、「」も参照してください。

- ホストレベルのメトリクス

メトリクス	説明	単位
ActiveDefragHits	アクティブなデフラグメンテーションプロセスで実行された 1 分あたりの値の再割り当て数。これは、INFO での active_defrag_hits 統計から算出されます。	数値
AuthenticationFailures	AUTH コマンドを使用した認証の失敗回数の総数。個々の認証失敗の詳細については、ACL ログ コマンドを使用して検索できます。不正アクセスの試みを検出するために、このアラームを設定することをお勧めします。	カウント

メトリクス	説明	単位
BytesUsedForMemoryDB	データセット、バッファなど、すべての目的で MemoryDB によって割り当てられた合計バイト数。	バイト
	データ階層化 を使用するクラスターの Dimension: Tier=SSD : SSD によって使用される合計バイト数です。	バイト
	データ階層化 を使用するクラスターの Dimension: Tier=Memory :メモリによって使用される合計バイト数です。これは、INFO での used_memory 統計の値です。	バイト
BytesReadFromDisk	ディスクから読み取られる 1 分あたりの合計バイト数です。データ階層化 を使用するクラスターのみがサポートされます。	バイト
BytesWrittenToDisk	ディスクに書き込まれる 1 分あたりの合計バイト数です。データ階層化 を使用するクラスターのみがサポートされます。	バイト
CommandAuthorizationFailures	ユーザーが呼び出すためのアクセス許可を持たないコマンドの実行に失敗した試行の合計数。個々の認証失敗の詳細については、ACL ログ コマンドを使用して検索できます。不正アクセスの試みを検出するために、このアラームを設定することをお勧めします。	カウント
CurrConnections	リードレプリカからの接続を除く、クライアント接続の数。MemoryDB は、それぞれのケースで 2 ~ 4 個の接続を使用してクラスターをモニタリングします。これは、INFO での connected_clients 統計から算出されます。	カウント

メトリクス	説明	単位
CurrItems	キャッシュの項目数。これは、keyspace 統計に基づき、キー空間全体のすべてのキーを合計することで算出されます。	カウント
	Dimension: Tier=Memory を使用するクラスターの です。データ階層化メモリ内の項目の数です。	カウント
	Dimension: Tier=SSD を使用するクラスターの (ソリッドステートドライブ) です。データ階層化SSD 内の項目の数です。	カウント
DatabaseMemoryUsagePercentage	使用中のクラスターで使用中のメモリの割合。これは、INFO の used_memory/maxmemory を使用して計算されます。	割合 (%)
DatabaseCapacityUsagePercentage	使用中のクラスターの総データ容量の割合。 データ階層化インスタンスでは、メトリクスは (used_memory - mem_not_counted_for_evict + SSD used) / (maxmemory + SSD total capacity) として計算され、used_memory と maxmemory は INFO から取得されます。 それ以外の場合、メトリクスは used_memory/maxmemory を使用して計算されます。	割合 (%)
DB0AverageTTL	INFO コマンドの keyspace 統計から DBO の avg_ttl を公開します。	ミリ秒

メトリクス	説明	単位
EngineCPUUtilization	Valkey または Redis OSS エンジンスレッドの CPU 使用率を提供します。エンジンはシングルスレッドであるため、このメトリクスを使用して、プロセス自体のロードを分析できます。EngineCPUUtilization メトリクスは、プロセスのより正確な可視性を提供します。CPUUtilization メトリクスと組み合わせてそれを使用できます。CPUUtilization は、他のオペレーティングシステムや管理プロセスを含むサーバーインスタンス全体の CPU 使用率を公開します。4 個以上の vCPU を持つ大きなノードの場合は、EngineCPUUtilization メトリクスを使用して、スケーリングのしきい値をモニタリングおよび設定します。 **Note** MemoryDB ホスト上で、マネージドデータベースのエクスペリエンスを提供するために、バックグラウンドプロセスがホストをモニタリングします。これらのバックグラウンドプロセスは、CPU ワークロードのかなりの部分を占有する可能性があります。これは、vCPU が 2 個を超える大規模なホストでは重要ではありません。ただし、vCPU が 2 個以下の小規模なホストには影響を与える可能性があります。EngineCPUUtilization メトリクスのみをモニタリングする場合、Valkey または Redis OSS エンジンからの CPU 使用率と、バックグラウンドモニタリングプロセスからの CPU 使用率の両方が高く、ホストが過負荷	割合 (%)

メトリクス	説明	単位
	になっている状況には気付くことができません。したがって、vCPU が 2 個以下のホストについては、CPUUtilization メトリクスをモニタリングすることをお勧めします。	
Evictions	maxmemory の制限のため排除されたキーの数。これは、INFO での evicted_keys 統計から算出されます。	カウント
IsPrimary	ノードが現在のシャードのプライマリノードかどうかを示します。メトリクスは 0 (プライマリではない) または 1 (プライマリ) にすることができます。	カウント
KeyAuthorizationFailures	ユーザーがアクセス許可を持たないキーへのアクセスに失敗した試行の合計数。個々の認証失敗の詳細については、ACL ログ コマンドを使用して検索できます。不正アクセスの試みを検出するために、このアラームを設定することをお勧めします。	カウント
KeyspaceHits	メインディクショナリで読み取り専用のキー検索に成功した数。これは、INFO での keyspace_hits 統計から算出されます。	カウント
KeyspaceMisses	メインディクショナリで読み取り専用のキー検索に失敗した数。これは、INFO での keyspace_misses 統計から算出されます。	カウント

メトリクス	説明	単位
KeysTracked	キートラッキングによって追跡されるキーの数 (tracking-table-max-keys のパーセンテージ)。キーラッキングは、クライアント側のキャッシュを支援するために使用され、キーが変更されたときにクライアントに通知します。	カウント
MaxReplicationThroughput	観測された最大スループット。スループットは、トラフィックバーストを特定するために短い時間間隔でサンプリングされます。サンプリングされた値の最大値がレポートされます。サンプリングは 1 分間隔で行われます。例えば、10 ミリ秒の間に 1 MB のデータが書き込まれる場合、このメトリクスの値は 100 MBps になります。このメトリクスが 100 MBps を超えると、書き込みスループットのスロットリングにより、より大きな書き込みレイテンシーが観測される可能性があることに注意してください。	1 秒あたりのバイト数
MemoryFragmentationRatio	Valkey または Redis OSS エンジンのメモリ割り当ての効率を示します。特定のしきい値は、異なる動作を意味します。推奨値は、1.0 を超える断片化です。これは、INFO の mem_fragmentation_ratio statistic から計算されます。	数値

メトリクス	説明	単位
MultiRegionClusterReplicationLag	MemoryDB マルチリージョンクラスターでは、MultiRegionClusterReplicationLag は、更新がリージョンクラスターのマルチ AZ トランザクションログに書き込まれてから、この更新がマルチリージョンクラスター内の別のリージョンクラスターのプライマリノードに書き込まれるまでの経過時間を測定します。このメトリクスは、すべてのレプリケート元リージョンとレプリケート先リージョンのペアについてシャードレベルで出力されます。	ミリ秒
NewConnections	この期間内にサーバーによって受け入れられた接続の総数。これは、INFO での total_connections_received 統計から算出されます。	カウント
NumItemsReadFromDisk	ディスクから取得される 1 分あたりの項目の総数です。データ階層化 を使用するクラスターのみがサポートされます。	カウント
NumItemsWrittenToDisk	ディスクに書き込まれる 1 分あたりの項目の総数です。データ階層化 を使用するクラスターのみがサポートされます。	カウント
PrimaryLinkHealthStatus	このステータスの値は、0 または 1 のいずれかになります。値 0 は、MemoryDB プライマリノードのデータが、EC2 の Valkey または Redis OSS エンジンと同期されていないことを示します。値 1 は、データが同期されていることを示します。	ブール値
Reclaimed	キーの有効期限切れイベントの総数。これは、INFO での expired_keys 統計から算出されます。	カウント

メトリクス	説明	単位
ReplicationBytes	レプリケートされたノードについては、ReplicationBytes は、プライマリがすべてのレプリカに対して送信するバイト数を報告します。このメトリクスは、クラスターでの書き込み負荷を表します。これは、INFO での master_repl_offset 統計から算出されます。	バイト
ReplicationDelayedWriteCommands	同期レプリケーションにより遅延した書き込みコマンドの数。レプリケーションは、ネットワークの輻輳や最大レプリケーションスループット の超過など、さまざまな要因により遅延する可能性があります。	カウント
ReplicationLag	このメトリクスは、リードレプリカとして実行中のノードにのみ適用できます。レプリカのプライマリノードからの変更適用の進行状況を秒で表します。	Seconds (秒)
SuccessfulWriteRequestLatency	書き込みリクエストが成功するまでのレイテンシー。 有効な統計: Average、Sum、Min、Max、Sample Count、p0 ~ p100 のパーセンタイル。サンプル数には、正常に実行されたコマンドのみが含まれます。Valkey 7.2 以降で使用できます。	マイクロ秒
SuccessfulReadRequestLatency	読み取りリクエストが成功するまでのレイテンシー。 有効な統計: Average、Sum、Min、Max、Sample Count、p0 ~ p100 のパーセンタイル。サンプル数には、正常に実行されたコマンドのみが含まれます。Valkey 7.2 以降で使用できます。	マイクロ秒

メトリクス	説明	単位
ErrorCount	指定された期間中に失敗したコマンドの総数。 有効な統計: Average、Sum、Min、Max	カウント

以下は特定の種類のコマンドの集計で、info commandstats から算出されています。コマンドスタッツのセクションには、呼び出し回数など、コマンドタイプに基づく統計情報が表示されます。

利用可能なコマンドの完全な一覧については、「コマンド」を参照してください。

メトリクス	説明	単位
EvalBasedCmds	eval ベースのコマンドの合計数。これは、eval、evalsha を合計することによって commandstats 統計から算出されます。	カウント
GeoSpatialBasedCmds	地理空間ベースのコマンドの総数。これは commandstats 統計から算出されます。これは、すべての geo の種類のコマンド (geoadd、geodist、geohash、geopos、georadiu、および georadiusbymember) を合計することによって算出されます。	カウント
GetTypeCmds	read-only 型のコマンドの合計数。これは、すべての read-only の種類のコマンド (get、hget、scard、lrange など) を合計することによって commandstats 統計から算出されます。	カウント
HashBasedCmds	ハッシュベースのコマンドの総数。これは、1 つ以上のハッシュに対して実行されるすべてのコマンド (hget、hkeys、hvals、hdel など) を合計することによって commandstats 統計から算出されます。	カウント
HyperLogLogBasedCmds	HyperLogLog ベースのコマンドの合計数。これは、すべての pf の種類のコマンド	カウント

メトリクス	説明	単位
	(pfadd、pfcount、pfmerge など) を合計することによって commandstats 統計から算出されます。	
JsonBasedCmds	JSON ベースのコマンドの総数。これは、commandstats 統計に基づき、1 つ以上の JSON ドキュメントオブジェクトに対して実行されるすべてのコマンドを合計して算出されます。	カウント
KeyBasedCmds	キーベースのコマンドの総数。これは、複数のデータ構造で 1 つ以上のキーに対して実行されるすべてのコマンド (del、expire、rename など) を合計することによって、commandstats 統計から算出されます。	カウント
ListBasedCmds	リストベースのコマンドの総数。これは、1 つ以上のリストに対して実行されるすべてのコマンド (lindex、lrange、lpush、ltrim など) を合計することによって commandstats 統計から算出されます。	カウント
PubSubBasedCmds	pub/sub 機能のコマンドの総数。これは、pub/sub 機能で使用されるすべてのコマンド (psubscribe、publish、pubsub、punsubscribe、subscribe、unsubscribe) を合計することによって commandstats 統計から算出されます。	カウント
SearchBasedCmds	読み取りコマンドと書き込みコマンドの両方を含む、セカンダリインデックスと検索コマンドの総数。これは、セカンダリインデックスに作用するすべての search コマンドを合計することにより、commandstats 統計から導出されます。	カウント

メトリクス	説明	単位
SearchBasedGetCmds	セカンダリインデックスと検索読み取り専用コマンドの総数。これは、すべてのセカンダリインデックスと search get コマンドを合計することによって、commandstats 統計から導出されます。	カウント
SearchBasedSetCmds	セカンダリインデックスと検索書き込みコマンドの総数。これは、すべてのセカンダリインデックスと search set コマンドを合計することによって、commandstats 統計から導出されます。	カウント
SearchNumberOfIndexes	インデックスの総数。	カウント
SearchNumberOfIndexedKeys	インデックス付きキーの総数	カウント
SearchTotalIndexSize	すべてのインデックスによって使用されるメモリ (バイト)。	バイト
SetBasedCmds	セットベースのコマンドの総数。これは、1 つ以上のセットに対して実行されるすべてのコマンド (scard、sdiff、sadd、sunion など) を合計することによって commandstats 統計から算出されます。	カウント
SetTypeCmds	write 型のコマンドの合計数。これは、データ上で動作する mutative の種類のすべてのコマンド (set、hset、sadd、lpop など) を合計することによって commandstats 統計から算出されます。	カウント

メトリクス	説明	単位
SortedSetBasedCmds	ソートされたセットベースのコマンドの総数。これは、1 つ以上のソートされたセットに対して実行されるすべてのコマンド (zcount、zrange、zrank、zadd など) を合計することによって commandstats 統計から算出されます。	カウント
StringBasedCmds	文字列ベースのコマンドの総数。これは、1 つ以上の文字列に対して実行されるすべてのコマンド (strlen、setex、setrange など) を合計することによって commandstats 統計から算出されます。	カウント
StreamBasedCmds	ストリームベースのコマンドの総数。これは、1 つ以上のストリームデータの種類に対して実行されるすべてのコマンド (xrange、xlen、xadd、xdel など) を合計することによって commandstats 統計から算出されます。	カウント

モニタリングすべきメトリクス

次の CloudWatch メトリクスは、MemoryDB パフォーマンスを把握するのに役立ちます。ほとんどの場合、パフォーマンスの問題が発生する前に修正作業を行うことができるように、これらのメトリクスに CloudWatch アラームを設定することをお勧めします。

モニタリングするメトリクス

- CPUUtilization (CPU使用度)
- EngineCPUUtilization
- SwapUsage
- Evictions
- CurrConnections
- メモリ
- ネットワーク
- レイテンシー
- レプリケーション

CPUUtilization (CPU使用度)

パーセント値でレポートされるホストレベルのメトリクスです。詳細については、「ホストレベルのメトリクス」を参照してください。

2 個以下の vCPU を持つ小さなノードタイプの場合は、`CPUUtilization` メトリクスを使用してワークロードをモニタリングします。

一般的に、利用可能な CPU の 90% にしきい値を設定することをお勧めします。Valkey および Redis OSS はシングルスレッドであるため、実際のしきい値はノードの総容量に占める割合のごく一部になるよう計算します。例えば、2 個のコアを搭載するノードタイプを使用しているとします。この場合、CPUUtilization のしきい値は 90/2、つまり 45% になります。ノードタイプに搭載されたコア数 (vCPU) を調べる方法については、「MemoryDB 料金表」を参照してください。

使用しているノードのコア数に基づいて独自のしきい値を決定する必要があります。このしきい値を超えた場合で、主なワークロードが読み込みリクエストから生成されている場合、リードレプリカを追加してクラスターをスケールします。主なワークロードが書き込みリクエストからのものである場合は、より多くのシャードを追加して、より多くのプライマリノード間で書き込みワークロードを分散することをお勧めします。

