aws

デベロッパーガイド

Amazon DocumentDB



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

Amazon DocumentDB: デベロッパーガイド

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

Amazon 社の商標およびトレードドレスは、Amazon 社製以外の製品またはサービスに関連して使用してはならず、お客様に混乱を招く可能性のある方法や Amazon 社を誹謗または中傷するような方法で使用してはならないものとします。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

Amazon DocumentDB とは ... 1
概要 ... 1
クラスター ... 3
インスタンス ... 4
リージョンと AZ ... 9
リージョン ... 9
アベイラビリティーゾーン ... 9
料金 ... 11
無料トライアル ... 12
モニタリング ... 12
インターフェイス ... 13
AWS Management Console ... 13
AWS CLI ... 13
MongoDB ドライバー ... 13
次のステップ ... 13
仕組み ... 14
Amazon DocumentDB エンドポイント ... 16
TLS サポート ... 20
Amazon DocumentDB のストレージ ... 20
Amazon DocumentDB レプリケーション ... 21
Amazon DocumentDB の信頼性 ... 21
読み込み設定のオプション ... 22
TTL 削除 ... 27
有料リソース ... 28
ドキュメントデータベースとは ... 31
ユースケース ... 31
ドキュメントについて理解する ... 32
ドキュメントでの作業 ... 38
入門ガイド ... 51
前提条件 ... 51
ステップ 1: クラスターを作成する ... 53
ステップ 2: クラスターに接続する ... 56
ステップ 3: データを挿入してクエリする ... 57
ステップ 4: 探索する ... 59

を使用したクイックスタート AWS CloudFormation 60
前提条件 60
必要な IAM アクセス許可 61
Amazon EC2 キーペア 64
Amazon DocumentDB AWS CloudFormation スタックを起動する 64
Amazon DocumentDB クラスターにアクセスする 69
終了保護と削除保護 69
MongoDB との互換性 71
MongoDB 5.0 との互換性 71
Amazon DocumentDB 5.0 の新機能 71
Amazon DocumentDB 5.0 の使用開始 72
Amazon DocumentDB 5.0 へのアップグレードまたは移行 73
機能的な違い 73
MongoDB 4.0 の互換 74
Amazon DocumentDB 4.0 機能 75
Amazon DocumentDB 4.0 の使用開始 76
Amazon DocumentDB 4.0 へのアップグレードまたは移行 77
機能的な違い 77
トランザクション 79
要件 79
ベストプラクティス 80
制限 80
モニタリングと診断 81
トランザクション分離レベル 82
ユースケース 82
マルチステートメントトランザクション 82
マルチコレクショントランザクション 84
コールバック API のトランザクション API の例 86
コア API のトランザクション API の例 86
サポートされているコマンド 120
サポート対象外の機能 121
セッション 122
因果整合性 122
再試行可能な書き込み 123
トランザクションエラー 123
ベストプラクティス 125

基本運用ガイドライン ... 125
インスタンスのサイズ指定 ... 127
インデックスの使用 ... 128
インデックスの構築 ... 128
インデックスの選択性 ... 128
インデックスがデータの書き込みに与える影響 ... 129
欠落しているインデックスの識別 ... 129
使用されていないインデックスの識別 ... 130
セキュリティに関するベストプラクティス ... 130
コスト最適化 ... 131
メトリクスを使用したパフォーマンスの問題の特定 ... 131
パフォーマンスメトリクスの表示 ... 132
CloudWatch アラームの設定 ... 132
パフォーマンスメトリクスの評価 ... 132
CloudWatch メトリクスで Amazon DocumentDB インスタンスの使用状況を評価する ... 134
クエリのチューニング ... 135
TTL および時系列ワークロード ... 136
移行 ... 136
クラスターパラメータグループの使用 ... 137
集約パイプラインクエリ ... 137
batchInsert および batchUpdate ... 137
MongoDB との機能的な違い ... 138
Amazon DocumentDB の機能上の利点 ... 138
暗示的なトランザクション ... 138
機能的な違いの更新 ... 139
配列インデックス作成 ... 140
マルチキーインデックス ... 141
文字列内の null 文字 ... 142
ロールベースアクセスコントロール ... 142
$regex インデックス作成 ... 142
ネストされたドキュメントの投影 ... 143
MongoDB との機能的な違い ... 143
$vectorSearchoperator ... 144
OpCountersCommand ... 144
管理データベースとコレクション ... 144
cursormaxTimeMS ... 144

explain() ... 144
インデックスビルド ... 145
パスに空のキーでルックアップ ... 145
MongoDB API、オペレーション、およびデータ型 ... 145
mongodump および mongorestore ユーティリティ ... 146
結果の順序 ... 146
再試行可能な書き込み ... 146
スパースインデックス ... 147
$all 式内での $elemMatch の使用 ... 147
$ne、$nin、$nor、$not、$exists および $elemMatch のインデックス作成 ... 148
フィールド名のドル ($) とドット (.) ... 149
$lookup ... 149
$natural および逆ソート ... 153
サポートされている MongoDB API、オペレーション、およびデータ型 ... 154
データベースコマンド ... 154
管理コマンド ... 155
集計 ... 157
認証 ... 157
診断コマンド ... 157
クエリ操作と書き込み操作 ... 158
ロール管理コマンド ... 159
セッションコマンド ... 160
ユーザー管理 ... 161
シャーディングコマンド ... 162
クエリおよびプロジェクション演算子 ... 163
配列演算子 ... 164
ビット演算子 ... 164
コメント演算子 ... 164
比較演算子 ... 165
要素演算子 ... 165
評価クエリ演算子 ... 165
論理演算子 ... 166
プロジェクション演算子 ... 166
更新演算子 ... 167
配列演算子 ... 167
ビット演算子 ... 168

フィールド演算子 168
更新修飾子 168
地理空間 169
ジオメトリ指定子 169
クエリセレクター 170
カーソルメソッド 170
集約パイプライン演算子 172
アキュムレータ式 173
算術演算子 174
配列演算子 175
ブール演算子 176
比較演算子 177
条件式演算子 177
データ型演算子 177
データサイズ演算子 178
日付演算子 178
リテラル演算子 179
マージ演算子 179
自然演算子 180
セット演算子 180
ステージ演算子 181
文字列演算子 182
システム変数 184
テキスト検索演算子 184
型変換演算子 184
変数演算子 185
その他の演算子 185
データ型 186
インデックスとインデックスプロパティ 187
インデックス 187
インデックスプロパティ 188
生成 AI 189
SageMaker Canvas 189
SageMaker AI Canvas を使用してノーコード ML モデルを構築する方法 189
SageMaker AI ドメインとユーザープロファイルの設定 190
Amazon DocumentDB と SageMaker AI Canvas の IAM アクセス許可の設定 190

SageMaker AI Canvas のデータベースユーザーとロールの作成 190
利用できるリージョン 191
ベクトル検索 192
ベクトルの挿入 192
ベクトルインデックスの作成 193
インデックス定義の取得 197
ベクトルのクエリ 198
特徴量と制限事項 202
ベストプラクティス 203
DocumentDB への移行 205
「クイックスタートガイド」 205
DMS ソースを準備する 205
DMS のセットアップ 206
DocumentDB 圧縮を有効にする 206
レプリケーションタスクを作成する 206
進行状況をモニタリングする 207
追加情報 207
移行ランブック 207
互換性 208
ワークロードの検出 211
インデックスの移行 215
ユーザー移行 216
データ移行 218
モニタリング 230
検証 233
DocumentDB のアップグレード 238
Amazon DocumentDB エンジンバージョンのアップグレード 238
MVU 前提条件と制限事項 239
インプレースメジャーバージョンアップグレードのベストプラクティス 240
インプレースメジャーバージョンアップグレードの実行 245
Amazon DocumentDB 3.6/4.0 から 5.0 にアップグレードされたクラスターと新規作成される Amazon DocumentDB 5.0 クラスターの違い 248
インプレースメジャーバージョンアップグレードのトラブルシューティング 249
を使用したアップグレード AWS DMS 250
ステップ 1: 変更ストリームを有効にする 250
ステップ 2: 変更ストリームログの保持期間を変更する 251