> **Tip**
>
> ホストレベルのメトリクス CPUUtilization を使用する代わりに、Valkey または Redis OSS エンジンコアの使用率をレポートするメトリクス EngineCPUUtilization を使用できる場合があります。 コードでこのメトリクスが利用できるかどうか、およびその詳細については、「MemoryDB のメトリクス」を参照してください。

4 個以上の vCPU を持つ大きなノードタイプでは、Valkey または Redis OSS エンジンコアでの使用量のパーセント値をレポートする EngineCPUUtilization メトリクスを使用することをお勧めします。コードでこのメトリクスが利用できるかどうか、およびその詳細については、「MemoryDB のメトリクス」を参照してください。

EngineCPUUtilization

4 個以上の vCPU を持つ大きなノードタイプでは、Valkey または Redis OSS エンジンコアでの使用量のパーセント値をレポートする EngineCPUUtilization メトリクスを使用することをお勧めします。コードでこのメトリクスが利用できるかどうか、およびその詳細については、「MemoryDB のメトリクス」を参照してください。

SwapUsage

バイト単位でレポートされるホストレベルのメトリクスです。詳細については、「ホストレベルのメトリクス」を参照してください。

FreeableMemory CloudWatch メトリクスが 0 に近い (つまり 100 MB 未満) 場合、または SwapUsage メトリクスが FreeableMemory メトリクスより大きい場合は、ノードがメモリプレッシャーを受けている可能性があります。

Evictions

これは、エンジンのメトリクスです。アプリケーションニーズに基づいてこのメトリクスの独自のアラームしきい値を決定することをお勧めします。

CurrConnections

これは、エンジンのメトリクスです。アプリケーションニーズに基づいてこのメトリクスの独自のアラームしきい値を決定することをお勧めします。

CurrConnections の値が大きくなった場合、アプリケーションに問題があることを示している可能性があります。アプリケーション動作を調査してこの問題を解決する必要があります。

メモリ

メモリは Valkey および Redis OSS の中核的な側面です。クラスターのメモリ使用率を理解することは、データの損失を回避し、データセットの将来の増加に対応するために必要です。ノードのメモリ使用率に関する統計は、INFO コマンドのメモリセクションで確認できます。

ネットワーク

クラスターのネットワーク帯域幅容量の決定要因の 1 つは、選択したノードの種類です。ノードのネットワーク容量の詳細については、「Amazon MemoryDB 料金表」を参照してください。

レイテンシー

レイテンシーメトリクス `SuccessfulWriteRequestLatency` および `SuccessfulReadRequestLatency` は、Valkey エンジンの MemoryDB がリクエストに応答するのにかかる合計時間を測定します。

> **Note**
>
> Valkey クライアントで CLIENT REPLY を有効にして Valkey パイプラインを使用すると、`SuccessfulWriteRequestLatency` および `SuccessfulReadRequestLatency` メトリクスの値が異常に大きくなる可能性があります。Valkey パイプラインは、個々のコマンドへの応答を待たずに複数のコマンドを一度に実行することでパフォーマンスを向上させる手法です。値が異常に大きくなるのを防ぐために、CLIENT REPLY OFF の状態でコマンドをパイプラインするように Redis クライアントを設定することをお勧めします。

レプリケーション

レプリケーションされるデータの量は、`ReplicationBytes` メトリクスを介して見ることができます。レプリケーション容量のスループットに対して`MaxReplicationThroughput` を監視できます。レプリケーション容量のスループットが最大になったら、シャードを追加することをお勧めします。

`ReplicationDelayedWriteCommands` はまた、ワークロードが最大レプリケーション容量スループットを超えているかどうかもわかります。MemoryDB でのレプリケーションの詳細については、「MemoryDB レプリケーションの概要」を参照してください

メトリクスの統計と期間の選択

CloudWatch では、各メトリクスの統計および期間を選択できますが、すべての組み合わせが役に立つとは言えません。例えば、CPUUtilization の Average、Minimum、および Maximum 統計は役に立ちますが、Sum 統計は役に立ちません。

MemoryDB のすべてのサンプルは、個々のノードに対して 60 秒間発行されています。任意の 60 秒間において、ノードメトリクスに含められるサンプルは 1 つだけです。

CloudWatch メトリクスを使用したモニタリング

MemoryDB と CloudWatch は、多様なメトリクスを収集できるように統合されています。CloudWatch を使用して、これらのメトリクスをモニタリングできます。

> **Note**
>
> 次の例には、CloudWatch コマンドラインツールが必要です。CloudWatch の詳細とデベロッパーツールのダウンロードについては、「CloudWatch 製品ページ」を参照してください。

次の手順は、CloudWatch を使用して、過去 1 時間のクラスターのストレージ領域統計を収集する方法を示しています。

> **Note**
>
> 以下の例で指定されている `StartTime` 値と `EndTime` 値は、例示を目的としています。実際のノードに適した開始時刻値および終了時刻値で置き換える必要があります。

MemoryDB の制限について詳しくは、「AWS MemoryDB のサービス制限」を参照してください。

CloudWatch メトリクスをモニタリングする (コンソール)

クラスターの CPU 使用率統計を収集するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. メトリクスを表示するノードを選択します。

> **Note**
>
> 20 個を超えるノードを選択すると、コンソールでメトリクスを表示できなくなります。

a. AWS マネジメントコンソールの [クラスター] ページで、1 つ以上のキャッシュクラスターの名前をクリックします。

   クラスターの詳細ページが表示されます。

b. ウィンドウ上部にある Nodes タブをクリックします。

c. 詳細ウィンドウの [ノード] タブで、メトリクスを表示するキャッシュノードを選択します。

   使用可能な CloudWatch メトリクスのリストがコンソールウィンドウの下部に表示されます。

d. CPU Utilization メトリクスをクリックします。

   CloudWatch コンソールが開き、選択されたメトリクスが表示されます。Statistic および Period ドロップダウンリストボックスや Time Range タブを使用すると、表示されるメトリクスを変更できます。

CloudWatch CLI を使用した CloudWatch メトリクスのモニタリング

クラスターの CPU 使用率統計を収集するには

- 以下のパラメータを指定して、CloudWatch コマンド aws cloudwatch get-metric-statistics を使用します (示されている開始時刻と終了時刻は例です。適切な開始時刻と終了時刻に置き換える必要があります)。

  Linux、macOS、Unix の場合:

```
aws cloudwatch get-metric-statistics CPUUtilization \
    --dimensions=ClusterName=mycluster,NodeId=0002" \
    --statistics=Average \
    --namespace="AWS/MemoryDB" \
    --start-time 2013-07-05T00:00:00 \
    --end-time 2013-07-06T00:00:00 \
    --period=60
```

Windows の場合:

```
mon-get-stats CPUUtilization ^
    --dimensions=ClusterName=mycluster,NodeId=0002" ^
    --statistics=Average ^
    --namespace="AWS/MemoryDB" ^
    --start-time 2013-07-05T00:00:00 ^
    --end-time 2013-07-06T00:00:00 ^
    --period=60
```

CloudWatch API を使用した CloudWatch メトリックスのモニタリング

クラスターの CPU 使用率統計を収集するには

- 以下のパラメータを指定して、CloudWatch API GetMetricStatistics を呼び出します (示されている開始時刻と終了時刻は例です。適切な開始時刻と終了時刻に置き換える必要があります)。
  - Statistics.member.1=Average
  - Namespace=AWS/MemoryDB
  - StartTime=2013-07-05T00:00:00
  - EndTime=2013-07-06T00:00:00
  - Period=60
  - MeasureName=CPUUtilization
  - Dimensions=ClusterName=mycluster,NodeId=0002

  Example

```
http://monitoring.amazonaws.com/
    ?SignatureVersion=4
    &Action=GetMetricStatistics
    &Version=2014-12-01
    &StartTime=2013-07-16T00:00:00
    &EndTime=2013-07-16T00:02:00
    &Period=60
    &Statistics.member.1=Average
```

```
&Dimensions.member.1="ClusterName=mycluster"
&Dimensions.member.2="NodeId=0002"
&Namespace=Amazon/memorydb
&MeasureName=CPUUtilization
&Timestamp=2013-07-07T17%3A48%3A21.746Z
&AWS;AccessKeyId=<&AWS; Access Key ID>
&Signature=<Signature>
```

MemoryDB イベントのモニタリング

重要なイベントがクラスター上で発生すると、MemoryDB から特定の Amazon SNS トピックに通知が送信されます。例には、ノードの追加の失敗、ノードの追加の成功、セキュリティグループの変更などが含まれます。主要イベントをモニタリングすることで、クラスターの現在の状態を知り、イベントに基づいて是正措置を取ることができます。

トピック

- MemoryDB Amazon SNS 通知の管理
- MemoryDB イベントの表示
- イベント通知と Amazon SNS

MemoryDB Amazon SNS 通知の管理

Amazon Simple Notification Service (Amazon SNS) を使用して重要なクラスターイベントの通知が送信されるように MemoryDB を設定できます。これらの例では、Amazon SNS トピックの Amazon リソースネーム (ARN) を使用してクラスターを設定し、通知を受け取るようにします。

> **Note**
>
> このトピックでは、Amazon SNS にサインアップし、Amazon SNS トピックをセットアップおよびサブスクライブしていることを前提としています。これを行う方法の詳細については、「Amazon Simple Notification Service デベロッパーガイド」を参照してください。

Amazon SNS トピックを追加する

以下のセクションでは、Amazon SNS トピックを AWS コンソール、AWS CLI、または MemoryDB API を使用して追加する方法について説明します。

Amazon SNS トピックを追加する (コンソール)

以下の手順は、クラスターの Amazon SNS トピックを追加する方法を示しています。

> **Note**
>
> このプロセスは、Amazon SNS トピックの変更に使用できます。

クラスターの Amazon SNS トピックを追加または変更するには (コンソール)

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. クラスター で、Amazon SNS トピック ARN を追加または変更するクラスターを選択します。
3. 変更を選択します。
4. クラスターを変更 の SNS 通知のトピック で、追加する SNS トピックを選択します。または、手動 ARN 入力 を選択して Amazon SNS トピックの ARN を入力します。
5. 変更を選択します。

Amazon SNS トピックを追加する (AWS CLI)

クラスターの Amazon SNS トピックを追加または変更するには、AWS CLI コマンド `update-cluster` を使用します。

次のコード例では、Amazon SNS トピック ARN を my-cluster に追加します。

Linux、macOS、Unix の場合:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --sns-topic-arn arn:aws:sns:us-east-1:565419523791:memorydbNotifications
```

Windows の場合:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --sns-topic-arn arn:aws:sns:us-east-1:565419523791:memorydbNotifications
```

詳細については、「UpdateCluster」を参照してください。

Amazon SNS トピックを追加する (MemoryDB API)

クラスターの Amazon SNS トピックを追加または変更するには、以下のパラメータを指定して UpdateCluster アクションを呼び出します。

- ClusterName=my-cluster
- SnsTopicArn=arn%3Aaws%3Asns%3Aus-east-1%3A565419523791%3AmemorydbNotifications

クラスターの Amazon SNS トピックを追加または更新するには、UpdateCluster アクションを呼び出します。

詳細については、「クラスターの更新」を参照してください。

Amazon SNS 通知の有効化と無効化

クラスターでは、通知を有効または無効にすることができます。次の手順は、Amazon SNS 通知を無効にする方法を示しています。

Amazon SNS 通知の有効化と無効化 (コンソール)

AWS マネジメントコンソール を使用して Amazon SNS 通知を無効にするには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 通知を変更するクラスターの左側にあるラジオボタンを選択します。
3. 変更を選択します。
4. クラスターを変更 の SNS 通知のトピック で、通知を無効にする を選択します。
5. 変更を選択します。

Amazon SNS 通知の有効化と無効化 (AWS CLI)

Amazon SNS 通知を無効にするには、以下のパラメータを指定して update-cluster コマンドを使用します。

Linux、macOS、Unix の場合:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
```

```
    --sns-topic-status inactive
```

Windows の場合:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --sns-topic-status inactive
```

Amazon SNS 通知の有効化と無効化 (MemoryDB API)

Amazon SNS 通知を無効にするには、以下のパラメータを指定して UpdateCluster アクションを呼び出します。

- ClusterName=my-cluster
- SnsTopicStatus=inactive

この呼び出しにより、以下のような出力が返されます。

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &ClusterName=my-cluster
    &SnsTopicStatus=inactive
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

MemoryDB イベントの表示

MemoryDB は、クラスターのインスタンス、セキュリティグループ、パラメータグループに関連するイベントを記録します。この情報には、イベントの日付と時刻、イベントのソース名とソースタイプ、イベントの説明などがあります。MemoryDB コンソール、AWS CLI `describe-events` コマンド、またはMemoryDB API アクション `DescribeEvents` を使用して、ログから簡単にイベントを取得できます。

次の手順は、過去 24 時間 (1440 分) のすべての MemoryDB イベントを表示する方法を示しています。

MemoryDB イベントの表示 (コンソール)

次の手順は、MemoryDB コンソールを使用してイベントを表示します。

MemoryDB コンソールを使用してイベント表示するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 左側のナビゲーションペインで イベント を選択します。

   イベント画面が開き、利用可能なすべてのイベントが一覧表示されます。Events 画面のリスト内の各行は 1 個のイベントを表し、イベントのソース、イベントの種類 (キャッシュクラスター、キャッシュパラメータグループ、キャッシュセキュリティグループ、キャッシュサブネットグループ)、イベントの GMT 時間、イベントの説明が表示されます。

   Filter を使用して、イベントリストにすべてのイベントを表示するか特定タイプのイベントのみを表示するかを指定できます。

メモリデータベースイベントの表示 (AWS CLI)

AWS CLI を使用して MemoryDB イベントのリストを作成するには、`describe-events` コマンドを使用します。オプションパラメータを使用して、一覧されるイベントのタイプ、イベントの期間、イベント一覧の最大数などを制御できます。

次のコードでは、最大 40 個のクラスターイベントを一覧表示します。

```
aws memorydb describe-events --source-type cluster --max-results 40
```

次のコードでは、過去 24 時間 (1440 分) のすべてのイベントを一覧表示します。

```
aws memorydb describe-events --duration 1440
```

describe-events のコマンドによる出力は次のようになります。

```
{
    "Events": [
        {
            "Date": "2021-03-29T22:17:37.781Z",
            "Message": "Added node 0001 in Availability Zone us-east-1a",
            "SourceName": "memorydb01",
            "SourceType": "cluster"
        },
        {
            "Date": "2021-03-29T22:17:37.769Z",
            "Message": "cluster created",
            "SourceName": "memorydb01",
            "SourceType": "cluster"
        }
    ]
}
```

使用できるパラメータおよび許可されたパラメータ値などの詳細については、「describe-events」を参照してください。

MemoryDB イベントの表示 (MemoryDB API)

MemoryDB API を使用して MemoryDB イベントのリストを生成するには、DescribeEvents アクションを使用します。オプションパラメータを使用して、一覧されるイベントのタイプ、イベントの期間、イベント一覧の最大数などを制御できます。

次のコードは、40 個の最新のクラスターイベントを一覧します。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeEvents
   &MaxResults=40
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cluster
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

次のコードは、過去 24 時間 (1440 分) のクラスターイベントを一覧します。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeEvents
   &Duration=1440
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cluster
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

上記のアクションでは、次のような出力が生成されます。

```
<DescribeEventsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
    <DescribeEventsResult>
        <Events>
            <Event>
                <Message>cluster created</Message>
                <SourceType>cluster</SourceType>
                <Date>2021-08-02T18:22:18.202Z</Date>
                <SourceName>my-memorydb-primary</SourceName>
            </Event>

 (...output omitted...)

        </Events>
    </DescribeEventsResult>
    <ResponseMetadata>
        <RequestId>e21c81b4-b9cd-11e3-8a16-7978bb24ffdf</RequestId>
    </ResponseMetadata>
</DescribeEventsResponse>
```

使用できるパラメータおよび許可されたパラメータ値などの詳細については、「DescribeEvents」を参照してください。

イベント通知と Amazon SNS

MemoryDBは、クラスターで重要なイベントが発生したときに、Amazon Simple Notification Service (SNS) を使用してメッセージを発行できます。この機能を使用すると、クラスターの個々のノードエンドポイントに接続されたクライアントコンピュータでサーバーリストを更新できます。

> **Note**
>
> 価格の情報やAmazon SNS ドキュメントへのリンクを含む、Amazon Simple Notification Service (SNS) の詳細については、「Amazon SNS 製品ページ」を参照してください。

通知は、指定した Amazon SNS トピック に発行されます。通知の要件は以下のとおりです:

- MemoryDB 通知に対して設定できるトピックは 1 つだけです。
- Amazon SNS トピックを所有する AWS アカウントは、通知が有効になっているクラスターを所有するアカウントと同じアカウントである必要があります。

MemoryDB イベント

以下の MemoryDB イベントにより、Amazon SNS 通知がトリガーされます:

イベント名	メッセージ	説明
MemoryDB: ノード追加が完了しました	`"Modified number of nodes from %d to %d"`	ノードがクラスターに追加され、使用可能になっています。
IPアドレスの空き不足による MemoryDB:AddNodeFailed	`"Failed to modify number of nodes from %d to %d due to insufficient free IP addresses"`	利用可能なIPアドレスが不足しているため、ノードを追加できませんでした。
MemoryDB: ClusterParametersChanged	`"Updated parameter group for the cluster"` 作成の場合は、`"Updated to use a ParameterGroup %s"` も送ります。	1 つ以上のクラスターパラメータが変更されました。
MemoryDB: クラスタープロビジョニングが完了しました	`"Cluster created."`	クラスターのプロビジョニングが完了し、クラスター内の

イベント名	メッセージ	説明
		ノードが使用可能になりました。
MemoryDB: 互換性のないネットワーク状態のため、クラスタープロビジョニングに失敗しました	`"Failed to create cluster due to incompatible network state. %s"`	存在しない 仮想プライベートクラウド (VPC) に新しいキャッシュクラスターに起動する試みが行われました。
MemoryDB: クラスターの復元に失敗しました	`"Restore from %s failed for node %s. %s"`	MemoryDB はスナップショットデータをクラスターに入力できませんでした。これは、Amazon S3 にスナップショットファイルが存在しないか、そのファイルに対する不適切なアクセス許可が原因である可能性があります。クラスターを記述する場合は、ステータスは `restore-failed` です。クラスターを削除して最初からやり直す必要があります。 詳細については、「外部で作成されたスナップショットによる新しいクラスターのシード」を参照してください。
MemoryDB: クラススケーリングが完了しました	`"Succeeded applying modification to node type to %s."`	キャッシュクラスターのスケールアップが正常に完了しました。
MemoryDB: クラススケーリングに失敗しました	`"Failed applying modification to node type to %s."`	クラスターのスケールアップが失敗しました。

イベント名	メッセージ	説明
MemoryDB:NodeReplaceStarted	`"Recovering node %s"`	MemoryDB が、ノードを実行しているホストのパフォーマンスが低下しているか、到達できないことを検出したため、ノードの置き換えを開始しました。 **Note** 置き換えられたノードの DNS エントリは変更されません。 ほとんどのインスタンスでは、このイベントが発生したときにクライアントのサーバーリストを更新する必要はありません。ただし、一部のクライアントライブラリは、MemoryDB がノードを置き換えた後でもノードの使用を停止する可能性があります。この場合、このイベントが発生したとき、アプリケーションがサーバーリストを更新する必要があります。

イベント名	メッセージ	説明
MemoryDB:NodeReplaceComplete	`"Finished recovery for node %s"`	MemoryDB が、ノードを実行しているホストのパフォーマンスが低下しているか、到達できないことを検出したため、ノードの置き換えを完了しました。 **Note** 置き換えられたノードの DNS エントリは変更されません。 ほとんどのインスタンスでは、このイベントが発生したときにクライアントのサーバーリストを更新する必要はありません。ただし、一部のクライアントライブラリは、MemoryDB がノードを置き換えた後でもノードの使用を停止する可能性があります。この場合、このイベントが発生したとき、アプリケーションがサーバーリストを更新する必要があります。
MemoryDB:CreateClusterComplete	`"Cluster created"`	クラスターが正常に作成されました。

イベント名	メッセージ	説明
MemoryDB:CreateClusterFailed	"Failed to create cluster due to unsuccessful creation of its node(s)." および "Deleting all nodes belonging to this cluster."	クラスターは作成されませんでした。
MemoryDB:DeleteClusterComplete	"Cluster deleted."	クラスターと関連するすべてのアプリケーションノードの削除が完了しました。
MemoryDB:FailoverComplete	"Failover to replica node %s completed"	レプリカノードへのフェイルオーバーが成功しました。
MemoryDB:NodeReplacementCanceled	"The replacement of node %s which was scheduled during the maintenance window from start time: %s, end time: %s has been canceled"	置き換え対象となっていたクラスター内のノードが置き換え対象ではなくなりました。
MemoryDB:NodeReplacementRescheduled	"The replacement in maintenance window for node %s has been re-scheduled from previous start time: %s, previous end time: %s to new start time: %s, new end time: %s"	以前置き換え対象になったクラスター内のノードのスケジュールが、通知に記載されている新しい期間に変更されました。 実行可能なアクションについては、「ノードの置換」を参照してください。

イベント名	メッセージ	説明
MemoryDB:NodeReplacementScheduled	`"The node %s is scheduled for replacement during the maintenance window from start time: %s to end time: %s"`	クラスター内のノードが、通知に記載されている期間中の置き換え対象となりました。 実行可能なアクションについては、「ノードの置換」を参照してください。
MemoryDB:RemoveNodeComplete	`"Removed node %s"`	ノードがクラスターから削除されました。
MemoryDB:SnapshotComplete	`"Snapshot %s succeeded for node %s"`	スナップショットの作成が正常に完了しました。
MemoryDB:SnapshotFailed	`"Snapshot %s failed for node %s"`	スナップショットが失敗しました。詳細な原因については、クラスターのイベントを参照してください。 DescribeSnapshotsを参照してスナップショットを記述する場合は、ステータスは`failed`です。

AWS CloudTrail で MemoryDB API 呼び出しを口グに記録する

MemoryDB は AWS CloudTrail と統合されており、これにより、ユーザー、ロール、または AWS サービスによって MemoryDB で実行されたアクションの記録を提供しています。CloudTrail は、MemoryDB のコンソールからの呼び出しや、MemoryDB API オペレーションへのコードからの呼び出しなど、MemoryDB のすべての API 呼び出しをイベントとしてキャプチャします。証跡を作成する場合は、MemoryDB のイベントを含む CloudTrail イベントを Amazon S3 バケットに継続的に配信できます。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表示できます。CloudTrail によって収集された情報を使用して、MemoryDB に対して行われたリクエスト、リクエストが行われた IP アドレス、リクエストを行った人、リクエストが行われた日時、その他の詳細を特定できます。

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 の MemoryDB 情報

AWS アカウントを作成すると、そのアカウントに対して CloudTrail が有効になります。アクティビティが MemoryDB で発生すると、そのアクティビティは [イベント履歴] にある他の AWS サービスイベントと共に CloudTrail イベントに記録されます。最近のイベントは、AWSアカウントで表示、検索、ダウンロードできます。詳細については、「CloudTrail イベント履歴でのイベントの表示」を参照してください。

MemoryDB のイベントなど、AWS アカウントのイベントの継続的な記録については、証跡を作成します。[Trail] (追跡) により、CloudTrail はログファイルを Simple Storage Service (Amazon S3) バケットに配信できます。デフォルトでは、コンソールで追跡を作成するときに、追跡がすべてのリージョンに適用されます。証跡は AWS パーティションのすべてのリージョンからのイベントを記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づいて対応するため、他の AWS サービスを構成できます。詳細については、次を参照してください:

- 証跡の作成のための概要
- CloudTrail がサポートするサービスと統合
- CloudTrail 用 Amazon SNS 通知の構成
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべての MemoryDB アクションは、CloudTrail により記録されます。例えば、`CreateCluster`、`DescribeClusters`、`UpdateCluster` 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。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。
- 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どうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、「CloudTrail userIdentity 要素」を参照してください。

MemoryDB のログファイルエントリについて

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントはあらゆるソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストのパラメータなどの情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次は、`CreateCluster` アクションを示す CloudTrail ログエントリの例です。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T17:56:46Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "CreateCluster",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.create-cluster",
    "requestParameters": {
        "clusterName": "memorydb-cluster",
        "nodeType": "db.r6g.large",
        "subnetGroupName": "memorydb-subnet-group",
        "aCLName": "open-access"
    },
    "responseElements": {
        "cluster": {
            "name": "memorydb-cluster",
            "status": "creating",
            "numberOfShards": 1,
            "availabilityMode": "MultiAZ",
            "clusterEndpoint": {
                "port": 6379
            },
            "nodeType": "db.r6g.large",
```

```
            "engineVersion": "6.2",
            "enginePatchVersion": "6.2.6",
            "parameterGroupName": "default.memorydb-redis6",
            "parameterGroupStatus": "in-sync",
            "subnetGroupName": "memorydb-subnet-group",
            "tLSEnabled": true,
            "aRN": "arn:aws:memorydb:us-east-1:123456789012:cluster/memorydb-cluster",
            "snapshotRetentionLimit": 0,
            "maintenanceWindow": "tue:06:30-tue:07:30",
            "snapshotWindow": "09:00-10:00",
            "aCLName": "open-access",
            "dataTiering": "false",
            "autoMinorVersionUpgrade": true
        }
    },
    "requestID": "506fc951-9ae2-42bb-872c-98028dc8ed11",
    "eventID": "2ecf3dc3-c931-4df0-a2b3-be90b596697e",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

以下の例は、DescribeClusters アクションを示す CloudTrail ログエントリです。MemoryDB Describe および List 呼び出し (Describe* および List*) では、responseElements セクションが削除され、null として表示されることに注意してください。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T18:39:51Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "DescribeClusters",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
```

```
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.describe-clusters",
    "requestParameters": {
        "maxResults": 50,
        "showShardDetails": true
    },
    "responseElements": null,
    "requestID": "5e831993-52bb-494d-9bba-338a117c2389",
    "eventID": "32a3dc0a-31c8-4218-b889-1a6310b7dd50",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

次の例では UpdateCluster アクションを記録する CloudTrail のログエントリを示します。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T19:23:20Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "UpdateCluster",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.update-cluster",
    "requestParameters": {
        "clusterName": "memorydb-cluster",
        "snapshotWindow": "04:00-05:00",
        "shardConfiguration": {
            "shardCount": 2
        }
    },
    "responseElements": {
```

```
        "cluster": {
            "name": "memorydb-cluster",
            "status": "updating",
            "numberOfShards": 2,
            "availabilityMode": "MultiAZ",
            "clusterEndpoint": {
                "address": "clustercfg.memorydb-cluster.cde8da.memorydb.us-
east-1.amazonaws.com",
                "port": 6379
            },
            "nodeType": "db.r6g.large",
            "engineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "parameterGroupName": "default.memorydb-redis6",
            "parameterGroupStatus": "in-sync",
            "subnetGroupName": "memorydb-subnet-group",
            "tLSEnabled": true,
            "aRN": "arn:aws:memorydb:us-east-1:123456789012:cluster/memorydb-cluster",
            "snapshotRetentionLimit": 0,
            "maintenanceWindow": "tue:06:30-tue:07:30",
            "snapshotWindow": "04:00-05:00",
            "autoMinorVersionUpgrade": true,
            "DataTiering": "false"
        }
    },
    "requestID": "dad021ce-d161-4365-8085-574133afab54",
    "eventID": "e0120f85-ab7e-4ad4-ae78-43ba15dee3d8",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

以下の例は、CreateUser アクションを示す CloudTrail ログエントリです。機密データを含む MemoryDB の呼び出しの場合、そのデータは以下の requestParameters セクションに示されるように、対応する CloudTrail イベントで削除されることに注意してください。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
```

```
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T19:56:13Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "CreateUser",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.create-user",
    "requestParameters": {
        "userName": "memorydb-user",
        "authenticationMode": {
            "type": "password",
            "passwords": [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "accessString": "~* &* -@all +@read"
    },
    "responseElements": {
        "user": {
            "name": "memorydb-user",
            "status": "active",
            "accessString": "off ~* &* -@all +@read",
            "aCLNames": [],
            "minimumEngineVersion": "6.2",
            "authentication": {
                "type": "password",
                "passwordCount": 1
            },
            "aRN": "arn:aws:memorydb:us-east-1:123456789012:user/memorydb-user"
        }
    },
    "requestID": "ae288b5e-80ab-4ff8-989a-5ee5c67cd193",
    "eventID": "ed096e3e-16f1-4a23-866c-0baa6ec769f6",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
```