ステップ 3: インデックスを移行する 251
ステップ 4: AWS DMS レプリケーションインスタンスを作成する 252
ステップ 5: AWS DMS ソースエンドポイントを作成する 255
ステップ 6: AWS DMS ターゲットエンドポイントを作成する 257
ステップ 7: 移行タスクの作成と実行 259
ステップ 8: アプリケーションエンドポイントから Amazon DocumentDB クラスター への変更 261
セキュリティ 262
を使用したパスワード管理 AWS Secrets Manager 263
Secrets Manager と Amazon DocumentDB の統合に関する制限事項 263
を使用したプライマリユーザーパスワードの管理の概要 AWS Secrets Manager 264
でのプライマリユーザーパスワードの Amazon DocumentDB 管理の強制 AWS Secrets Manager 264
Secrets Manager を使用したクラスターのプライマリユーザーパスワードの管理 265
データ保護 266
クライアント側のフィールドレベルの暗号化 267
保管中のデータの暗号化 275
Encrypting data in transit 280
キー管理 291
Identity and Access Management 291
対象者 292
アイデンティティを使用した認証 292
ポリシーを使用したアクセスの管理 296
Amazon DocumentDB と IAM との連携方法 299
アイデンティティベースのポリシーの例 306
トラブルシューティング 310
Amazon DocumentDB リソースへのアクセス許可の管理 312
アイデンティティベースのポリシー (IAM ポリシー) の使用 317
AWS Amazon DocumentDB の マネージドポリシー 322
Amazon DocumentDB API 許可リファレンス 341
IAM ID を使用した認証 350
IAM ユーザーとロールの開始方法 350
コンピューティングの設定 354
認証リクエストのモニタリング 354
IAM 認証の使用 355
IAM をサポートするドライバー 355

IAM ID 認証に関するよくある質問 355
Amazon DocumentDB ユーザーの管理 356
プライマリおよび serviceadmin ユーザ 357
追加ユーザーの作成 357
パスワードの自動ローテーション 359
ロールベースのアクセスコントロール 360
RBAC の概念 361
RBAC 組み込みロールの開始方法 363
RBAC ユーザー定義ロールの開始方法 367
ユーザーとして Amazon DocumentDB に接続する 371
一般的なコマンド 373
機能的な違い 378
制限 378
ロールベースのアクセス制御 (RBAC) を使用したデータベースへのアクセス 379
ログ記録とモニタリング 388
証明書の更新 388
アプリケーションと Amazon DocumentDB クラスターの更新 389
サーバー証明書の自動ローテーション 393
トラブルシューティング 394
よくある質問 395
証明書の更新 — GovCloud 401
アプリケーションと Amazon DocumentDB クラスターの更新 389
トラブルシューティング 394
よくある質問 395
コンプライアンス検証 413
耐障害性 415
インフラストラクチャセキュリティ 416
VPC エンドポイントAWS PrivateLink 417
VPC エンドポイントに関する考慮事項 418
利用可能なリージョン 418
Amazon DocumentDB API 用のインターフェイス VPC エンドポイントの作成 419
Amazon DocumentDB API 向け VPC エンドポイントポリシーの作成 420
セキュリティに関するベストプラクティス 421
監査イベント 422
サポートされるイベント 423
監査を有効にする 428

監査を無効にする ... 434
イベントへのアクセス ... 437
DML イベントをフィルタリングする ... 437
バックアップと復元 ... 444
バックアップと復元: 概念 ... 445
バックアップストレージの使用状況を確認する ... 447
データのダンプ、復元、インポート、およびエクスポート ... 448
mongodump ... 448
mongorestore ... 449
mongoexport ... 450
mongoimport ... 450
チュートリアル ... 451
クラスタースナップショットに関する考慮事項 ... 454
バックアップストレージ ... 454
バックアップウィンドウ ... 455
バックアップの保存期間 ... 457
クラスタースナップショットの暗号化をコピーする ... 457
自動スナップショットと手動スナップショットの比較 ... 458
手動クラスタースナップショットの作成 ... 460
クラスタースナップショットのコピー ... 463
共有スナップショットのコピー ... 464
間でのスナップショットのコピー AWS リージョン ... 464
制限 ... 464
暗号化の処理 ... 465
パラメータグループに関する考慮事項 ... 465
クラスタースナップショットのコピー ... 466
クラスターのスナップショットの共有 ... 473
暗号化されたスナップショットの共有 ... 473
スナップショットの共有 ... 477
クラスタースナップショットからの復元 ... 480
特定時点への復元 ... 488
クラスタースナップショットの削除 ... 494
Amazon DocumentDB の管理 ... 497
運用タスクの概要 ... 497
Amazon DocumentDB クラスターにレプリカを追加する ... 498
クラスターとインスタンスの説明 ... 499

クラスタースナップショットの作成 501
スナップショットからの復元 502
クラスターからインスタンスを削除する 503
クラスターの削除 504
グローバルクラスター 504
グローバルクラスターとは 504
グローバルクラスターはどのように役立つのか 505
グローバルクラスターの現在の制限は何ですか。 505
「クイックスタートガイド」 506
グローバルクラスターの管理 522
グローバルクラスターの接続 530
グローバルクラスターのモニタリング 531
ディザスタリカバリ 531
クラスターの管理 548
クラスターについて 549
[クラスター設定] 551
クラスターのストレージ設定 554
クラスターのステータスの評価 557
クラスターのライフサイクル 559
クラスターのスケーリング 602
クラスターボリュームのクローン作成 605
クラスターの耐障害性について 618
インスタンスの管理 620
インスタンスのステータスの判断 620
インスタンスのライフサイクル 620
インスタンスクラスの管理 645
NVMe-backed インスタンス 657
サブネットグループの管理 660
サブネットグループの作成 661
サブネットグループの説明 666
サブネットグループの変更 669
サブネットグループの削除 672
高可用性とレプリケーション 674
読み取りのスケーリング 674
高可用性 675
レプリカの追加 676

フェイルオーバー ... 677
レプリケーションの遅延 ... 681
インデックスの管理 ... 683
Amazon DocumentDB インデックスの作成 ... 683
Amazon DocumentDB インデックスのメンテナンス ... 689
ドキュメント圧縮の管理 ... 690
ドキュメント圧縮の管理 ... 691
ドキュメント圧縮のモニタリング ... 692
イベントの管理 ... 693
イベントカテゴリの表示 ... 694
Amazon DocumentDB イベントの表示 ... 696
リージョンとアベイラビリティーゾーンの選択 ... 699
利用可能なリージョン ... 700
クラスターパラメータグループの管理 ... 702
クラスターパラメータグループの説明 ... 703
クラスターパラメータグループの作成 ... 710
クラスターパラメータグループの変更 ... 713
カスタマイズしたクラスターパラメータグループを使用するためのクラスターの変更 ... 718
クラスターパラメータグループのコピー ... 719
クラスターパラメータグループのリセット ... 721
クラスターパラメータグループの削除 ... 724
クラスターパラメータリファレンス ... 727
エンドポイントについて ... 744
クラスターのエンドポイントの検索 ... 745
インスタンスのエンドポイントの検索 ... 747
エンドポイントへの接続 ... 751
Amazon DocumentDB ARN の理解 ... 752
ARN の構築 ... 752
ARN の検索 ... 756
リソースのタグ付け ... 758
リソースタグの概要 ... 759
タグの制約 ... 760
タグの追加または更新するには ... 760
タグを一覧表にする ... 761
タグの削除 ... 763
Amazon DocumentDB のメンテナンス ... 764

エンジンパッチ通知 ... 765
実施待ちのメンテナンスの確認 ... 767
エンジンの更新 ... 768
ユーザーが起動する更新 ... 772
メンテナンスウィンドウの管理 ... 773
オペレーティングシステムの更新 ... 775
サービスにリンクされたロールを理解する ... 779
サービスにリンクされたロールのアクセス許可 ... 780
サービスにリンクされたロールの作成 ... 782
サービスにリンクされたロールの変更 ... 782
サービスにリンクされたロールの削除 ... 782
Amazon DocumentDB サービスリンクロールがサポートされるリージョン ... 783
変更ストリームの使用 ... 783
サポートされているオペレーション ... 784
請求 ... 785
制限 ... 785
変更ストリームの有効化 ... 786
Python での変更ストリームの使用 ... 788
ドキュメント全体の参照 ... 790
変更ストリームの再開 ... 791
`startAtOperationTime` を用いて再開する ... 793
`postBatchResumeToken` を用いて再開する ... 795
トランザクション ... 796
ログ保持期間の変更 ... 796
セカンダリインスタンスでの変更ストリーム ... 799
変更ストリーム AWS Lambda での の使用 ... 801
制限 ... 802
Elastic クラスターの使用 ... 803
エラスティッククラスターのユースケース ... 803
ユーザープロファイル ... 804
コンテンツの管理と履歴の記録 ... 804
エラスティッククラスターの利点 ... 804
AWS サービス統合 ... 804
リージョンとバージョンの可用性 ... 805
Elastic クラスターのリージョン可用性 ... 805
バージョンの可用性 ... 806