```
}
```

MemoryDB のコンプライアンス検証

サードパーティーの監査者は、さまざまな AWS コンプライアンスプログラムの一環として MemoryDB 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、以下が含まれます。

- Payment Card Industry Data Security Standard (PCI DSS)。詳細については、「PCI DSS」を参照してください。
- 医療保険の相互運用性と説明責任に関する法律の事業提携契約 (HIPAA BAA)。詳細については、「HIPAA コンプライアンス」を参照してください。
- System and Organization Controls (SOC) 1、2、および 3。詳細については、「SOC」を参照してください。
- Federal Risk and Authorization Management Program (FedRAMP) Moderate。詳細については、「FedRAMP」を参照してください。
- ISO/IEC 27001:2013、27017:2015、27018:2019、および ISO/IEC 9001:2015。詳細については、「AWS ISO and CSA STAR certifications and services」を参照してください。

特定のコンプライアンスプログラムの対象範囲に含まれる AWS の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。

AWS Artifact を使用して、サードパーティーの監査レポートをダウンロードできます。詳細については、「AWS Artifact でレポートをダウンロードする」を参照してください。

MemoryDB を使用する際のユーザーのコンプライアンス責任は、ユーザーのデータの機密性や貴社のコンプライアンス目的、適用される法律および規制によって決まります。AWS では、コンプライアンスに役立つ以下のリソースを提供しています。

- セキュリティおよびコンプライアンスのクイックスタートガイド – これらの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 でデプロイするための手順を説明します。
- AWS コンプライアンスリソース – このワークブックおよびガイドのコレクションは、お客様の業界と拠点に適用される場合があります。
- AWS Config デベロッパーガイドの「ルールでのリソースの評価」– AWS Config は、リソース設定が、社内のプラクティス、業界のガイドラインそして規制にどの程度適合しているのかを評価します。

- AWS Security Hub CSPM – この AWS のサービスは、AWS 内でのユーザーのセキュリティ状態に関する包括的な見解を提供し、業界のセキュリティ標準、およびベストプラクティスに対するコンプライアンスを確認するために役立ちます。
- AWS Audit Manager – この AWS サービスでは、AWS の使用状況を継続的に監査し、リスクの管理方法と、規制や業界標準へのコンプライアンスの管理方法を簡素化できます。

MemoryDB のインフラストラクチャセキュリティ

マネージドサービスである MemoryDB は、ホワイトペーパー「Amazon Web Services: セキュリティプロセスの概要」に記載されている AWS グローバルネットワークセキュリティの手順で保護されています。

AWS が発行している API コールを使用して、ネットワーク経由で MemoryDB にアクセスします。クライアントは、Transport Layer Security (TLS) 1.2 以降をサポートする必要があります。TLS 1.3 以降が推奨されます。また、一時的ディフィー・ヘルマン Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポートされている必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストは、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token ServiceAWS STSを使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

ネットワーク間のトラフィックのプライバシー

MemoryDB では、以下の方法によりデータを不正なアクセスからセキュリティで保護します。

- MemoryDB と Amazon VPC では、インストールに必要なセキュリティグループのタイプを説明します。
- MemoryDB API とインターフェイス VPC エンドポイント (AWS PrivateLink) は VPC と MemoryDB API エンドポイントの間にプライベート接続を確立できます。
- MemoryDB でのアイデンティティとアクセス権の管理 は、ユーザー、グループ、グループ、ロールの付与と制限のためのものです。

MemoryDB と Amazon VPC

Amazon Virtual Private Cloud (Amazon VPC) サービスは、従来のデータセンターに非常によく似た仮想ネットワークを定義します。お客様が Amazon VPCで仮想プライベートクラウド (VPC) を設定すると、IP アドレス範囲の選択、サブネットの作成、ルートテーブル、ネットワークゲートウェイ、セキュリティの設定などが可能になります。仮想ネットワークにクラスターを追加でき、Amazon VPC のセキュリティグループを使用して、クラスターへのアクセスを制御できます。

このセクションでは、VPC 内で手動で MemoryDB クラスターを設定する方法を説明します。この情報は、MemoryDB と Amazon VPC との連携について理解を深めたいユーザーを対象としています。

トピック

- MemoryDB と VPC について
- Amazon VPC の MemoryDB クラスターにアクセスするためのアクセスパターン
- Virtual Private Cloud (VPC) の作成

MemoryDB と VPC について

MemoryDB は Amazon VPC と完全に統合されています。MemoryDB ユーザーにとって、これは次のことを意味します。

- MemoryDB は常に VPC でクラスターを起動します。
- AWS を初めて使用する場合は、デフォルト VPC が自動的に作成されます。
- デフォルト VPC をお持ちのお客様が、クラスター起動時にサブネットを指定しなかった場合は、そのクラスターはお客様のデフォルト Amazon VPC で起動されます。

詳細については、「サポートされているプラットフォームとデフォルト VPC があるかどうかを確認する」を参照してください。

Amazon VPCを使用することによって、従来のデータセンターに非常によく似た仮想ネットワークを AWS クラウド内に作成できます。お客様の VPC はお客様が設定できます。例えば、IP アドレス範囲の選択、サブネットの作成、ルートテーブル、ネットワークゲートウェイ、セキュリティの設定などが可能です。

MemoryDB では、ソフトウェアのアップグレード、パッチ、障害検出、および復旧を管理します。

VPC での MemoryDB の概要

- VPC は、独自の IP アドレスのブロックが割り当てられた AWS クラウドの独立した部分です。
- インターネットゲートウェイは VPC を直接インターネットに接続し、他の AWS リソースへのアクセスを提供します。それには、VPC の外部で実行されている Amazon Simple Storage Service (Amazon S3) などのリソースが含まれます。
- VPC サブネットは、セキュリティおよび運用上のニーズに合わせて AWS リソースを分離できる Amazon VPC の IP アドレス範囲のセグメントです。
- Amazon VPC セキュリティグループは、MemoryDB クラスターと Amazon EC2 インスタンスのインバウンドとアウトバウンドのトラフィックを制御します。
- サブネットで MemoryDB クラスターを起動できます。ノードは、サブネットのアドレス範囲のプライベート IP アドレスを持ちます。
- サブネットで Amazon EC2 インスタンスを起動することもできます。各 Amazon EC2 インスタンスはサブネットのアドレス範囲内のプライベート IP アドレスを持ちます。Amazon EC2 インスタンスは、同じサブネット内のすべてのノードに接続できます。
- インターネットからアクセス可能な VPC 内の Amazon EC2 インスタンスの場合は、インスタンスに Elastic IP アドレスと呼ばれる静的なパブリックアドレスを割り当てる必要があります。

前提条件

VPC 内に MemoryDB クラスターを作成するには、VPC が次の要件を満たしている必要があります。

- VPCは、専用ではない Amazon EC2 インスタンスを許可する必要があります。ハードウェア専有インスタンスのテナンシー用に設定された VPC では、MemoryDB を使用できません。
- VPC 用にサブネットグループを定義する必要があります。MemoryDB はそのキャッシュサブネットグループを使用して、そのサブネット内でノードに関連付けるサブネットおよび IP アドレスを選択します。
- VPC 用にセキュリティグループを定義する必要があります。または、用意されているデフォルトを使用できます。
- 各サブネットの CIDR ブロックは、メンテナンス作業で使用する予備の IP アドレスを MemoryDB に提供するのに十分な大きさが必要です。

ルーティングとセキュリティ

VPC でルーティングを設定して、トラフィックの送信先 (インターネットゲートウェイ、仮想プライベートゲートウェイなど) を制御できます。インターネットゲートウェイの場合、VPC は、同じ VPC で実行されているのではない他の AWS リソースに直接アクセスできます。お客様の組織のローカルネットワークに接続された仮想プライベートゲートウェイのみを選択した場合、VPN 経由でインターネット宛てのトラフィックをルーティングし、ローカルセキュリティポリシーとファイアウォールを使用して出口を制御できます。この場合、インターネット経由で AWS リソースにアクセスする際に、追加の帯域幅料金が発生します。

Amazon VPC セキュリティグループを使用して、Amazon VPC 内の MemoryDB クラスターと Amazon EC2 インスタンスをセキュリティで保護することができます。セキュリティグループは、サブネットレベルでなくインスタンスレベルでファイアウォールのように動作します。

> **Note**
>
> 基礎となる IP アドレスは変わる可能性があるため、ノードに接続するには DNS 名を使用することを強くお勧めします。

Amazon VPC ドキュメント

Amazon VPC に関するドキュメントには、Amazon VPC の作成および使用方法について説明する独自のドキュメントがあります。Amazon VPC ガイドの情報の参照先について以下の表にまとめます。

説明	ドキュメント
Amazon VPC の使用を開始する方法	Amazon VPC の開始方法
AWS マネジメントコンソール を通じて Amazon VPC を使用する方法	Amazon VPC User Guide
すべての Amazon VPC コマンドの詳細説明	Amazon EC2 コマンドラインリファレンス (Amazon VPC コマンドは、Amazon EC2 リファレンスに記載されています)
Amazon VPC API オペレーション、データタイプ、およびエラーの詳細説明	Amazon EC2 API リファレンス (Amazon VPC API オペレーションは、Amazon EC2 リファレンスに記載されています)
オプションとして IPsec VPN 接続のゲートウェイを設定する必要のあるネットワーク管理者向け情報	AWS Site-to-Site VPN とは

Amazon Virtual Private Cloud の詳細については、「Amazon Virtual Private Cloud」を参照してください。

Amazon VPC の MemoryDB クラスターにアクセスするためのアクセスパターン

MemoryDB は、Amazon VPC 内のクラスターにアクセスするための以下のシナリオをサポートしています。

目次

- MemoryDB クラスターと Amazon EC2 インスタンスが同じ Amazon VPC にある場合の MemoryDB クラスターへのアクセス
- MemoryDB クラスターと Amazon EC2 インスタンスが異なる Amazon VPC にある場合のアクセス
  - MemoryDB クラスターと Amazon EC2 インスタンスが同じリージョン内の異なる Amazon VPC にある場合のアクセス
    - トランジット・ゲートウェイ の使用
  - MemoryDB クラスターと Amazon EC2 インスタンスが異なるリージョン内の異なる Amazon VPC にある場合のアクセス
    - トランジット VPC の使用
- 顧客のデータセンター内で実行されるアプリケーションからの MemoryDB クラスターへのアクセス
  - 顧客のデータセンター内で実行されるアプリケーションからの VPN 接続を使用した MemoryDB クラスターへのアクセス
  - 顧客のデータセンター内で実行されるアプリケーションからの Direct Connect を使用した MemoryDB クラスターへのアクセス

MemoryDB クラスターと Amazon EC2 インスタンスが同じ Amazon VPC にある場合の MemoryDB クラスターへのアクセス

最も一般的ユースケースは、EC2 インスタンスにデプロイされたアプリケーションが同じ VPC のクラスターに接続する必要がある場合です。

同じ VPC 内の EC2 インスタンスとクラスター間のアクセスを管理する方法として最も簡単なのは、次の方法です。

1. クラスターの VPC セキュリティグループを作成します。このセキュリティグループを使用して、クラスターへのアクセスを制限できます。例えば、クラスターを作成したときに割り当てたポートと、クラスターにアクセスするのに使用する IP アドレスを使用して TCP へのアクセスを許可する、このセキュリティグループのカスタムルールを作成できます。

MemoryDB クラスターのデフォルトのポートは 6379 です。

2. EC2 インスタンス (ウェブサーバーとアプリケーションサーバー) 用の VPC セキュリティグループを作成します。このセキュリティグループは、必要に応じて VPC のルーティングテーブルを介してインターネットから EC2 インスタンスへのアクセスを許可できます。例えば、ポート 22 経由で EC2 インスタンスへの TCP アクセスを許可するルールをこのセキュリティグループに設定できます。
3. EC2 インスタンス用に作成したセキュリティグループからの接続を許可するクラスターのセキュリティグループで、カスタムルールを作成します。これは、セキュリティグループのメンバーにクラスターへのアクセスを許可します。

他のセキュリティグループからの接続を許可する VPC セキュリティグループでルールを作成するには

1. AWS マネジメントコンソールにサインインして、Amazon VPC コンソール (https://console.aws.amazon.com/vpc) を開きます。
2. 左のナビゲーションペインで セキュリティグループを選択します。
3. クラスターに使用するセキュリティグループを選択または作成します。インバウンドルールで、インバウンドルールの編集 を選択し、ルールの追加 を選択します。このセキュリティグループは、他のセキュリティグループのメンバーへのアクセスを許可します。
4. Type で Custom TCP Rule を選択します。

   a. Port Range ポートには、クラスター作成時に使用したポートを指定します。

      MemoryDB クラスターのデフォルトのポートは 6379 です。

   b. ソース ボックスに、セキュリティグループの ID の入力を開始します。リストから、Amazon EC2 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。
5. 終了したら、保存 を選択します。

MemoryDB クラスターと Amazon EC2 インスタンスが異なる Amazon VPC にある場合のアクセス

クラスターにアクセスするために使用している EC2 インスタンスとは別の VPC にクラスターがある場合、クラスターにアクセスするにはいくつかの方法がある。クラスターとEC2インスタンスが異なるVPCにあるが、同じリージョンにある場合は、VPCピアリングを使用できる。クラスターとEC2インスタンスが異なるリージョンにある場合、リージョン間でVPN接続を作成できる。

トピック

- MemoryDB クラスターと Amazon EC2 インスタンスが同じリージョン内の異なる Amazon VPC にある場合のアクセス
- MemoryDB クラスターと Amazon EC2 インスタンスが異なるリージョン内の異なる Amazon VPC にある場合のアクセス

MemoryDB クラスターと Amazon EC2 インスタンスが同じリージョン内の異なる Amazon VPC にある場合のアクセス

同じリージョンの異なる Amazon VPC で Amazon EC2 インスタンスによってアクセスされるクラスター - VPC ピア接続

VPC ピア接続は、プライベート IP アドレスを使用して 2 つの VPC 間でトラフィックをルーティングすることを可能にするネットワーク接続です。どちらの VPC のインスタンスも、同じネットワーク内に存在しているかのように、相互に通信できます。VPC ピア接続は、自分の Amazon VPC 間、または、1 つのリージョン内の他の AWS アカウントにある Amazon VPC との間に作成できます。Amazon VPC ピア接続の詳細については、「VPC ドキュメント」を参照してください。

ピア接続経由で別の Amazon VPC のクラスターにアクセスするには

1. 2 つの VPC に、重複する IP 範囲がないことを確認します。重複する IP 範囲がある場合、それらをピア接続することができません。
2. 2 つの VPC をピア接続します。詳細については、「VPC ピア接続の作成と使用」を参照してください。
3. ルーティングテーブルを更新します。詳細については、「VPC ピア接続のルートテーブルを更新する」を参照してください
4. MemoryDBクラスターのセキュリティグループを変更し、ピアリングされたVPCのApplication Security Groupからのインバウンド接続を許可します。詳細については、「ピア VPC セキュリティグループの参照」を参照してください。

ピア接続によりクラスターにアクセスすると、追加のデータ転送コストが発生します。

トランジット・ゲートウェイ の使用

トランジットゲートウェイを使用すると、同じ AWS リージョンに VPC と VPN 接続をアタッチして、それらの間でトラフィックをルーティングできます。トランジットゲートウェイは AWS アカウント間で機能し、AWS Resource Access Manager を使用してトランジットゲートウェイを他のアカウントと共有できます。他のAWSアカウントとトランジットゲートウェイを共有した後、アカウントの所有者はトランジットゲートウェイにそれらの VPC をアタッチすることができます。どちらのアカウントのユーザーも、アタッチメントをいつでも削除できます。

トランジットゲートウェイでマルチキャストを有効にしてから、ドメインに関連付ける VPC アタッチメントを介してマルチキャストソースからマルチキャストグループメンバーにマルチキャストトラフィックを送信できるようにする トランジットゲートウェイマルチキャストドメインを作成できます。

また、異なるAWSリージョンでトランジットゲートウェイ間にピア接続アタッチメントを作成することもできます。これにより、異なるリージョン間でトランジットゲートウェイのアタッチメント間でトラフィックをルーティングできます。

詳細については、「トランジットゲートウェイ」を参照してください。

MemoryDB クラスターと Amazon EC2 インスタンスが異なるリージョン内の異なる Amazon VPC にある場合のアクセス

トランジット VPC の使用

VPC ピアリングの代わりに使用する、複数の、地理的に離れた VPC とリモートネットワークを接続する別の一般的な方法は、グローバルなネットワーク中継センターとして機能する中継 VPC の作成です。中継 VPC はネットワーク管理を単純化して、複数の VPC とリモートのネットワークを接続するために必要な接続数を最小限に抑えます。この設計は、コロケーション中継ハブを物理的に設立したり、物理的なネットワーク設備をデプロイしたりするための従来の費用をほとんどかけずに実装できるため、時間と労力を節約し、コストも削減できます。

異なるリージョンの異なる VPC 間での接続

Transit Amazon VPCが確立されると、あるリージョンの "スポーク "VPCにデプロイされたアプリケーションは、別のリージョン内の "スポーク "VPCにあるMemoryDBクラスターに接続することができます。

別の AWS リージョン内の異なる VPC のクラスターにアクセスするには

1. Transit VPC ソリューションをデプロイします。詳細については、「AWS トランジットゲートウェイ」を参照してください。
2. アプリとVPCのVPCルーティングテーブルを更新して、VGW (Virtual Private Gateway) とVPNアプライアンスを経由するトラフィックをルーティングします。ボーダーゲートウェイプロトコル (BGP) を使用した動的ルーティングの場合、ルートは自動的に伝達される可能性があります。
3. MemoryDBクラスターのセキュリティグループを変更して、アプリケーションインスタンスの IP 範囲からのインバウンド接続を許可します。このシナリオでは、アプリケーションサーバーセキュリティグループを参照することはできません。

リージョン間でクラスターにアクセスすると、ネットワークのレイテンシーが生じ、リージョン間のデータ転送コストが追加で発生します。

顧客のデータセンター内で実行されるアプリケーションからの MemoryDB クラスターへのアクセス

もう 1 つのシナリオとして、クライアントまたは顧客のデータセンター内のアプリケーションが VPC の MemoryDB クラスターにアクセスする必要がある場合のようなハイブリッドアーキテクチャが考えられます。このシナリオは、顧客の VPC とデータセンター間で VPN または Direct Connect による接続がある場合にサポートされます。

トピック

- 顧客のデータセンター内で実行されるアプリケーションからの VPN 接続を使用した MemoryDB クラスターへのアクセス
- 顧客のデータセンター内で実行されるアプリケーションからの Direct Connect を使用した MemoryDB クラスターへのアクセス

顧客のデータセンター内で実行されるアプリケーションからの VPN 接続を使用した MemoryDB クラスターへのアクセス

VPN によるデータセンターから MemoryDB への接続

VPN 接続経由でオンプレミスアプリケーションから VPC のクラスターにアクセスするには

1. VPC にハードウェア仮想プライベートゲートウェイを追加して、VPN 接続を確立します。詳細については、「VPC へのハードウェア仮想プライベートゲートウェイの追加」を参照してください。
2. MemoryDB クラスターがデプロイされているサブネットの VPC ルーティングテーブルを更新して、オンプレミスアプリケーションサーバーからのトラフィックを許可します。BGP を使用した動的ルーティングの場合、ルートは自動的に伝達される可能性があります。
3. MemoryDB クラスターのセキュリティグループを変更して、オンプレミスアプリケーションサーバーからのインバウンド接続を許可します。

VPN 接続経由でクラスターにアクセスすると、ネットワークのレイテンシーが生じ、追加のデータ転送コストが発生します。

顧客のデータセンター内で実行されるアプリケーションからの Direct Connect を使用した MemoryDB クラスターへのアクセス

Direct Connect によるデータセンターから MemoryDB への接続

Direct Connect を使用して、ネットワークで実行されるアプリケーションから MemoryDB クラスターにアクセスするには

1. Direct Connect 接続を確立します。詳細については、「AWS Direct Connect 入門ガイド」を参照してください。
2. MemoryDB クラスターのセキュリティグループを変更して、オンプレミスアプリケーションサーバーからのインバウンド接続を許可します。

DX 接続経由でクラスターにアクセスすると、ネットワークのレイテンシーが生じ、追加のデータ転送料金が発生する場合があります。

Virtual Private Cloud (VPC) の作成

この例では、各アベイラビリティーゾーンのプライベートサブネットを持つ Amazon VPCサービスに基づいて仮想プライベートクラウド (VPC) を作成します。

VPC の作成 (コンソール)

Amazon Virtual Private Cloud 内に MemoryDB キャッシュクラスターを作成するには

1. AWS マネジメントコンソールにサインインして Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. VPC ダッシュボードで、[VPC を作成] を選択します。
3. Resources to create (作成するリソース) で、VPC only (VPC など) を選択します。
4. Number of Availability Zones (AZs) (アベイラビリティーゾーンの数 (AZ)) で、サブネットを起動するアベイラビリティーゾーンの数を選択します。
5. Number of public subnets (パブリックサブネットの数) で、VPC に追加するパブリックサブネットの数を選択します。
6. Number of private subnets (プライベートサブネットの数) で、VPC に追加するプライベートサブネットの数を選択します。

   > **Tip**
   >
   > サブネットの識別子と、どちらがパブリックで、どちらがプライベートであるかを書き留めておきます。この情報は、後でクラスターを起動し、Amazon VPC に Amazon EC2 インスタンスを追加するときに必要になります。

7. Amazon VPC セキュリティグループを作成します。クラスターと Amazon EC2 インスタンスでは、このグループを使用します。

   a. AWS マネジメントコンソール の左のナビゲーションペインで [セキュリティグループ] を選択します。
   b. Create Security Group (セキュリティグループの作成) を選択します。
   c. 対応するボックスにセキュリティグループの名前と説明を入力します。[VPC] の ID を選択します。
   d. すべての設定が正しいことを確認したら、Yes, Create を選択します。

8. セキュリティグループのネットワーク Ingress ルールを定義します。このルールは、Secure Shell (SSH) を使用して Amazon EC2 インスタンスに接続することを許可します。

a. 左のナビゲーションペインで セキュリティグループを選択します。

b. リストで対象となるセキュリティグループを探して選択します。

c. Security Group の下で、Inbound タブを選択します。Create a new rule ボックスで、SSH を選択し、Add Rule を選択します。

   新しいインバウンドルールに次の値を設定して、HTTP へのアクセスを許可します。

   - Type: HTTP
   - ソース: 0.0.0.0/0

d. 新しいインバウンドルールに次の値を設定して、HTTP へのアクセスを許可します。

   - Type: HTTP
   - ソース: 0.0.0.0/0

   Apply Rule Changes を選択します。

これで、VPC内にサブネットグループを作成し、クラスターを作成する準備が整いました。

サブネットおよびサブネットグループ

サブネットグループ は、Amazon Virtual Private Cloud (VPC) 環境で実行しているクラスターに対して指定できるサブネット (通常はプライベート) の集合です。

Amazon VPC でクラスターを作成する場合、サブネットグループを指定するか、デフォルトで提供されるサブネットグループを使用できます。MemoryDB はそのキャッシュサブネットグループを使用して、そのサブネット内でノードに関連付けるサブネットおよび IP アドレスを選択します。

このセクションでは、サブネットおよびサブネットグループを作成し活用して、MemoryDB リソースへのアクセスを管理する方法を扱います。

Amazon VPC 環境でのサブネットグループの使用方法の詳細については、「ステップ 3: クラスターへのアクセスの許可」を参照してください。

サポートされている MemoryDB AZ ID

リージョン名/リージョン	サポートされる AZ ID		
米国東部 (オハイオ) リージョン us-east-2	use2-az1, use2-az2, use2-az3		
米国東部(バージニア州北部) リージョン us-east-1	use1-az1, use1-az2, use1-az4, use1-az5, use1-az6		
US West (N. California) リージョン us-west-1	usw1-az1, usw1-az2, usw1-az3		
米国西部 (オレゴン) リージョン us-west-2	usw2-az1, usw2-az2, usw2-az3, usw2-az4		
カナダ (中部) リージョン ca-central-1	cac1-az1, cac1-az2, cac1-az4		
アジアパシフィック (香港) リージョン ap-east-1	ape1-az1, ape1-az2, ape1-az3		
アジアパシフィック (ムンバイ) リージョン ap-south-1	aps1-az1, aps1-az2, aps1-az3		

リージョン名/リージョン	サポートされる AZ ID		
アジアパシフィック (東京) リージョン ap-northeast-1	apne1-az1, apne1-az2, apne1-az4		
Asia Pacific (Seoul) Region ap-northeast-2	apne2-az1, apne2-az2, apne2-az3		
アジアパシフィック (シンガポール) リージョン ap-southeast-1	apse1-az1, apse1-az2, apse1-az3		
アジアパシフィック (シドニー) リージョン ap-southeast-2	apse2-az1, apse2-az2, apse2-az3		
欧州 (フランクフルト) リージョン eu-central-1	euc1-az1, euc1-az2, euc1-az3		
欧州 (アイルランド) リージョン eu-west-1	euw1-az1, euw1-az2, euw1-az3		
欧州 (ロンドン) リージョン eu-west-2	euw2-az1, euw2-az2, euw2-az3		

リージョン名/リージョン	サポートされる AZ ID		
欧州 (パリ) リージョン eu-west-3	euw3-az1, euw3-az2, euw3-az3		
欧州 (ストックホルム) リージョン eu-north-1	eun1-az1, eun1-az2, eun1-az3		
欧州 (ミラノ) リージョン eu-south-1	eus1-az1, eus1-az2, eus1-az3		
南米 (サンパウロ) リージョン sa-east-1	sae1-az1, sae1-az2, sae1-az3		
中国 (北京) リージョン cn-north-1	cnn1-az1, cnn1-az2		
中国 (寧夏) リージョン cn-northwest-1	cnw1-az1, cnw1-az2, cnw1-az3		
us-gov-east-1	usge1-az1, usge1-az2, usge1-az3		
us-gov-west-1	usgw1-az1, usgw1-az2, usgw1-az3		

リージョン名/リージョン	サポートされる AZ ID		
欧州 (スペイン) リージョン eu-south-2	eus2-az1, eus2-az2, eus2-az3		

トピック

- MemoryDB と IPV6
- サブネットグループの作成
- サブネットグループの更新
- サブネットグループの詳細の表示
- サブネットグループの削除

MemoryDB と IPV6

デュアルスタックおよび ipv6 専用サブネットをサブネットグループに提供することで、Valkey エンジンと Redis OSS エンジンの両方を備えた新しいデュアルスタックおよび ipv6 専用クラスターを作成できます。既存のクラスターのネットワークタイプを変更することはできません。

この機能を使用すると、次のことができます。

- デュアルスタックサブネットに ipv4 専用およびデュアルスタッククラスターを作成する。
- ipv6 専用サブネットに ipv6 専用クラスターを作成する。
- ipv4 専用、デュアルスタック、および ipv6 専用サブネットをサポートする新しいサブネットグループを作成する。
- 既存のサブネットグループを、基盤となる VPC の追加のサブネットを含むように変更する。
- サブネットグループの既存のサブネットを変更する
  - IPv6 専用サブネットを、IPv6 用に設定されたサブネットグループに追加する
  - IPv4 およびデュアルスタックをサポートするように設定されたサブネットグループに IPv4 またはデュアルスタックサブネットを追加する
- デュアルスタックおよび ipv6 クラスター用のエンジン検出コマンドを使用して、ipv4 アドレスまたは ipv6 アドレスを持つクラスター内のノードをすべて検出する。このような検出コマンドには、`redis_info` や `redis_cluster` などがあります。
- デュアルスタックおよび ipv6 クラスター用の DNS 検出コマンドを使用して、クラスター内のすべてのノードの ipv4 アドレスと ipv6 アドレスを検出する。

サブネットグループの作成

新しいサブネットグループを作成する場合は、使用可能な IP アドレス数に注意してください。サブネットの空き IP アドレス数が非常に少ない場合は、クラスターに追加できるノード数が制約される可能性があります。この問題を解決するために、クラスターのアベイラビリティーゾーンで十分な数の IP アドレスを使用できるように、サブネットグループに 1 つ以上のサブネットを割り当てることができます。その後で、クラスターにノードを追加できます。

以下の手順では、`mysubnetgroup` (コンソール)、AWS CLI、および MemoryDB API というサブネットグループを作成する方法を示します。

サブネットグループの作成 (コンソール)

次の手順では、サブネットグループ (コンソール) を作成する方法を示します。

サブネットグループ (コンソール) を作成するには

1. AWS マネジメントコンソールにサインインし、 (https://console.aws.amazon.com/memorydb/) MemoryDBのコンソールを開きます。
2. 左の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. Create Subnet Group を選択します。
4. [サブネットグループの作成] ウィンドウで次の操作を行います。

   a. Name ボックスにサブネットグループの名前を入力します。

      クラスターの命名に関する制約は次のとおりです。

      - 1~40 個の英数字またはハイフンを使用する必要があります。
      - 先頭は文字を使用する必要があります。
      - 連続する 2 つのハイフンを含めることはできません。
      - ハイフンで終わることはできません。

   b. Description ボックスにサブネットグループの説明を入力します。

   c. VPC ID ボックスで、作成した Amazon VPC を選択します。まだ作成していない場合は、[VPC を作成] ボタンを選択し、手順に従って作成してください。

   d. [選択されたサブネット] でプライベートサブネットのアベイラビリティーゾーンと ID を選択し、[選択] を選択します。
5. [タグ] では、オプションでタグを適用してサブネットを検索およびフィルタリングしたり、AWS のコストを追跡したりできます。
6. すべての設定が正しいことを確認したら、[作成] を選択します。
7. 表示された確認メッセージで、Close を選択します。

MemoryDB コンソールの [サブネットグループ] のリストに新しい DB サブネットグループが表示されます。ウィンドウの下部で、サブネットグループを選択して、ウィンドウの下部で詳細 (このグループに関連付けられているすべてのサブネットなど) を確認します。

サブネットグループの作成 AWS

コマンドプロンプトで、`create-subnet-group` コマンドを使用してサブネットグループを作成します。

Linux、macOS、Unix の場合:

```
aws memorydb create-subnet-group \
    --subnet-group-name mysubnetgroup \
    --description "Testing" \
    --subnet-ids subnet-53df9c3a
```

Windows の場合:

```
aws memorydb create-subnet-group ^
    --subnet-group-name mysubnetgroup ^
    --description "Testing" ^
    --subnet-ids subnet-53df9c3a
```

このコマンドでは、次のような出力が生成されます。

```
    {
        "SubnetGroup": {
            "Subnets": [
                {
                    "Identifier": "subnet-53df9c3a",
                    "AvailabilityZone": {
                    "Name": "us-east-1a"
                    }
                }
            ],
            "VpcId": "vpc-3cfaef47",
            "Name": "mysubnetgroup",
            "ARN": "arn:aws:memorydb:us-east-1:012345678912:subnetgroup/
mysubnetgroup",
            "Description": "Testing"
        }
    }
```

詳細については、AWS CLI のトピック「create-subnet-group」を参照してください。

サブネットグループの作成 (MemoryDB API)

以下のパラメータを指定して、MemoryDB API を使用して CreateSubnetGroup を呼び出します。

- SubnetGroupName=*mysubnetgroup*
- Description=*Testing*
- SubnetIds.member.1=*subnet-53df9c3a*

サブネットグループの更新

サブネットグループの説明を更新することや、サブネットグループに関連付けられたサブネット ID のリストを変更することができます。クラスターが現在サブネットを使用している場合、サブネットグループからそのサブネット ID を削除することはできません。

次の手順では、サブネットグループを更新する方法を示します。

サブネットグループの更新 (コンソール)

サブネットグループを更新するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 左の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リストで、変更するグループを選択します。
4. [名前]、[VPC ID]、[説明] フィールドは変更できません。
5. [選択されたサブネット] セクションで [管理] をクリックし、サブネットに必要なアベイラビリティーゾーンに変更を加えます。変更を保存するには保存を選択します。

サブネットグループの更新 (AWS CLI)

コマンドプロンプトで、`update-subnet-group` コマンドを使用してサブネットグループを更新します。

Linux、macOS、Unix の場合:

```
aws memorydb update-subnet-group \
    --subnet-group-name mysubnetgroup \
    --description "New description" \
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

Windows の場合:

```
aws memorydb update-subnet-group ^
    --subnet-group-name mysubnetgroup ^
    --description "New description" ^
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

このコマンドでは、次のような出力が生成されます。

```
{
    "SubnetGroup": {
        "VpcId": "vpc-73cd3c17",
        "Description": "New description",
        "Subnets": [
            {
                "Identifier": "subnet-42dcf93a",
                "AvailabilityZone": {
                    "Name": "us-east-1a"
                }
            },
            {
                "Identifier": "subnet-48fc12a9",
                "AvailabilityZone": {
                    "Name": "us-east-1a"
                }
            }
        ],
        "Name": "mysubnetgroup",
        "ARN": "arn:aws:memorydb:us-east-1:012345678912:subnetgroup/mysubnetgroup",
    }
}
```

詳細については、AWS CLI のトピック「update-subnet-group」を参照してください。

サブネットグループの更新 (MemoryDB API)

以下のパラメータを指定して、MemoryDB API を使用して UpdateSubnetGroup を呼び出します。

- SubnetGroupName=*mysubnetgroup*
- 変更したいその他のパラメーター値。この例では、Description=*New%20description* を使用してサブネットグループの説明を変更します。

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateSubnetGroup
    &Description=New%20description
    &SubnetGroupName=mysubnetgroup
    &SubnetIds.member.1=subnet-42df9c3a
```

```
    &SubnetIds.member.2=subnet-48fc21a9
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20141201T220302Z
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

> **Note**
>
> 新しいサブネットグループを作成する場合は、使用可能な IP アドレス数に注意してください。サブネットの空き IP アドレス数が非常に少ない場合は、クラスターに追加できるノード数が制約される可能性があります。この問題を解決するために、クラスターのアベイラビリティーゾーンで十分な数の IP アドレスを使用できるように、サブネットグループに 1 つ以上のサブネットを割り当てることができます。その後で、クラスターにノードを追加できます。

サブネットグループの詳細の表示

次の手順では、サブネットグループの詳細を表示する方法を示します。

サブネットグループの詳細の表示 (コンソール)

サブネットグループの詳細を表示するには (コンソール)

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 左の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] ページで、[名前] の下にあるサブネットグループを選択するか、検索バーにサブネットグループの名前を入力します。
4. [サブネットグループ] ページで、[名前] の下にあるサブネットグループを選択するか、検索バーにサブネットグループの名前を入力します。
5. [サブネットグループ設定] では、サブネットグループの名前、説明、VPC ID、Amazon リソースネーム (ARN) を表示できます。

6. [サブネット] では、サブネットグループのアベイラビリティーゾーン、サブネット ID、CIDR ブロックを表示できます
7. [タグ] には、サブネットグループに関連付けられているすべてのタグが表示されます。

サブネットグループの詳細の表示 (AWS CLI)

コマンドプロンプトで、コマンド `describe-subnet-groups` を使用して、指定したサブネットグループの詳細の情報を表示します。

Linux、macOS、Unix の場合:

```
aws memorydb describe-subnet-groups \
    --subnet-group-name mysubnetgroup
```

Windows の場合:

```
aws memorydb describe-subnet-groups ^
    --subnet-group-name mysubnetgroup
```

このコマンドでは、次のような出力が生成されます。

```
{
  "subnetgroups": [
    {
      "Subnets": [
        {
          "Identifier": "subnet-060cae3464095de6e",
          "AvailabilityZone": {
            "Name": "us-east-1a"
          }
        },
        {
          "Identifier": "subnet-049d11d4aa78700c3",
          "AvailabilityZone": {
            "Name": "us-east-1c"
          }
        },
        {
          "Identifier": "subnet-0389d4c4157c1edb4",
          "AvailabilityZone": {
            "Name": "us-east-1d"
```

```
          }
        }
      ],
      "VpcId": "vpc-036a8150d4300bcf2",
      "Name": "mysubnetgroup",
      "ARN": "arn:aws:memorydb:us-east-1:53791xzzz7620:subnetgroup/mysubnetgroup",
      "Description": "test"
    }
  ]
}
```

すべてのサブネットグループの詳細を表示するには、サブネットグループ名を指定せずに同じコマンドを使用します。

```
aws memorydb describe-subnet-groups
```

詳細については、AWS CLI のトピック「describe-subnet-groups」を参照してください。

サブネットグループの表示 (MemoryDB API)

以下のパラメータを指定して、MemoryDB API を使用して DescribeSubnetGroups を呼び出します。

SubnetGroupName=*mysubnetgroup*

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateSubnetGroup
    &Description=New%20description
    &SubnetGroupName=mysubnetgroup
    &SubnetIds.member.1=subnet-42df9c3a
    &SubnetIds.member.2=subnet-48fc21a9
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20211801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20210801T220302Z
    &X-Amz-Expires=20210801T220302Z
```

```
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

サブネットグループの削除

サブネットグループが必要ではなくなったと判断した場合、サブネットグループを削除できます。サブネットグループがクラスターで現在使用されている場合、サブネットグループを削除できません。また、マルチ AZ が有効になっているクラスターのサブネットグループを削除したときに、そのクラスターのサブネット数が 2 つ未満になる場合は、サブネットグループを削除できません。最初に [マルチ AZ] のチェックを外し無効にしてから、サブネットを削除する必要があります。

次の手順では、サブネットグループを削除する方法を示します。

サブネットグループの削除 (コンソール)

サブネットグループを削除するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. 左の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リストで、削除する項目 [アクション] をクリックしてから、[削除] を選択します。

   > **Note**
   >
   > デフォルトのサブネットグループやクラスターに関連付けられているサブネットグループは削除できません。

4. [サブネットグループの削除] の確認画面が表示されます。
5. サブネットグループを削除するには、確認テキストボックスに `delete` を入力します。サブネットグループを保持するには、キャンセル を選択します。

サブネットグループの削除 AWS

AWS CLI を使用して、以下のパラメーターでコマンド delete-subnet-group を呼び出します。

- `--subnet-group-name` *`mysubnetgroup`*

Linux、macOS、Unix の場合:

```
aws memorydb delete-subnet-group \
```

```
    --subnet-group-name mysubnetgroup
```

Windows の場合:

```
aws memorydb delete-subnet-group ^
    --subnet-group-name mysubnetgroup
```

詳細については、AWS CLI のトピック「delete-subnet-group」を参照してください。

サブネットグループの削除 (MemoryDB API)

以下のパラメータを指定して、MemoryDB API を使用して DeleteSubnetGroup を呼び出します。

- SubnetGroupName=*mysubnetgroup*

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DeleteSubnetGroup
    &SubnetGroupName=mysubnetgroup
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20210801T220302Z
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

このコマンドでは何も出力されません。

詳細については、MemoryDB API トピックの「DeleteSubnetGroup」を参照してください。

MemoryDB API とインターフェイス VPC エンドポイント (AWS PrivateLink)

インターフェイス VPC エンドポイントを作成して、VPC と Amazon MemoryDB API エンドポイント間にプライベート接続を確立できます。インターフェイスエンドポイントは AWS PrivateLink を使用します。AWS PrivateLink を使用すると、インターネットゲートウェイ、NAT デバイス、VPN

接続、または AWS Direct Connect 接続なしで、MemoryDB API オペレーションにプライベートにアクセスできます。

VPC のインスタンスは、パブリック IP アドレスがなくても MemoryDB API エンドポイントと通信できます。また、MemoryDB API オペレーションの使用にも、パブリック IP アドレスを必要としません。VPC と MemoryDB 間のトラフィックは、Amazon ネットワークを離れません。各インターフェイスエンドポイントは、サブネット内の 1 つ以上の Elastic Network Interface によって表されます。Elastic Network Interface の詳細については、Amazon EC2 ユーザーガイドの「Elastic Network Interface」を参照してください。

- VPC エンドポイントの詳細については、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。
- 「MemoryDB API オペレーション」の詳細については、「MemoryDB API オペレーション」を参照してください。

インターフェイス VPC エンドポイントを作成した後、エンドポイントのプライベート DNS ホスト名を有効にすると、デフォルトの MemoryDB エンドポイント (https://memorydb.*Region*.amazonaws.com が VPC エンドポイントに解決されます。 プライベート DNS ホスト名を有効にしない場合は、Amazon VPC が以下の形式で使用できる DNS エンドポイント名を提供します。

```
VPC_Endpoint_ID.memorydb.Region.vpce.amazonaws.com
```

詳細については、Amazon VPC ユーザーガイドの「AWSインターフェイス VPC エンドポイント (PrivateLink)」をご参照ください。MemoryDB は、VPC 内のすべての API アクションの呼び出しをサポートしています。

> **Note**
>
> プライベート DNS ホスト名は、VPC 内の 1 つの VPC エンドポイントに対してのみ有効にできます。追加の VPC エンドポイントを作成する場合は、プライベート DNS ホスト名を無効にする必要があります。

VPC エンドポイントに関する考慮事項

MemoryDB API エンドポイントのインターフェイス VPC エンドポイントを設定する前に、Amazon VPC ユーザーガイド の「インターフェイスエンドポイントのプロパティと制限」を確認してくだ

さい。MemoryDB リソースの管理に関連するすべての MemoryDB API オペレーションは、AWS PrivateLink を使用して VPC から利用できます。VPC エンドポイントポリシーは MemoryDB API エンドポイントでサポートされます。デフォルトでは、エンドポイント経由で MemoryDB API オペレーションへのフルアクセスが許可されます。詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイントによるサービスのアクセスコントロール」を参照してください。

MemoryDB API 用のインターフェイス VPC エンドポイントの作成

MemoryDB API 用の VPC エンドポイントは、Amazon VPC コンソールまたは AWS CLI で作成できます。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エンドポイントの作成を参照してください。

インターフェイス VPC エンドポイントを作成した後、エンドポイントのプライベート DNS ホスト名を有効にできます。実行すると、デフォルトの MemoryDB エンドポイント (https://memorydb.*Region*.amazonaws.com) が VPC エンドポイントに解決されます。詳細については、「Amazon VPC ユーザーガイド」の「インターフェイスエンドポイントを介したサービスへのアクセス」を参照してください。

Amazon MemoryDB API 用の VPC エンドポイントポリシーの作成

VPC エンドポイントに MemoryDB API へのアクセスをコントロールするエンドポイントポリシーをアタッチできます。本ポリシーでは、以下を規定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイントでサービスへのアクセスを制御する」を参照してください。

Example MemoryDB API アクションの VPC エンドポイントポリシー

MemoryDB API のエンドポイントポリシーの例を次に示します。このポリシーは、エンドポイントにアタッチされると、すべてのリソースのすべてのプリンシパルに対して、登録されている MemoryDB API アクションへのアクセスを許可します。

```
{
 "Statement": [{
```

```
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "memorydb:CreateCluster",
   "memorydb:UpdateCluster",
   "memorydb:CreateSnapshot"
  ],
  "Resource": "*"
 }]
}
```

Example 指定した AWS アカウントからのすべてのアクセスを拒否する VPC エンドポイントポリシー

次の VPC エンドポイントポリシーは、AWS アカウント *123456789012* がエンドポイントを使用するリソースへのすべてのアクセスを拒否します。このポリシーは、他のアカウントからのすべてのアクションを許可します。