制限 ... 806
エラスティッククラスターの管理 ... 806
クエリ操作と書き込み操作 ... 807
コレクションとインデックスの管理 ... 807
管理と診断 ... 807
オプトイン機能 ... 808
仕組み ... 808
Amazon DocumentDB エラスティッククラスターのシャーディング ... 808
エラスティッククラスターの移行 ... 812
エラスティッククラスターのスケーリング ... 812
エラスティッククラスターの信頼性 ... 812
エラスティッククラスターのストレージと可用性 ... 812
Amazon DocumentDB 4.0 と Elastic クラスター の機能上の違い ... 813
はじめに ... 814
前提条件 ... 815
ステップ 1: エラスティッククラスターを作成する ... 816
ステップ 2: Elastic クラスターに接続する ... 822
ステップ 3: コレクションをシャードし、データを挿入してクエリする ... 824
ステップ 4: 探索する ... 827
ベストプラクティス ... 827
シャードキーの選択 ... 828
接続管理 ... 828
非シャード型コレクション ... 828
エラスティッククラスターのスケーリング ... 829
エラスティッククラスターのモニタリング ... 829
Elastic クラスターの管理 ... 830
Elastic クラスター設定の変更 ... 830
Elastic クラスターのモニタリング ... 834
Elastic クラスターの削除 ... 838
Elastic クラスタースナップショットの管理 ... 840
Elastic クラスターの停止と開始 ... 855
エラスティッククラスターのメンテナンス ... 859
保管時のデータ暗号化 ... 867
Amazon DocumentDB エラスティッククラスターが で許可を使用する方法 AWS KMS ... 869
カスタマーマネージドキーを作成する ... 870
Amazon DocumentDB エラスティック クラスター の暗号化キーのモニタリング ... 871

詳細 ... 877
サービスにリンクされた役割 ... 877
エラスティッククラスターに対するサービスにリンクされたロールのアクセス許可 ... 878
Amazon DocumentDB のモニタリング ... 882
クラスターのステータスのモニタリング ... 883
クラスターステータスの値 ... 884
クラスターのステータスのモニタリング ... 885
インスタンスのステータスのモニタリング ... 887
インスタンスのステータス値 ... 888
AWS Management Console または を使用したインスタンスステータスのモニタリング AWS CLI ... 890
インスタンスのヘルスステータス値 ... 892
を使用したインスタンスのヘルスステータスのモニタリング AWS Management Console ... 893
Amazon DocumentDB に関する推奨事項の表示 ... 894
イベントサブスクリプション ... 897
イベントのサブスクライブ ... 898
サブスクリプションを管理する ... 901
カテゴリとメッセージ ... 905
Amazon DocumentDB と CloudWatch のモニタリング ... 908
Amazon DocumentDB のメトリクス ... 909
CloudWatch データの表示 ... 925
Amazon DocumentDB ディメンション ... 931
Opcounter メトリクスのモニタリング ... 932
データベース接続のモニタリング ... 932
CloudTrail を使用した Amazon DocumentDB API コールのログ記録 ... 932
CloudTrail 内の Amazon DocumentDB 情報 ... 933
オペレーションのプロファイリング ... 934
サポートされているオペレーション ... 935
制限 ... 935
プロファイラーの有効化 ... 935
プロファイラーの無効化 ... 940
プロファイラーログのエクスポートの無効化 ... 941
プロファイラーのログへのアクセス ... 943
一般的なクエリ ... 944
Performance Insights を使用したモニタリング ... 944
Performance Insights の概要 ... 946

Performance Insights の有効化と無効化 950
Performance Insights 用のアクセスポリシーの設定 953
Performance Insights ダッシュボードを使用してメトリクスを分析する 958
Performance Insights API によるメトリクスの取得 980
Performance Insights の Amazon CloudWatch メトリクス 995
カウンターメトリクス用の Performance Insights 997
OpenSearch 統合 999
送信先としての Amazon OpenSearch Service。 999
ステップ 1: Amazon OpenSearch Service ドメインまたは OpenSearch サーバーレスコレクションを作成する 1000
ステップ 2: Amazon DocumentDB クラスターで変更ストリームを有効にする 1000
ステップ 3: Amazon S3 バケットと送信先のドメインまたはコレクションへの書き込みアクセス許可を持つパイプラインロールを設定する 1000
ステップ 4: X-ENI を作成するために必要なアクセス許可をパイプラインロールに追加する 1001
ステップ 5: パイプラインを作成する 1002
制限 1003
DocumentDB サーバーレスの使用 1004
サーバーレスユースケース 1004
既存のプロビジョニングされたワークロードに Amazon DocumentDB サーバーレスを使用する 1006
Amazon DocumentDB サーバーレスの利点 1006
サーバーレスの仕組み 1007
概要 1008
Amazon DocumentDB クラスターの設定 1009
Amazon DocumentDB サーバーレススケーリング容量 1010
Amazon DocumentDB サーバーレススケーリング 1011
アイドル状態 (0.5 DCUs) 1013
サーバーレスの要件と制限 1014
機能の要件 1014
機能の制限 1015
サーバーレスを使用するクラスターの作成 1015
Amazon DocumentDB サーバーレスクラスターの作成 1016
Amazon DocumentDB サーバーレスインスタンスの追加 1019
サーバーレスへの移行 1019
既存の DocumentDB クラスターをサーバーレスに移行する 1020

MongoDB から DocumentDB サーバーレスへの移行 1025
サーバーレスの管理 1025
クラスターのスケーリング容量範囲設定の表示と変更 1025
サーバーレスリーダーの昇格階層の表示と変更 1028
サーバーレスインスタンスの制限 1031
サーバーレススケーリング設定 1033
DocumentDB サーバーレスクラスターのスケーリング容量範囲の選択 1033
DocumentDB サーバーレスクラスターMinCapacityの設定の選択 1034
DocumentDB サーバーレスクラスターMaxCapacityの設定の選択 1036
メモリ不足エラーを回避する 1038
サーバーレスインスタンスがスケールダウンしないのはなぜですか？ 1039
サーバーレスのモニタリング 1040
メモリ不足: incompatible-parameters ステータス 1040
DocumentDB サーバーレスの Amazon CloudWatch メトリクス 1040
Performance Insights による DocumentDB サーバーレスパフォーマンスのモニタリング . 1045
Amazon DocumentDB の操作 1046
DocumentDB に接続する 1046
プログラムによる接続 1046
Amazon VPC の外部からの接続 1074
レプリカセットとして接続する 1076
Studio 3T を使用して接続 1081
DataGrip を使用して接続する 1092
Amazon EC2 を使用した接続 1100
JDBC ドライバーによる接続 1138
ODBC ドライバーを使用して接続します 1156
DocumentDB を使用したプログラム 1172
DocumentDB Java プログラミングガイド 1172
JSON スキーマ検証の使用 1204
クォータと制限 1215
サポートされるインスタンスタイプ 1215
サポートされるリージョン 1217
リージョン別クォータ 1219
集約限度 1221
クラスター限度 1221
インスタンス制限 1223
命名に関する制約 1225

TTL の制約 1227
エラスティッククラスターの制限 1227
エラスティッククラスターのシャード制限 1228
1 シャードあたりの Elastic クラスターの CPU、メモリ、接続、カーソル限度 1229
クエリ 1230
ドキュメントのクエリ 1230
すべてのドキュメントを取得する 1231
フィールド値の一致 1231
埋め込まれたドキュメント 1231
埋め込まれたドキュメントのフィールド値 1232
配列の一致 1232
配列内の値の一致 1232
演算子の使用 1233
クエリプラン 1233
クエリプラン 1233
クエリプランキャッシュ 1235
結果の説明 1235
スキャンとフィルタのステージ 1236
インデックス公差 1237
インデックス結合 1238
複数のインデックス交差/結合 1238
複合インデックス 1239
ソートステージ 1240
グループステージ 1240
地理空間データ 1240
概要 1
地理空間データのインデックス作成と格納 1241
地理空間データのクエリ 1243
制限 1246
部分インデックス 1246
部分インデックスを作成する 1247
サポートされている演算子 1247
部分インデックスを使用したクエリ 1247
部分インデックス機能 1248
部分インデックスに関する制限事項 1253
テキスト検索 1254

サポートされている機能 1254
Amazon DocumentDB テキストインデックスの使用 1255
MongoDB との違い 1260
ベストプラクティスとガイドライン 1261
制限 1261
トラブルシューティング 1262
接続の問題 1262
Amazon DocumentDB エンドポイントに接続できない 1262
Amazon DocumentDB インスタンスへの接続のテスト 1268
無効なエンドポイントへの接続 1268
接続数に影響するドライバー設定 1269
インデックスの作成 1269
インデックスの構築が失敗する 1269
バックグラウンドインデックスビルドのレイテンシーの問題と失敗 1270
パフォーマンスとリソース使用率 1270
挿入、更新、削除に関する統計の表示 1271
キャッシュのパフォーマンスの分析 1273
長時間実行されているクエリやブロックされているクエリを見つけて終了する 1274
クエリプランを参照してクエリを最適化する 1276
Elastic クラスターのクエリプランを確認する方法 1278
インスタンスで実行中のすべての操作をリストする 1280
クエリが進行するタイミングを知る 1283
システムの実行速度が突然遅くなった理由を判断する 1285
CPU 使用率が高くなる原因を特定する 1287
インスタンス上で開いているカーソルを検索します。 1288
Amazon DocumentDB エンジンの現在のバージョンを表示します 1288
インデックスの使用状況を分析し、未使用のインデックスを特定します。 1289
見逃したインデックスの特定 1291
便利なクエリの概要 1293
リソース管理 API リファレンス 1295
アクション 1295
Amazon DocumentDB (with MongoDB compatibility) 1298
Amazon DocumentDB Elastic Clusters 1488
データ型 1561
Amazon DocumentDB (with MongoDB compatibility) 1563
Amazon DocumentDB Elastic Clusters 1648

共通エラー 1667
共通パラメータ 1668
リリースノート 1671
2025 年 7 月 30 日 1673
新機能 1673
2025 年 7 月 29 日 1674
新機能 1674
バグの修正とその他の変更 1675
2025 年 6 月 18 日 1675
新機能 1675
2025 年 5 月 8 日 1675
新機能 1675
2025 年 4 月 2 日 1675
バグの修正とその他の変更 1675
2025 年 3 月 24 日 1676
新機能 1676
2025 年 2 月 6 日 1676
新機能 1676
2025 年 1月 28 日 1676
新機能 1676
2025年 1 月 15 日 1676
新機能 1676
バグの修正とその他の変更 1677
2024 年 12 月 18 日 1677
新機能 1677
2024 年 11 月 12 日 1677
新機能 1677
2024 年 11 月 6 日 1678
新機能 1678
2024 年 11 月 1 日 1678
新機能 1678
2024 年 10 月 22 日 1678
新機能 1678
2024 年 9 月 18 日 1679
新機能 1679
2024 年 9 月 17 日 1679

新機能 1679
バグの修正とその他の変更 1679
2024 年 8 月 22 日 1680
新機能 1680
2024 年 8 月 20 日 1680
新機能 1680
2024 年 8 月 8 日 1680
新機能 1680
2024 年 7 月 23 日 1681
新機能 1681
バグの修正とその他の変更 1682
2024 年 7 月 22 日 1682
新機能 1682
バグの修正とその他の変更 1683
2024 年 7 月 9 日 1683
新機能 1683
2024 年 7 月 8 日 1683
新機能 1683
2024 年 6 月 25 日 1684
新機能 1684
2024 年 5 月 29 日 1684
新機能 1684
2024 年 4 月 3 日 1684
新機能 1684
バグの修正とその他の変更 1685
2024 年 2 月 22 日 1685
新機能 1685
2024 年 1 月 30 日 1686
新機能 1686
2024 年 1 月 10 日 1686
新機能 1686
バグの修正とその他の変更 1687
2023 年 12 月 20 日 1688
その他の変更 1688
2023 年 12 月 13 日 1688
新機能 1688

2023 年 11 月 29 日 1688
新機能 1688
2023 年 11 月 21 日 1688
新機能 1688
2023 年 11 月 17 日 1689
新機能 1689
バグの修正とその他の変更 1689
2023 年 11 月 6 日 1689
新機能 1689
バグの修正とその他の変更 1690
2023 年 9 月 25 日 1690
新機能 1690
2023 年 9 月 20 日 1690
新機能 1690
2023 年 9 月 15 日 1690
新機能 1690
2023 年 9 月 11 日 1690
新機能 1690
2023 年 8 月 3 日 1691
新機能 1691
2023 年 7 月 13 日 1691
新機能 1691
バグの修正とその他の変更 1691
2023 年 6 月 7 日 1692
バグの修正とその他の変更 1692
2023 年 5 月 10 日 1692
バグの修正とその他の変更 1692
2023 年 4 月 4 日 1692
バグの修正とその他の変更 1692
2023 年 3 月 22 日 1693
新機能 1693
2023 年 3 月 1 日 1693
新機能 1693
2023 年 2 月 27 日 1694
バグの修正とその他の変更 1694
2023 年 2 月 2 日 1694

バグの修正とその他の変更 1694
2022 年 11 月 30 日 1695
新機能 1695
2022 年 8 月 9 日 1695
新機能 1695
バグの修正とその他の変更 1695
2022 年 7 月 25 日 1696
新機能 1696
2022 年 6 月 27 日 1696
新機能 1696
2022 年 4 月 29 日 1696
新機能 1696
2022 年 4 月 7 日 1696
新機能 1696
2022 年 3 月 16 日 1697
新機能 1697
2022 年 2 月 8 日 1697
新機能 1697
2022 年 1 月 24 日 1697
新機能 1697
2022 年 1 月 21 日 1698
新機能 1698
2021 年 10 月 25 日 1698
新機能 1698
バグの修正とその他の変更 1699
2021 年 6 月 24 日 1699
新機能 1699
2021 年 5 月 4 日 1699
新機能 1699
バグの修正とその他の変更 1700
2021 年 1 月 15 日 1700
新機能 1700
2020 年 11 月 9 日 1701
新機能 1701
バグの修正とその他の変更 1702
2020 年 10 月 30 日 1703

新機能 1703
バグの修正とその他の変更 1703
2020 年 9 月 22 日 1704
新機能 1704
バグの修正とその他の変更 1704
2020 年 7 月 10 日 1704
新機能 1704
バグの修正とその他の変更 1704
2020 年 6 月 30 日 1705
新機能 1705
バグの修正とその他の変更 1705
ドキュメント履歴 1706
..... mdccxviii

Amazon DocumentDB (MongoDB 互換) とは

Amazon DocumentDB (MongoDB 互換) は高速で信頼性が高く、完全マネージド型のデータベースサービスです。Amazon DocumentDB は、クラウド内で MongoDB と互換性のあるデータベースを簡単にセットアップ、運用、スケールすることができます。Amazon DocumentDB では、同じアプリケーションコードを実行し、MongoDB で使用するのと同じドライバーとツールを使用できます。

Amazon DocumentDB を使用する前に、「仕組み」で説明されている概念と機能を確認しておく必要があります。その後で、「入門ガイド」の各ステップを実行します。

トピック

- Amazon DocumentDB の概要
- クラスター
- インスタンス
- リージョンとアベイラビリティーゾーン
- Amazon DocumentDB の料金
- モニタリング
- インターフェイス
- 次のステップ
- Amazon DocumentDB の 仕組み
- ドキュメントデータベースとは

Amazon DocumentDB の概要

次に示すのは Amazon DocumentDB の主な機能の一部です。

- Amazon DocumentDB は、インスタンスベースのクラスターとエラスティッククラスターの 2 種類のクラスターをサポートしています。エラスティッククラスターは、1 秒あたり数百万回の読み取り/書き込みとペタバイトのストレージ容量を持つワークロードをサポートします。エラスティッククラスターの詳細については、「Amazon DocumentDB のエラスティッククラスター」を参照してください。以下の内容は Amazon DocumentDB インスタンスベースのクラスターに関するものです。
- Amazon DocumentDB は、データベースのストレージニーズが増大すると自動的にストレージボリュームのサイズを拡張します。ストレージボリュームは 10 GB ごとに最大 128 TiB まで拡張さ