```
{
 "Statement": [{
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*",
   "Principal": "*"
  },
  {
   "Action": "*",
   "Effect": "Deny",
   "Resource": "*",
   "Principal": {
    "AWS": [
     "123456789012"
    ]
   }
  }
 ]
}
```

共通脆弱性識別子 (CVE): MemoryDB で対処されたセキュリティ脆弱性

一般的な脆弱性と露出 (CVE) は、一般的に知られているサイバーセキュリティの脆弱性に関するエントリの一覧です。各エントリは、識別番号、説明、および少なくとも 1 つの公開リファレンスを

含むリンクです。このページでは、MemoryDB で対処されたセキュリティ脆弱性のリストを確認できます。

MemoryDB を既知の脆弱性から保護するために、常に最新バージョンにアップグレードすることをお勧めします。MemoryDB は PATCH コンポーネントを公開します。パッチバージョンは、下位互換性のあるバグ修正、セキュリティ修正、非機能的な変更を目的とします。

以下の表を参照して、MemoryDB の特定のバージョンに特定のセキュリティ脆弱性に対する修正があるかどうかを確認できます。お使いの MemoryDB キャッシュがサービスの更新を保留している場合は、以下に示すセキュリティ脆弱性のいずれかに対して脆弱である可能性があります。サービスの更新を適用することをお勧めします。サポートされている MemoryDB エンジンのバージョンとアップグレード方法の詳細については、「エンジンバージョン」を参照してください。

> **Note**
>
> - CVE が 1 つの MemoryDB バージョンで対処されている場合、それ以降のバージョンでも対処されていることを意味します。
> - 以下の表のアスタリスク (*) は、セキュリティ脆弱性に対処するために、記載されているバージョンを実行している MemoryDB クラスターに最新のサービス更新を適用する必要があることを示しています。クラスターが実行されている MemoryDB バージョンに最新のサービス更新が適用されていることを確認する方法の詳細については、「サービスの更新の管理」を参照してください。

MemoryDB バージョン	対処された CVE
Valkey 7.3 および Valkey のそれ以前のすべてのバージョン Redis OSS 7.1 および Redis OSS のそれ以前のすべてのバージョン	CVE-2025-49844*、CVE-2025-46817*、CVE-2025-46818*、CVE-2025-46819*
Valkey 7.2 および 7.3	CVE-2025-21607*、CVE-2025-21605*、CVE-2024-31449*、CVE-2024-31227*、CVE-2024-31228*
Valkey 7.2.7	CVE-2024-51741

MemoryDB バージョン	対処された CVE
Redis OSS 7.1 および 6.2	CVE-2025-21605*、CVE-2024-31449*、CVE-2024-31227*、CVE-2024-31228*、CVE-2023-41056
Redis OSS 7.0.7	CVE-2023-41056*
Redis OSS 6.2.7	CVE-2024-46981
Redis OSS 6.2.6	CVE-2022-24834*、CVE-2022-35977*、CVE-2022-36021*、CVE-2023-22458、CVE-2023-25155、CVE-2023-28856 CVE-2023-45145: この CVE は、Redis OSS 6.2 および 7.0 では対処されていますが、Redis OSS 7.1 では対処されていないことに注意してください。
Redis OSS 6.0.5	CVE-2022-24735*、CVE-2022-24736*

MemoryDB のサービスの更新

MemoryDB は、サービスの更新が利用可能になったときに適用されるように、クラスターとノードのフリートを自動的にモニタリングします。通常、MemoryDB がこれらの更新を適用できるように、事前定義されたメンテナンスウィンドウを設定します。ただし、場合によっては、このアプローチが厳格すぎて、ビジネスフローが制限される可能性もあります。

MemoryDB のサービスの更新では、更新を適用するタイミングと内容を制御できます。選択した MemoryDB クラスターに対するこれらの更新の進行状況をリアルタイムでモニタリングすることもできます。

サービスの更新の管理

MemoryDB のサービスの更新は定期的にリリースされています。これらのサービスの更新の対象となるクラスターが 1 つ以上ある場合は、更新がリリースされたときに、E メール、SNS、Personal Health Dashboard (PHD)、および Amazon CloudWatch Events より通知が送信されます。更新

は、MemoryDB コンソールの [サービスの更新] ページにも表示されます。このダッシュボードを使用すると、MemoryDB フリートに関するサービスの更新とそのステータスをすべて表示できます。

自動更新を開始する前に、更新を適用するタイミングを制御します。最新のセキュリティパッチで MemoryDB を常に最新の状態に維持できるように、できるだけ早く [セキュリティ更新] タイプの更新を適用することを強くお勧めします。

以下のセクションでは、これらのオプションについて詳しく説明します。

トピック

- Amazon MemoryDB のマネージドメンテナンスとサービスの更新の概要

Amazon MemoryDB のマネージドメンテナンスとサービスの更新の概要

当社は、インスタンスにシームレスに適用されるパッチとアップグレードを使用して、頻繁に MemoryDB フリートのアップグレードを行います。これには以下の 2 つの方法があります。

1. 継続的マネージドメンテナンス。
2. サービスの更新。

このメンテナンスとサービスの更新は、セキュリティ、信頼性、運用パフォーマンスを強化するアップグレードを適用するために必要です。

継続的マネージドメンテナンスは、ユーザーのアクションを必要とせずに、メンテナンスウィンドウ内で随時、直接行われます。メンテナンスウィンドウはすべてのお客様に必須であり、オプトアウトはできないことに注意してください。この設定されたメンテナンスウィンドウ中にクリティカルな、または重要なアクティビティを行わないようにすることを強く推奨します。また、システムのセキュリティと最適なパフォーマンスを確保するために、重要な更新はスキップできないことにも注意してください。

サービスの更新では、更新を自分の判断で柔軟に適用できます。サービスの更新には期限があります。期限が切れると、更新がメンテナンスウィンドウに移動され、当社によって適用される場合があります。

更新は、できるだけ早く適用することによって適切に管理できます。また、ノードの交換によって管理することもできます。交換時に自動的に更新が適用されるからです。次回のメンテナンスウィンドウの前に更新がすべてのノードに適用されている場合、次回のメンテナンスウィンドウ中に更新アクティビティが行われることはありません。

サービスの更新

MemoryDB のサービスの更新では、特定のサービス更新を自分の判断で適用できます。これらの更新には、セキュリティパッチとマイナーなソフトウェア更新があります。これらの更新は、クラスターのセキュリティ、信頼性、運用パフォーマンスの強化に役立ちます。

これらのサービスの更新の価値は、更新を適用するタイミングを制御できることです (例えば、MemoryDB クラスターの 24 時間 365 日の可用性を必要とする重要なビジネスイベントがある場合には、サービス更新の適用を遅らせることができます)。

これらのサービスの更新の対象となるクラスターが 1 つ以上ある場合は、更新がリリースされたときに、E メール、Amazon SNS、AWS Health Dashboard、および Amazon CloudWatch Events より通知が送信されます。更新は、MemoryDB コンソールの [サービスの更新] ページにも表示されます。このダッシュボードを使用すると、MemoryDB フリートに関するサービスの更新とそのステータスをすべて表示できます。

自動更新を開始する前に、更新を適用するタイミングを制御します。最新のセキュリティパッチで MemoryDB を常に最新の状態に維持できるように、できるだけ早く [セキュリティ更新] タイプの更新を適用することを強くお勧めします。

クラスターは、複数の異なるサービス更新の対象である場合があります。ほとんどの更新では、更新を個別に適用する必要はありません。クラスターに 1 つの更新を適用すると、該当する他の更新も「完了」とマークされます。ステータスが自動的に「完了」に変わらない場合は、同じクラスターに複数の更新を個別に適用する必要が生じることがあります。

サービスの更新による影響とダウンタイム

ユーザーまたは Amazon MemoryDB が 1 つ以上の MemoryDB クラスターにサービスの更新を適用した場合、更新は、選択したすべてのクラスターが更新されるまで、各シャード内で一度に 1 つのノードにのみ適用されます。更新中のノードでは数秒のダウンタイムが発生しますが、残りのクラスターはトラフィックを処理し続けます。

- クラスター設定は変更されません。
- CloudWatch メトリクスに遅延が生じますが、メトリクスは可能な限り早く追いつきます。

ノード交換はアプリケーションにどのように影響するか? – MemoryDB ノードの場合、交換プロセスは耐久性と可用性を保証するように設計されています。単一ノードの MemoryDB クラスターの場合、MemoryDB はレプリカを動的にスピンアップし、耐久性コンポーネントからデータを復元してから、そのレプリカにフェイルオーバーします。複数のノードで構成されるレプリケーショングループの場合、MemoryDB は既存のレプリカを交換し、耐久性コンポーネントから新しいレプリカに

データを同期します。MemoryDB は複数のノードがある場合にのみマルチ AZ であるため、このシナリオでは、プライマリを交換すると、リードレプリカへのフェイルオーバーがトリガーされます。計画されたノード交換は、クラスターが受信した書き込みリクエストを処理する間に完了します。ノードが 1 つしかない場合、MemoryDB はプライマリを交換し、耐久性コンポーネントからデータを同期します。この間、プライマリノードは使用できなくなるため、書き込みの中断が長くなります。

交換をスムーズに行い、データ損失を最小限に抑えるには、どのようなベストプラクティスに従う必要があるか? – MemoryDB では、データの耐久性が高いため、単一ノードの実装でさえもデータ損失は予期されません。ただし、万一障害が発生した場合の損失の可能性を最小限に抑えるため、マルチ AZ およびバックアップ戦略を実施することを推奨します。当社は、交換をスムーズに行えるようクラスターを安定した状態に維持するために、同じクラスターの必要最小限のノードの交換を試みます。マルチ AZ を有効にすると、プライマリとリードレプリカを異なるアベイラビリティーゾーンにプロビジョニングできます。この場合、ノードが交換されると、プライマリロールはシャード内のレプリカにフェイルオーバーします。このシャードはトラフィックを処理し、データはその耐久性コンポーネントから復元されます。シャードごとにプライマリとレプリカがそれぞれ 1 つしか含まれていない構成の場合は、パッチ適用の前にレプリカを追加することを推奨します。これにより、パッチ適用プロセス中の可用性の低下が防止されます。交換は受信書き込みトラフィックが少ない期間中にスケジュールすることを推奨します。

メンテナンス中のアプリケーションの中断を最小限に抑えるには、どのようなクライアント設定のベストプラクティスに従う必要があるか? – MemoryDB では、クラスターモード設定は常に有効になっています。これにより、マネージド型またはアンマネージド型のオペレーション中に最高の可用性が提供されます。レプリカノードの個々のノードエンドポイントは、あらゆる読み取り操作に使用できます。MemoryDB では、クラスターで常に自動フェイルオーバーが有効になっています。つまり、プライマリノードが変更される可能性があります。したがって、アプリケーションでノードのロールを確認し、すべての読み取りエンドポイントを更新することで、プライマリに大きな負荷がかからないようにする必要があります。同様に、メンテナンスウィンドウ中にレプリカが読み取りリクエストで過負荷状態になるのを回避します。これを実現する方法の 1 つは、メンテナンス中の読み取りの中断を避けるために、少なくとも 2 つのリードレプリカを用意することです。

クライアントアプリケーションをテストし、アプリケーションが Redis/Valkey クラスタープロトコルに準拠していること、およびリクエストをノード間で適切にリダイレクトできることを確認することが重要です。メンテナンスおよび交換作業中に MemoryDB ノードが過負荷状態になるのを回避するために、バックオフおよび再試行戦略を実施することを推奨します。

再スケジュール – メンテナンスウィンドウを変更することで、サービスの更新を延期できます。スケジュールされた更新は、スケジュールされた日付がクラスターのメンテナンスウィンドウに一致

する場合にのみクラスターに適用されます。メンテナンスウィンドウを変更し、スケジュールされた日付を過ぎると、サービスの更新は、今後の週に新しく指定したウィンドウに再スケジュールされます。新しい日付の 1 週間前に新しい通知が届きます。

AWS のセキュリティは責任共有です。更新をできるだけ早く適用することを強く推奨します。

サービスの更新のオプトアウト – サービスの更新をオプトアウトできるかどうかを判断するには、[自動更新開始日] 属性の値を確認します。サービスの更新の [自動更新開始日] 属性の値が設定されている場合、MemoryDB は残りのクラスターへのサービスの更新を今後のメンテナンスウィンドウにスケジュールします。これをオプトアウトすることはできません。ただし、メンテナンスウィンドウの前に残りのクラスターにサービスの更新を適用した場合、MemoryDB はメンテナンスウィンドウ中にサービスの更新を再適用しません。詳細については、「サービスの更新の適用」を参照してください。

メンテナンスウィンドウ中に MemoryDB がサービスの更新を直接適用できないのはなぜか? – サービスの更新の目的は、更新を適用するタイミングをユーザーが柔軟に決定できるようにすることです。MemoryDB がサポートするコンプライアンスプログラムに参加していないクラスターは、これらの更新を適用しないか、年間を通じて低い頻度で適用するかを選択できます。ただし、規制への準拠を維持する更新を適用することを推奨します。これは、サービスの更新の [自動更新開始日] 属性の値が存在しない場合にのみ当てはまります。詳細については、「MemoryDB のコンプライアンス検証」を参照してください。

メンテナンスウィンドウで適用される更新は、サービスの更新とどのように異なるか? – 継続的マネージドメンテナンスを介して適用された更新は、メンテナンスウィンドウに直接スケジュールされます。ユーザー側のアクションは必要ありません。サービスの更新には期限があり、ユーザーは [自動更新開始日] を期限として適用するタイミングを制御できます。期限までに適用されない場合、MemoryDB はメンテナンスウィンドウでこれらの更新をスケジュールすることがあります。

継続的マネージドメンテナンスの更新

これらの更新は必須であり、メンテナンスウィンドウに直接適用されます。ユーザー側のアクションは必要ありません。これらの更新は、サービスの更新で提供される更新とは異なります。

継続的メンテナンスによる影響とダウンタイム

ノードの交換にはどのくらいの時間がかかるか? – 交換は通常は 30 分以内に完了します。特定のインスタンス設定やトラフィックパターンでは、交換にそれ以上の時間がかかる場合があります。

ノード交換はアプリケーションにどのように影響するか? – 継続的マネージドメンテナンスの更新は、ノード交換を通じて「サービスの更新」と同じ方法で適用されます。詳細については、上記の「サービスの更新による影響とダウンタイム」セクションを参照してください。

ノード交換を自分で管理するにはどうすればよいか? – このような交換を、スケジュールされたノード交換ウィンドウより前に、任意のタイミングで独自に管理することもできます。交換を自分で管理することを選択した場合は、ユースケースに応じてさまざまなアクションを実行できます。

- クラスター内のノードを 1 つ以上のシャードに交換する: バックアップと復元またはスケールアウト後のスケールインを使用してノードを交換できます。
- メンテナンスウィンドウを変更する: クラスターのメンテナンスウィンドウを変更することもできます。後でメンテナンスウィンドウをより便利な時間に変更するには、UpdateCluster API、update-cluster CLI を使用するか、MemoryDB マネジメントコンソールで [変更] をクリックします。メンテナンスウィンドウを変更すると、MemoryDB は新しく指定されたウィンドウ中にノードのメンテナンスをスケジュールします。

  これが実際にどのように機能するか見てみましょう。今が 11/09、木曜日の 1500 とします。次のメンテナンスウィンドウは 11/10、金曜日の 1700 です。次の 3 つのシナリオがあります。
  - メンテナンスウィンドウを金曜日の 1600 に変更します (現在の日時より後で、次の予定メンテナンスウィンドウより前)。ノードは 11/10、金曜日の 1600 に置き換えられます。
  - メンテナンスウィンドウを土曜日の 1600 に変更します (現在の日時より後で、次の予定メンテナンスウィンドウより後)。ノードは 11/11、土曜日の 1600 に置き換えられます。
  - メンテナンスウィンドウを水曜日の 1600 に変更します (週の中で、現在の日時より前)。ノードは 11/15、次の水曜日の 1600 に置き換えられます。

  詳細については、「メンテナンスの管理」を参照してください。

  さまざまなリージョンのさまざまなクラスターのノードを同時に交換できます。ただし、対象となるクラスターのメンテナンスウィンドウが同じになるように設定されていることが条件です。

今後予定されている交換について確認するにはどうすればよいか? – AWS Health Dashboard でヘルス通知を受け取る必要があります。DescribeServiceUpdates API を使用して、さまざまなサービスアップグレードのステータスを確認することもできます。当社では、予測可能な交換について事前にお客様に通知するよう努めています。ただし、予測不可能な障害などの例外的なケースでは、予告なしに交換を行う可能性があります。

スケジュールされたメンテナンスをより適切なタイミングに変更できるか? – はい。メンテナンスウィンドウを変更することで、スケジュールされたメンテナンスをより適切な時間に延期できます。

これらのノード交換を行うのはなぜか? – これらの交換は、基盤となるホストに必須のソフトウェア更新を適用するために必要です。これらの更新は、セキュリティ、信頼性、運用パフォーマンスの強化に役立ちます。

これらの交換は複数アベイラビリティーゾーンのノードと異なるリージョンのクラスターに同時に影響するか? – 交換は、クラスターのメンテナンスウィンドウに応じて、複数のアベイラビリティーゾーンまたはリージョンで並行して実行できます。

サービスの更新の適用

フリートに対するサービスの更新の適用は、更新が 使用可能 ステータスになってから開始することができます。サービスの更新は累積的です。つまり、未適用の更新も最新の更新に含まれます。

サービスの更新で自動更新が有効になっている場合、使用可能になったときにアクションを実行しないよう選択できます。MemoryDB は、[自動更新開始日] 以降、クラスターのメンテナンス期間中に更新を適用するようにスケジュールします。更新のステージごとに、関連する通知を受け取ります。

> **Note**
>
> ステータスが 使用可能または スケジュール済み であるサービスの更新だけを適用できます。

該当する MemoryDB クラスターへのサービス固有の更新の確認および適用の詳細については、「コンソールを使用したサービスの更新の適用」を参照してください。

1 つ以上の MemoryDB クラスター向けに新しいサービスの更新が利用可能になったら、MemoryDB コンソール、API、または AWS CLI を使用して更新を適用できます。次のセクションでは、更新の適用に使用できるオプションについて説明します。

コンソールを使用したサービスの更新の適用

使用可能なサービスの更新のリストと他の情報を確認するには、コンソールの サービスの更新 (サービスの更新) ページに移動します。

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/memorydb/ で MemoryDB のコンソールを開きます。
2. ナビゲーションペインで、サービスの更新を選択します。

[サービスの更新の詳細] では、次の項目を表示できます。

- サービスの更新名: サービスの更新の一意の名前
- サービスの更新名: サービスの更新に関する詳細情報

- 自動更新開始日:この属性が設定されている場合、MemoryDB は、この日付以降に適切なメンテナンスウィンドウでクラスターを自動更新するようにスケジューリングを始めます。正確なスケジュールされたメンテナンスウィンドウに事前に通知が届きますが、[自動更新開始日] の直後のメンテナンスウィンドウではない場合があります。ただし、クラスターにはいつでもアップデートを適用できます。属性が設定されていない場合、サービスの更新は自動更新が有効になっておらず、MemoryDB はクラスターを自動的に更新しません。

クラスターの更新ステータスセクションでは、サービスの更新が適用されていない、または最近適用されたばかりのクラスターのリストを表示できます。クラスターごとに、以下を表示できます。

- クラスター名: クラスターの名前
- ノードを更新しました: 特定のクラスター内で更新された、または特定のサービスの更新に対して利用可能な状態の個々のノードの比率。
- 更新タイプ: サービスの更新のタイプ (セキュリティ更新 または エンジン更新のいずれか)
- ステータス: クラスター上のサービス更新のステータス。以下のいずれかです。
  - 使用可能: 必要なクラスターでこの更新が利用可能です。
  - 進行中: このクラスターに更新を適用しています。
  - スケジュール済み: 更新日がスケジュールされています。
  - 完了: 更新が正常に適用されました。完了ステータスのクラスターは、完了後 7 日間表示されます。

  ステータスが 使用可能または スケジュール済み (スケジュール済み) であるクラスターのいずれかまたはすべてを選択してから、今すぐ適用を選択した場合、更新がそれらのクラスターに適用され始めます。

AWS CLI を使用してサービスの更新を適用する

サービスの更新が利用可能であるという通知を受け取ったら、AWS CLI を使用してそれらの更新を確認し、適用することができます。

- 利用可能なサービスの更新の説明を取得するには、次のコマンドを実行します。