れます。将来の拡張に備えてクラスターに余分なストレージをプロビジョニングする必要はありません。

- Amazon DocumentDB では、最大 15 個のレプリカインスタンスを作成して、大量のアプリケーションリクエストをサポートように読み取りスループットを増やすことができます。Amazon DocumentDB レプリカは同じ基盤となるストレージを共有しているため、コストを削減でき、レプリカノードで書き込みを実行する必要はありません。この機能により、読み取り要求を処理するための処理能力が解放され、レプリカのラグタイム (多くの場合、一桁ミリ秒に短縮されます) が短縮されます。ストレージボリュームのサイズに関係なく、レプリカを数分で追加できます。Amazon DocumentDB にはリーダーエンドポイントも提供するため、アプリケーションは追加および削除されるときにレプリカを追跡することなく接続できます。
- Amazon DocumentDB では、各インスタンスを拡大または縮小するため、コンピューティングおよびメモリリソースをスケールすることができます。通常、コンピューティングのスケーリングは数分以内に完了します。
- Amazon DocumentDB は Amazon VPC (Amazon Virtual Private Cloud) で実行されるため、独自の仮想ネットワークでデータベースを分離することができます。また、ファイアウォール設定を指定して、クラスターへのネットワークアクセスを制御することができます。
- Amazon DocumentDB はクラスターの正常性を継続的にモニタリングします。インスタンスに障害が発生すると、Amazon DocumentDB はインスタンスと関連するプロセスを自動的に再起動します。Amazon DocumentDB では、データベース REDO ログのクラッシュリカバリのリプレイが必要ないため、再起動時間が大幅に短縮されます。Amazon DocumentDB は、データベースキャッシュをデータベースプロセスから分離し、インスタンスの再起動後もキャッシュを有効にします。
- インスタンスに障害が発生した場合、Amazon DocumentDB は、他のアベイラビリティーゾーンで作成する最大 15 個の Amazon DocumentDB レプリカの 1 つに自動的にフェイルオーバーします。レプリカがプロビジョニングされていない場合に障害が発生すると、Amazon DocumentDB は自動的に新しい Amazon DocumentDB インスタンスの作成を試みます。
- Amazon DocumentDB のバックアップ機能により、クラスターのポイントインタイムリカバリが可能です。この機能によって、最大 5 分前まで、保持期間内の任意の時点にクラスターを復元させることができます。自動バックアップ保持期間は、最大 35 日間まで設定できます。自動化されたバックアップは、99.999999999% の耐久性を持つように設計された Amazon S3 (Amazon Simple Storage Service) に保存されます。Amazon DocumentDB バックアップは自動、増分、継続的であり、クラスターのパフォーマンスへの影響はありません。
- Amazon DocumentDB では、 AWS Key Management Service () を使用して作成および制御するキーを使用してデータベースを暗号化できますAWS KMS。Amazon DocumentDB 暗号化を使用して実行されているデータベースクラスターでは、基盤となるストレージに保存されている保管中の

データは暗号化されます。同じクラスター内の自動バックアップ、スナップショット、およびレプリカも暗号化されます。

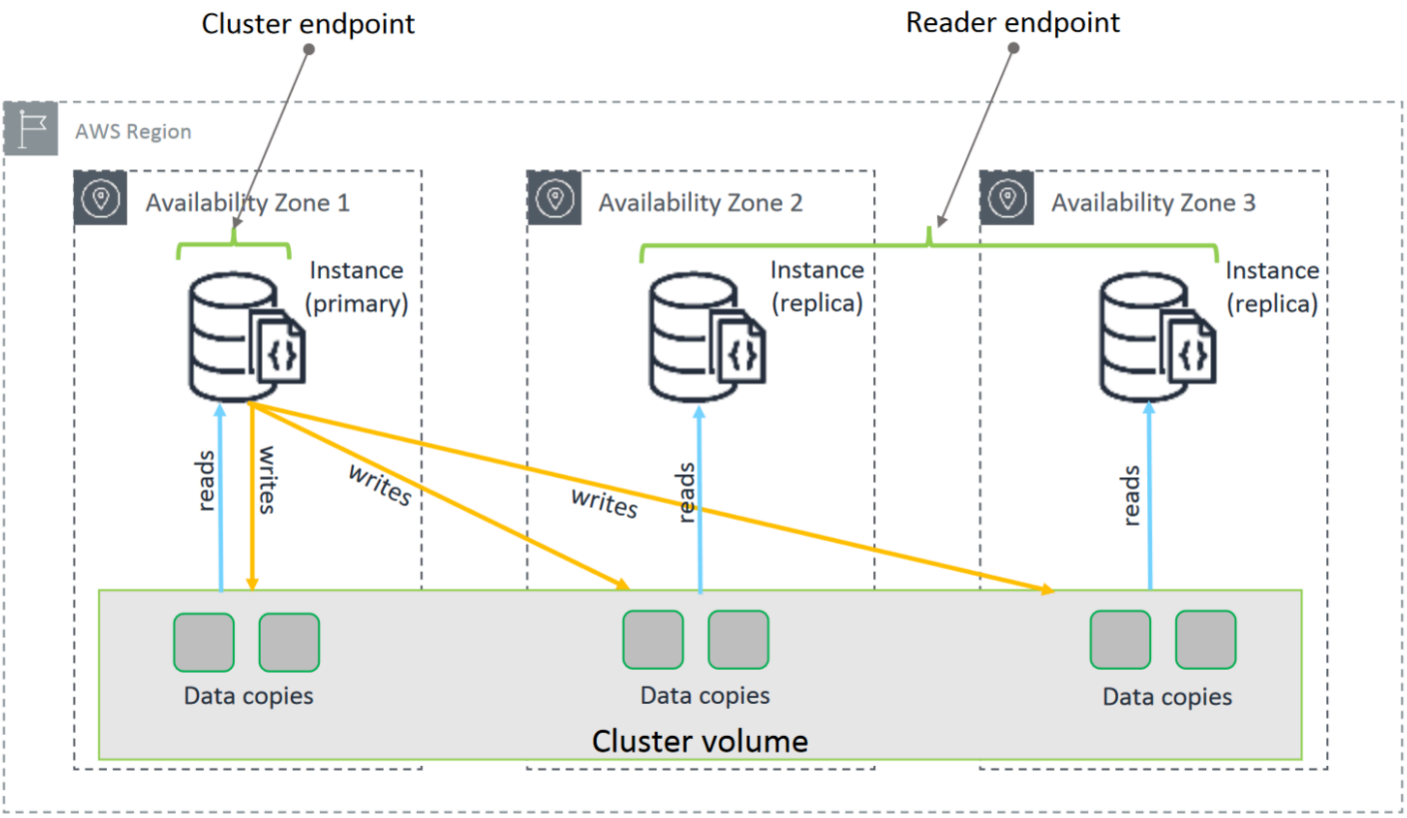
- Amazon DocumentDB は、「連邦リスク承認管理プログラム (FedRAMP)」の認可事業者です。これには AWS GovCloud (米国) リージョンの FedRAMP High 認可と AWS 、米国東部/西部リージョンの FedRAMP Moderate 認可があります。 AWS とコンプライアンスの取り組みの詳細については、AWS「コンプライアンスプログラムによる対象範囲内のサービス」を参照してください。

AWS サービスを初めて使用する場合は、次のリソースを使用して詳細を確認してください。

- AWS は、コンピューティング、データベース、ストレージ、分析、その他の機能のためのサービスを提供します。すべての AWS サービスの概要については、「Amazon Web Services によるクラウドコンピューティング」を参照してください。
- AWS には、多数のデータベースサービスが用意されています。使用している環境に最適なサービスに関するガイダンスについては、「AWSでのデータベース」を参照してください。

クラスター

クラスター は、0 ~ 16 のインスタンスと、これらのインスタンスのデータを管理する 1 つのクラスターストレージボリュームで構成されます。すべての書き込みはプライマリインスタンスを介して行われます。すべてのインスタンス (プライマリとレプリカ) は読み込みをサポートします。クラスターのデータはクラスターボリュームに保存され、3 つの異なるアベイラビリティーゾーンにコピーが保存されます。



Amazon DocumentDB 5.0 インスタンスベースのクラスターは、Amazon DocumentDB 標準と Amazon DocumentDB I/O 最適化の 2 つの Amazon DocumentDB ストレージ設定をサポートしています。詳細については、「Amazon DocumentDB クラスターストレージ設定」を参照してください。

インスタンス

Amazon DocumentDB インスタンスは、クラウド内の独立したデータベース環境です。インスタンスには、複数のユーザーが作成したデータベースを含めることができます。または を使用してインスタンスを作成および変更できます AWS Management Console AWS CLI。

インスタンスの計算とメモリの容量は、インスタンスクラス によって決まります。お客様のニーズに最も合うインスタンスを選択できます。時間の経過とともにニーズが変化した場合は、別のインスタンスクラスを選択できます。インスタンスクラスの仕様については、「インスタンスクラスの仕様」を参照してください。

Amazon DocumentDB インスタンスは、Amazon VPC 環境でのみ実行されます。Amazon VPC により仮想ネットワーク環境を管理できます。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コントロールリスト (ACL) を設定できます。

Amazon DocumentDB インスタンスを作成する前に、クラスターを作成してインスタンスを含める必要があります。

すべてのインスタンスクラスがすべてのリージョンでサポートされているわけではありません。次の表では、どのインスタンスクラスが各リージョンでサポートされているかを示しています。

> **Note**
>
> 各インスタンスクラスの Amazon DocumentDB でサポートされているインスタンスタイプの詳細なリストについては、「」を参照してくださいインスタンスクラスの仕様。

リージョン別のサポートされるインスタンスクラス

リージョン	R6GD	R6G	R5	R4	T4G	T3
米国東部 (オハイオ)	サポート	サポート対象	サポート対象	サポート対象	サポート対象	サポート
米国東部 (バージニア北部)	サポート	サポート対象	サポート対象	サポート対象	サポート対象	サポート
米国西部 (オレゴン)	サポート	サポート対象	サポート対象	サポート対象	サポート対象	サポート
アフリカ (ケープタウン)		サポート	サポート対象		サポート対象	サポート
南米 (サンパウロ)	サポート	サポー	サポート対象		サポー	サポート

リージョン	R6GD	R6G	R5	R4	T4G	T3
		ト対象			ト対象	
アジアパシフィック (香港)		サポート	サポート対象		サポート対象	サポート
アジアパシフィック (ハイデラバード)			サポート			サポート
アジアパシフィック (ムンバイ)	サポート	サポート対象	サポート対象		サポート対象	サポート
アジアパシフィック (ソウル)	サポート	サポート対象	サポート対象		サポート対象	サポート
アジアパシフィック (シドニー)	サポート	サポート対象	サポート対象		サポート対象	サポート
アジアパシフィック (シンガポール)	サポート	サポート対象	サポート対象		サポート対象	サポート
アジアパシフィック (東京)	サポート	サポート対象	サポート対象		サポート対象	サポート

リージョン	R6GD	R6G	R5	R4	T4G	T3
カナダ (中部)	サポート	サポート対象	サポート対象		サポート対象	サポート
欧州 (フランクフルト)	サポート	サポート対象	サポート対象		サポート対象	サポート
欧州 (アイルランド)	サポート	サポート対象	サポート対象	サポート対象	サポート対象	サポート
欧州 (ロンドン)	サポート	サポート対象	サポート対象		サポート対象	サポート
欧州 (ミラノ)		サポート	サポート対象		サポート対象	サポート
欧州 (パリ)	サポート	サポート対象	サポート対象		サポート対象	サポート
欧州 (スペイン)		サポート	サポート対象		サポート対象	サポート

リージョン	R6GD	R6G	R5	R4	T4G	T3
欧州 (ストックホルム)		サポート	サポート対象		サポート対象	サポート
中東 (アラブ首長国連邦)		サポート	サポート対象		サポート対象	サポート
中国 (北京)	サポート	サポート対象	サポート対象		サポート対象	サポート
中国 (寧夏)		サポート	サポート対象		サポート対象	サポート
イスラエル (テルアビブ)		サポート	サポート対象		サポート対象	サポート
AWS GovCloud (米国西部)	サポート	サポート対象	サポート対象		サポート対象	サポート
AWS GovCloud (米国東部)	サポート	サポート対象	サポート対象		サポート対象	サポート

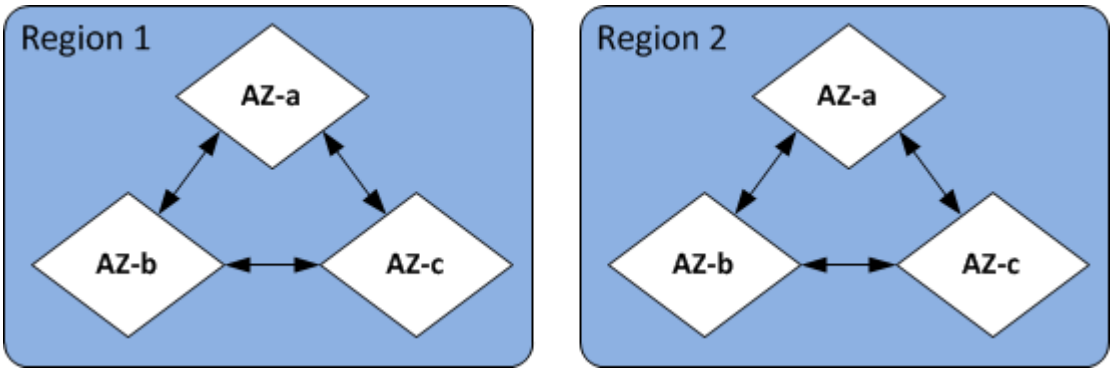
リージョンとアベイラビリティーゾーン

リージョンとアベイラビリティーゾーンは、クラスターとインスタンスの物理的な場所を定義します。

リージョン

AWS クラウドコンピューティングリソースは、世界中のさまざまな地域 (北米、欧州、アジアなど) にある高可用性データセンター施設に収容されています。各データセンターの場所は、リージョンと呼ばれます。

各 AWS リージョンは、他の AWS リージョンから完全に分離されるように設計されています。各リージョン内には複数のアベイラビリティーゾーンがあります。別のアベイラビリティーゾーンでノードを起動して、最大限の耐障害性を実現できます。次の図は、 AWS リージョンとアベイラビリティーゾーンの仕組みの概要を示しています。



アベイラビリティーゾーン

各 AWS リージョンには、アベイラビリティーゾーンと呼ばれる複数の異なる場所が含まれています。各アベイラビリティーゾーンは、他のアベイラビリティーゾーンにおける障害の影響は受けず、同じリージョン内の他のアベイラビリティーゾーンには、低コスト、低レイテンシーでネットワーク接続できるように設計されています。複数のアベイラビリティーゾーンで特定のクラスターのインスタンスを起動することにより、アベイラビリティーゾーンが失敗するというまれなイベントからアプリケーションを保護できます。

Amazon DocumentDB アーキテクチャは、ストレージとコンピューティングを分離します。ストレージレイヤーの場合、Amazon DocumentDB は 3 つの AWS アベイラビリティーゾーンにデータの 6 つのコピーをレプリケートします。例えば、2 つのアベイラビリティーゾーンのみをサポートするリージョンで Amazon DocumentDB クラスターを起動している場合、データストレージは 3 つのアベイラビリティーゾーンにわたって 6 つの方法でレプリケートされますが、コンピューティングインスタンスは 2 つのアベイラビリティーゾーンでしか使用できません。

次の表に、特定の でクラスターのコンピューティングインスタンスをプロビジョニング AWS リージョン するために使用できるアベイラビリティーゾーンの数を示します。

リージョン名	リージョン	アベイラビリティーゾーン (コンピューティング)
米国東部 (オハイオ)	us-east-2	3
米国東部 (バージニア北部)	us-east-1	6
米国西部 (オレゴン)	us-west-2	4
アフリカ (ケープタウン)	af-south-1	3
南米 (サンパウロ)	sa-east-1	3
アジアパシフィック (香港)	ap-east-1	3
アジアパシフィック (ハイデラバード)	ap-south-2	3
アジアパシフィック (ムンバイ)	ap-south-1	3
アジアパシフィック (ソウル)	ap-northeast-2	4
アジアパシフィック (シンガポール)	ap-southeast-1	3
アジアパシフィック (シドニー)	ap-southeast-2	3
アジアパシフィック (東京)	ap-northeast-1	3
カナダ (中部)	ca-central-1	3