  ```
  aws memorydb describe-service-updates --status available
  ```

  詳細については、「describe-service-updates」を参照してください。
- クラスターのリストにサービスの更新を適用するには、次のコマンドを実行します。

```
aws memorydb batch-update-cluster --service-update
ServiceUpdateNameToApply=sample-service-update --cluster-names cluster-1
cluster2
```

詳細については、「batch-update-cluster」を参照してください。

参照資料

このセクションのトピックでは、MemoryDB API および MemoryDB の AWS CLI セクションの使用について説明します。また、このセクションには一般的なエラーメッセージとサービス通知も含まれます。

- MemoryDB API の使用
- 「MemoryDB API リファレンス」
- 「AWS CLI リファレンスの「MemoryDB」セクション」

MemoryDB API の使用

このセクションでは、MemoryDB のオペレーションを使用および実装する方法を、メソッドに重点を置いて説明します。これらのオペレーションの詳細については、「MemoryDB API リファレンス」を参照してください。

トピック

- クエリ API を使用する
- 利用可能なライブラリ
- アプリケーションのトラブルシューティング

クエリ API を使用する

クエリパラメータ

HTTP クエリベースのリクエストとは、HTTP 動詞 (GET または POST) とクエリパラメータ `Action` で記述する HTTP リクエストです。

各クエリリクエストに、アクションの認証と選択を処理するための一般的なパラメータがいくつか含まれている必要があります。

オペレーションの中にはパラメータのリストを取るものがあります。これらのリストは、`param.`*n* 表記を使用して指定されます。*n* 値は、1 から始まる整数です。

クエリリクエストの認証

HTTPS 経由でのみリクエストを送信できます。また、各クエリリクエストには署名を含める必要があります。このセクションでは、署名を作成する方法について説明します。次に説明する方法は、署名バージョン 4 と呼ばれます。

AWS へのリクエストを認証するために使用される基本的な手順を次に示します。この手順では、AWS に登録されており、アクセスキー ID とシークレットアクセスキーを持っていることを前提としています。

クエリ認証プロセス

1. 送信者は、AWS へのリクエストを構築します。

2. このトピックの次のセクションに示すように、送信者は、SHA-1 ハッシュ関数を使用してリクエストの署名 (Hash-based Message Authentication Code (HMAC) のキー付きハッシュ) を生成します。
3. リクエストの送信者は、リクエストデータ、署名、およびアクセスキー ID (使用するシークレットアクセスキーのキー識別子) を AWS に送信します。
4. AWS ではアクセスキー ID を使用して、シークレットアクセスキーを調べます。
5. AWS では、リクエストの署名を生成する際に使用したものと同じアルゴリズムを使い、リクエストデータとシークレットアクセスキーから署名を生成します。
6. 署名が一致すると、リクエストは認証されたものと見なされます。もし署名が一致しなかった場合、リクエストの処理は拒否され、AWS はエラーレスポンスを返します。

> **Note**
>
> リクエストに `Timestamp` パラメータが含まれている場合、リクエストに対して生成された署名はパラメータの値の 15 分後に期限が切れます。
> リクエストに `Expires` パラメータが含まれている場合、署名は `Expires` パラメータで指定された時刻に期限が切れます。

リクエストの署名を計算するには

1. 本手順で後に必要となる、正規化されたクエリ文字列を作成します。
    a. 自然なバイト順のパラメータ名で、UTF-8 のクエリ文字列コンポーネントを並び替えます。パラメータは、GET URI または POST ボディから取得される場合があります。(Content-Type が application/x-www-form-urlencoded の場合)
    b. URL は、以下の規則に応じてパラメータ名と値をエンコードします。
        i. RFC 3986 が定義する非予約文字を、URL がエンコードすることはありません。非予約文字とは、A〜Z、a〜z、0〜9、ハイフン (-)、アンダーバー (_)、ピリオド (.)、およびチルド (~) です。
        ii. 他のすべての文字についても、%XY (X および Y には HEX 文字の 0-9 および大文字の A-F が入る) によるパーセントエンコードが必要です。
        iii. パーセントは、拡張 UTF-8 文字を %XY%ZA.... 形式でエンコードします。
        iv. パーセントは、スペース文字を %20 (通常エンコードスキーマが行なうような + ではありません) としてエンコードします。

c. パラメータの値が空値の場合でも、エンコードされるパラメータ名とエンコードされる値の間に等号 (=) (ASCII コード 61) を入れます。

d. それぞれのパラメータ名と値のペアをアンド (&) (ASCII コード 38) で分割します。

2. 文字列を作成し、以下の擬似文法に従って ("\n" は ASCII 新規行を意味します) 署名を作成します。

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

HTTPRequestURI 要素は URI の HTTP 絶対パス要素ですが、クエリ文字列は含みません。HTTPRequestURI が空値の場合は、スラッシュ (/) を使用してください。

3. 作成したばかりの文字列を使い、シークレットアクセスキーをキーとして、また SHA256 または SHA1 をハッシュアルゴリズムとして、RFC 2104 に準拠した HMAC を計算します。

詳細については、https://www.ietf.org/rfc/rfc2104.txt を参照してください。

4. 結果の値を base64 に変換します。

5. その値は、`Signature` パラメータの値としてリクエストに含めます。

サンプルのリクエストを次に示します (見やすくするために改行が追加されています)。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeClusters
    &ClusterName=myCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2021-01-01
```

前のクエリ文字列では、次の文字列に対する HMAC 署名が生成されます。

```
GET\n
    memory-db.amazonaws.com\n
    Action=DescribeClusters
    &ClusterName=myCluster
    &SignatureMethod=HmacSHA256
```

```
    &SignatureVersion=4
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE%2F20140523%2Fus-east-1%2Fmemorydb%2Faws4_request
    &X-Amz-Date=20210801T223649Z
    &X-Amz-SignedHeaders=content-type%3Bhost%3Buser-agent%3Bx-amz-content-sha256%3Bx-
amz-date
        content-type:
        host:memory-db.us-east-1.amazonaws.com
        user-agent:ServicesAPICommand_Client
    x-amz-content-sha256:
    x-amz-date:
```

結果の署名付きリクエストは次のようになります。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeClusters
    &ClusterName=myCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20141201/us-east-1/memorydb/aws4_request
    &X-Amz-Date=20210801T223649Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=2877960fced9040b41b4feaca835fd5cfeb9264f768e6a0236c9143f915ffa56
```

プロセスへの署名とリクエスト署名の計算の詳細については、トピック「Signature Version 4 signing process」とそのサブトピックを参照してください。

利用可能なライブラリ

AWS では、クエリ API の代わりに言語固有の API を使用してアプリケーションを構築するソフトウェアデベロッパー向け Software Development Kit (SDK) を提供します。こうした SDK には、リクエスト認証、リクエストの再実行、エラー処理など、 (API には含まれない) 基本的な機能が用意されていて、簡単に開始できるようになっています。次のプログラミング言語の SDK と追加のリソースがあります。

- Java
- Windows および .NET

- PHP
- Python *
- Ruby

他の言語については、「サンプルコードとライブラリ」を参照してください。

アプリケーションのトラブルシューティング

MemoryDB では、MemoryDB API とのやり取りで発生する問題をトラブルシューティングする際に役立つ、具体的でわかりやすいエラーを提供します。

エラーの取得

通常、アプリケーションでは、結果を処理する前にリクエストでエラーが生成されたかどうかを必ず確認します。エラーが発生したかどうかを確認する最も簡単な方法は、MemoryDB API からのレスポンスで `Error` ノードを検索することです。

XPath 構文を使用すると、簡単な方法で `Error` ノードがあるかどうかを検索し、エラーコードとメッセージを取得することができます。次のコードでは、Perl および XML::XPath モジュールによって、リクエスト時のエラーの発生を判定しています。エラーが発生した場合、レスポンス内の最初のエラーコードとメッセージが表示されます。

```
use XML::XPath;
my $xp = XML::XPath->new(xml =>$response);
if ( $xp->find("//Error") )
{print "There was an error processing your request:\n", " Error code: ",
$xp->findvalue("//Error[1]/Code"), "\n", " ",
$xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

トラブルシューティングのヒント

MemoryDB API の問題を診断して解決するには、次の手順を実行することをお勧めします。

- MemoryDB が正しく実行されていることを確認します。

  これを行うには、ブラウザウィンドウを開き、MemoryDBサービス (https://memory-db.us-east-1.amazonaws.com など) にクエリリクエストを送信します。MissingAuthenticationTokenException または UnknownOperationException は、サービスが利用可能であり、リクエストに応答していることを示します。

- リクエストの構文を確認します。

  「API リファレンス」には、各 MemoryDB オペレーションについてのリファレンスページがあります。パラメータを正しく使用していることをもう一度確認してください。間違っている可能性がある部分を判断するヒントとして、同様のオペレーションを実行しているサンプルのリクエストやユーザーシナリオを調べてください。

- フォーラムを確認します。

  MemoryDB にはディスカッションフォーラムがあります。このフォーラムでは、これまで他のユーザーが経験してきた問題に対する解決策を探すことができます。フォーラムを見るには、以下をご覧ください。

  https://forums.aws.amazon.com/

MemoryDB のクォータ

AWS アカウントには、AWS のサービスごとにデフォルトのクォータ (以前は制限と呼ばれていました) があります。特に明記されていない限り、クォータは地域固有です。一部のクォータについては引き上げをリクエストできますが、その他のクォータについては引き上げることはできません。

クォータの引き上げをリクエストするには、「Service Quotas ユーザーガイド」の「クォータ引き上げリクエスト」を参照してください。Service Quotasでクォータがまだ利用できない場合は、制限の引き上げ のフォームを使用してください。

お客様の AWS アカウントには、MemoryDB に関連する以下のクォータがあります。

名前	デフォルト値	説明	メトリクス名
リージョンあたりのノード	300	リージョン内の MemoryDB クラスター全体にわたるノードの最大数。このクォータは、特定のリージョン内のリザーブドノードとノンリザーブドノードの両方に適用されます。同じリージョンで、リザーブドノードを 300 まで、ノンリザーブドノードを 300 まで保持できます。	NodesPerRegion
クラスターあたりのノード数 (Redis OSS クラスターモードが有効)	90	MemoryDB 用の個々の Redis OSS クラスターのノードの最大数。	NodesPerCluster

名前	デフォルト値	説明	メトリクス名
リージョンあたりのパラメータグループ	300	リージョンで作成できる、パラメータグループの最大数。	ParameterGroup
リージョンあたりのサブネットグループ	300	リージョンで作成できる、サブネットグループの最大数。	SubnetGroup
サブネットグループあたりのサブネット	20	サブネットグループに対して定義するサブネットの最大数。	SubnetsPerSubnetGroup
リージョンあたりのユーザー数	2000	リージョンで作成できる、ユーザーの最大数。	ユーザー
リージョンあたりのユーザーグループ数	200	リージョンで作成できる、ユーザーグループの最大数。	UserGroup
ユーザーグループあたりのユーザー	100	ユーザーグループに対して定義できるユーザーの最大数。	UsersPerUserGroup

MemoryDB ユーザーガイドのドキュメント履歴

次の表では、MemoryDB のドキュメントリリースについて説明します。

変更	説明	日付
MemoryDB マルチリージョンがリリースされました。	MemoryDB マルチリージョンがリリースされました。	2024 年 12 月 1 日
MemoryDB マルチリージョンの IAM とセキュリティポリシーが更新されました。	IAM とセキュリティポリシーが更新されました。サービスにリンクされたロールの詳細については、「サービスにリンクされたロールの使用」および「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。	2024 年 12 月 1 日
MemoryDB が Valkey をサポートするようになりました。	MemoryDB が Valkey をサポートするようになりました。	2024 年 10 月 8 日
MemoryDB では、IAM を使用したユーザー認証がサポートされるようになりました	IAM 認証では、AWS Identity and Access Management アイデンティティを使用して MemoryDB への接続を認証できます。これにより、セキュリティモデルを強化し、多くの管理セキュリティタスクを簡素化できます。詳細については、「IAM を使った認証」を参照してください。	2023 年 5 月 10 日
MemoryDB が、Redis OSS 7 をサポートするようになりました	このリリースでは、Redis OSS 関数、ACL の改善、シャードされた Pub/Sub、拡張 I/O 多重化など、いくつか	2023 年 5 月 9 日

	の新機能が MemoryDB に追加されました。詳細については、「Redis OSS エンジンのバージョン」を参照してください。	
MemoryDB はリザーブドノードを提供するようになりました	オンデマンドノードの料金と比べて、リザーブドノードには大幅な割引が適用されます。リザーブドノードは物理ノードではなく、アカウント内のオンデマンドノードの使用に適用される割引です。詳細については、「MemoryDB reserved nodes」を参照してください。	2022 年 12 月 27 日
MemoryDB がデータ階層化をサポートするようになりました	MemoryDB データ階層化。データ階層化は、クラスターを数百テラバイトの容量までスケールするための低コストな方法として使用できます。詳細については、データ階層化を参照してください。	2022 年 11 月 3 日

MemoryDB では、ネイティブ JavaScript Object Notation (JSON) 形式がサポートされるようになりました	ネイティブ JavaScript Object Notation (JSON) 形式は、Redis OSS クラスター内の複雑なデータセットをエンコードするためのシンプルでスキーマレスな方法です。Redis OSS クラスター内で JavaScript Object Notation (JSON) 形式を使用してデータをネイティブに保存およびアクセスし、それらのクラスターに保存されている JSON データを更新できます。カスタムコードを管理してシリアル化および逆シリアル化する必要はありません。詳細については、「JSON の使用開始」を参照してください。	2022 年 5 月 25 日
MemoryDB が AWS PrivateLink をサポートするようになりました	AWS PrivateLink を使用すると、インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続なしで、MemoryDB API オペレーションにプライベートにアクセスできます。詳細については、「MemoryDB API とインターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。	2022 年 1 月 24 日
初回リリース	MemoryDB ユーザーガイドの初回リリース。詳細については、「MemoryDB とは」を参照してください。	2021 年 8 月 19 日