リージョン名	リージョン	アベイラビリティーゾーン (コンピューティング)
中国 (北京) リージョン	cn-north-1	3
中国 (寧夏)	cn-northwest-1	3
欧州 (フランクフルト)	eu-central-1	3
欧州 (アイルランド)	eu-west-1	3
欧州 (ロンドン)	eu-west-2	3
欧州 (ミラノ)	eu-south-1	3
欧州 (パリ)	eu-west-3	3
欧州 (スペイン)	eu-south-2	3
欧州 (ストックホルム)	eu-north-1	3
中東 (アラブ首長国連邦)	me-central-1	3
イスラエル (テルアビブ)	il-central-1	3
AWS GovCloud (米国西部)	us-gov-west-1	3
AWS GovCloud (米国東部)	us-gov-east-1	3

Amazon DocumentDB の料金

Amazon DocumentDB クラスターは、以下のコンポーネントに基づいて請求されます。

- インスタンス時間 (1 時間あたり) - インスタンスのインスタンスクラスに基づきます (例: db.r5.xlarge)。料金は 1 時間単位で表示されますが、請求の計算方法には秒単位が適用され、

時間は 10 進形式で表示されます。Amazon DocumentDB の使用料は 1 秒ごとに課金され、10 分未満の場合は 10 分の料金が発生します。詳細については、「インスタンスクラスの管理」を参照してください。

- I/O リクエスト (1 か月あたりの 100 万リクエストごと) - 請求サイクル内で行ったストレージ I/O リクエストの合計数。
- バックアップストレージ (1 か月当たりの GiB) - バックアップストレージは、自動データベースバックアップおよび作成したアクティブなデータベースのスナップショットに関連付けられているストレージです。バックアップ保持期間を延長するか、追加のデータベーススナップショットを撮ると、データベースが消費するバックアップストレージが増加します。バックアップストレージは GB 月単位で計算され、1 秒単位は適用されません。詳細については、「Amazon DocumentDB でのバックアップと復元」を参照してください。
- データ転送 (GB あたり) — インターネットまたは他の AWS リージョンとの間でインスタンスとの間で送受信されるデータ転送。

詳細については、「Amazon DocumentDB の料金」を参照してください。

無料トライアル

Amazon DocumentDB は、1 か月間の無料トライアルを使用して無料でお試しいただけます。詳細については、「Amazon DocumentDB の料金」の「無料トライアル」または「Amazon DocumentDB 無料トライアルのよくある質問」を参照してください。

モニタリング

インスタンスのパフォーマンスと正常性を追跡する方法は複数あります。無料の Amazon CloudWatch サービスを使用して、インスタンスのパフォーマンスと正常性をモニタリングできます。Amazon DocumentDB コンソールでパフォーマンスチャートを見つけることができます。インスタンス、スナップショット、パラメータグループ、セキュリティグループで変更が発生したときに通知される Amazon DocumentDB のイベントにサブスクライブできます。

詳細については次を参照してください:

- Amazon DocumentDB と CloudWatch のモニタリング
- を使用した Amazon DocumentDB API コールの口グ記録 AWS CloudTrail

インターフェイス

Amazon DocumentDB を操作するには、 AWS Management Console や など、複数の方法があります AWS CLI。

AWS Management Console

AWS Management Console はシンプルなウェブベースのユーザーインターフェイスです。プログラミングなしでコンソールからクラスターとインスタンスを管理できます。Amazon DocumentDB コンソールにアクセスするには、 にサインイン AWS Management Console し、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。

AWS CLI

AWS Command Line Interface （ AWS CLI) を使用して、Amazon DocumentDB クラスターとインスタンスを管理できます。最小限の設定で、使い慣れたターミナルプログラムから、Amazon DocumentDB コンソールで提供されるすべての機能の使用を開始できます。

- をインストールするには AWS CLI、AWS 「 コマンドラインインターフェイスのインストール」を参照してください。
- Amazon DocumentDB AWS CLI の の使用を開始するには、AWS Amazon DocumentDB のコマンドラインインターフェイスリファレンス」を参照してください。

MongoDB ドライバー

Amazon DocumentDB クラスターに対するアプリケーションの開発と作成に、MongoDB ドライバーを Amazon DocumentDB で使用することもできます。詳細については、 TLS が有効な場合の接続または の MongoDB シェルタブを参照してくださいTLS が無効な場合の接続。

次のステップ

前のセクションでは、Amazon DocumentDB が提供する基本的なインフラストラクチャのコンポーネントを紹介しました。次に実行すべきこと 状況に応じて、以下のトピックの 1 つを参照し、使用を開始してください。

- を使用してクラスターとインスタンスを作成して、Amazon DocumentDB の使用を開始します AWS CloudFormation を使用した Amazon DocumentDB クイックスタート AWS CloudFormation。

- 「入門ガイド」の手順を使用してクラスターとインスタンスを作成することにより、Amazon DocumentDB の使用を開始します。
- Amazon DocumentDB エラスティッククラスターの開始方法 の手順を使用してエラスティッククラスターを作成することにより、Amazon DocumentDB を開始します。
- Amazon DocumentDB への移行 のガイダンスを使用して、MongoDB 実装を Amazon DocumentDB に移行します。

Amazon DocumentDB の 仕組み

Amazon DocumentDB (MongoDB 互換) は、フルマネージドの MongoDB と互換性のあるデータベースサービスです。Amazon DocumentDB では、MongoDB と同じアプリケーションコードを実行し、同じドライバーとツールを使用できます。Amazon DocumentDB は、MongoDB 3.6、4.0 および 5.0 と互換性を持っています。

トピック

- Amazon DocumentDB エンドポイント
- TLS サポート
- Amazon DocumentDB のストレージ
- Amazon DocumentDB レプリケーション
- Amazon DocumentDB の信頼性
- 読み込み設定のオプション
- TTL 削除
- 有料リソース

Amazon DocumentDB を使用するときは、最初に クラスター を作成します。クラスターは、ゼロ以上のデータベースインスタンスと、これらのインスタンスのデータを管理する 1 つのクラスターボリュームで構成されます。Amazon DocumentDB の クラスターボリューム は、複数のアベイラビリティーゾーンにまたがる仮想データベースストレージボリュームです。アベイラビリティーゾーンごとに、クラスターデータのコピーがあります。

Amazon DocumentDB クラスターは、主要な 2 つのコンポーネントで構成されています。

- クラスターボリューム - クラウドネイティブなストレージサービスを使用して、3 つのアベイラビリティーゾーンにわたって 6 つの方法でデータをレプリケートし、耐久性と可用性に優れたス

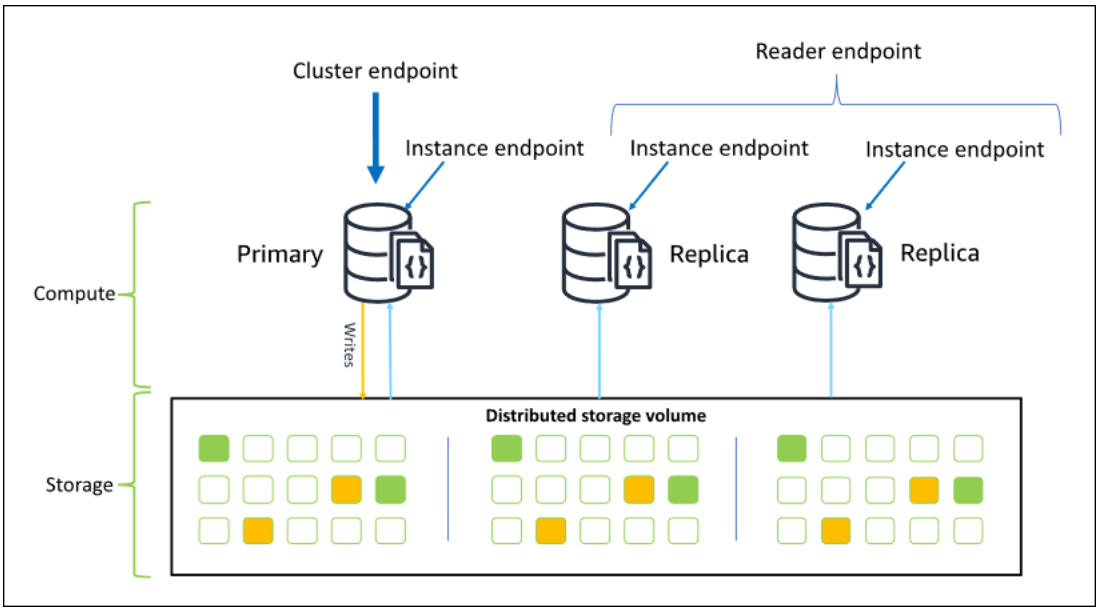
トレージを提供します。Amazon DocumentDB クラスターには 1 個のクラスターボリュームがあり、最大 128 TiB のデータを格納できます。

- インスタンス - データベースの処理能力を提供し、クラスターストレージボリュームとの間でデータの書き込みと読み取りを行います。Amazon DocumentDB クラスターには 0 ~ 16 のインスタンスを持つことができます。

インスタンスは以下の 2 つのロールのいずれかを提供します。

- プライマリインスタンス - 読み書きオペレーションをサポートし、クラスターボリュームに対するすべてのデータ変更を実行します。各 Amazon DocumentDB クラスターには 1 つのプライマリインスタンスがあります。
- レプリカインスタンス - 読み取りオペレーションのみをサポートします。Amazon DocumentDB クラスターには、プライマリインスタンスに加えて 15 個までのレプリカを含めることができます。複数のレプリカがあると、読み取りワークロードを分散させることができます。さらに、別のアベイラビリティーゾーンにレプリカを配置することによって、クラスターの可用性を高めることもできます。

次の図は、Amazon DocumentDB クラスター内のクラスターボリューム、プライマリインスタンス、およびレプリカの関係を示しています。



クラスターインスタンスは、同じインスタンスクラスである必要はなく、必要に応じてプロビジョニングおよび終了することができます。このアーキテクチャでは、ストレージとは無関係にクラスターのコンピューティング性能をスケールできます。

アプリケーションがプライマリインスタンスにデータを書き込むときに、プライマリはクラスターボリュームに堅牢な書き込みを実行します。次に、その書き込み (データではありません) の状態がアクティブな各レプリカにレプリケートされます。Amazon DocumentDB レプリカは書き込み処理に関与しないため、Amazon DocumentDB レプリカは読み取りスケーリングにとって有利となります。Amazon DocumentDB レプリカからの読み取りは、最短のレプリカラグで結果整合性があります。通常、プライマリインスタンスがデータを書き込んだ後、100 ミリ秒未満になります。レプリカからの読み込みは、プライマリに書き込まれた順序で読み込まれることが保証されます。レプリカラグは、データ変更のレートによって異なり、書き込みアクティビティが多い期間はレプリカラグが大きくなる可能性があります。詳細については、「Amazon DocumentDB のメトリクス」で `ReplicationLag` のメトリクスを参照してください。

Amazon DocumentDB エンドポイント

Amazon DocumentDB は複数の接続オプションを提供し、幅広いユースケースに対応します。Amazon DocumentDB クラスターのインスタンスに接続するには、インスタンスのエンドポイントを指定します。エンドポイントでは、ホストアドレスとポート番号がコロンで区切られています。

リーダーエンドポイントまたはインスタンスエンドポイントに接続する特定のユースケースがない限り、クラスターエンドポイントを使用してレプリカセットモード (「レプリカセットとしての Amazon DocumentDB に接続する」を参照) でクラスターに接続することをお勧めします。レプリカにリクエストをルーティングするには、アプリケーションの読み込み整合性要件に合わせて読み取りスケーリングを最大化する、ドライバー読み込み設定を選択します。`secondaryPreferred` 読み込み設定を使用すると、レプリカの読み取りを有効にし、プライマリインスタンスを解放してより多くの作業を行うことができます。

次のエンドポイントは、Amazon DocumentDB クラスターから取得できます。

クラスターエンドポイント

クラスターエンドポイントはクラスターの現在のプライマリインスタンスに接続します。このクラスターエンドポイントは、読み取りと書き込みの両方のオペレーションに使用できます。Amazon DocumentDB クラスターには、厳密には 1 つのクラスターエンドポイントがあります。

クラスターエンドポイントは、クラスターへの読み取り/書き込み接続のフェイルオーバーサポートを提供します。クラスターの現在のプライマリインスタンスが失敗し、クラスターに少なくとも 1 つのアクティブなリードレプリカがある場合、クラスターエンドポイントは自動的に接続リクエストを新しいプライマリインスタンスにリダイレクトします。Amazon DocumentDB クラスターに接続

するときは、クラスターエンドポイントを使用し、レプリカセットモード (「レプリカセットとしての Amazon DocumentDB に接続する」を参照) でクラスターに接続することをお勧めします。

Amazon DocumentDB クラスターエンドポイントの例を以下に示します。

```
sample-cluster.cluster-123456789012.us-east-1.docdb.amazonaws.com:27017
```

次の接続文字列は、このクラスターエンドポイントを使用した例です。

```
mongodb://username:password@sample-cluster.cluster-123456789012.us-
east-1.docdb.amazonaws.com:27017
```

クラスターのエンドポイントを見つける方法については、「クラスターのエンドポイントの検索」を参照してください。

リーダーエンドポイント

リーダーエンドポイントは、クラスター内の使用可能なすべてのレプリカ間で、読み取り専用接続の負荷分散を行います。クラスターリーダーエンドポイントは、replicaSet モードで接続する場合、クラスターエンドポイントとして機能します。つまり、接続文字列では、レプリカセットパラメータは &replicaSet=rs0 となります。この場合、プライマリに対して書き込みオペレーションを実行できます。ただし、directConnection=true を指定してクラスターに接続した後、リーダーエンドポイントへの接続を経由して書き込みオペレーションを実行しようとすると、エラーが発生します。Amazon DocumentDB クラスターには、厳密には 1 つのリーダーエンドポイントがあります。

クラスターに 1 つの (プライマリ) インスタンスのみが含まれている場合、リーダーみエンドポイントはプライマリインスタンスのみに接続します。Amazon DocumentDB クラスターにレプリカインスタンスを追加した場合、リーダーエンドポイントは、アクティブになった時点で新しいレプリカへの読み取り専用接続を開きます。

Amazon DocumentDB クラスターのリーダーエンドポイントの例を以下に示します。

```
sample-cluster.cluster-ro-123456789012.us-east-1.docdb.amazonaws.com:27017
```

次の接続文字列は、リーダーエンドポイントを使用した例です。

```
mongodb://username:password@sample-cluster.cluster-ro-123456789012.us-
east-1.docdb.amazonaws.com:27017
```

リーダーエンドポイントは、読み込みリクエストではなく、読み取り専用接続の負荷分散を行います。一部のリーダーエンドポイント接続が他のリーダーエンドポイント接続よりも多く使用されている場合、読み込みリクエストはクラスター内のインスタンス間で均等に分散されない可能性があります。クラスターエンドポイントにレプリカセットとして接続し、secondaryPreferred 読み込み設定オプションを使用して、リクエストを分散することをお勧めします。

クラスターのエンドポイントを見つける方法については、「クラスターのエンドポイントの検索」を参照してください。

インスタンスエンドポイント

クラスター内で特定のインスタンスに接続するインスタンスエンドポイントです。現在のプライマリインスタンスのインスタンスエンドポイントは、読み込み/書き込みオペレーションに使用できます。ただし、リードレプリカのインスタンスエンドポイントに書き込みオペレーションを実行しようとすると、エラーが発生します。Amazon DocumentDB クラスターは、アクティブなインスタンスごとに 1 つのインスタンスのエンドポイントを持ちます。

インスタンスエンドポイントは、クラスターエンドポイントやリーダーエンドポイントが適切でないシナリオ向けに、特定のインスタンスへの接続の直接制御を提供します。ユースケースの例として、定期的な読み取り専用分析ワークロード用のプロビジョニングがあります。通常よりも大きいレプリカインスタンスのプロビジョニング、インスタンスエンドポイントを使用した新しいより大きなインスタンスへの直接接続、分析クエリの実行を行った後で、インスタンスを終了できます。インスタンスエンドポイントを使用すると、分析トラフィックが他のクラスターインスタンスに影響を与えるのを防ぐことできます。

次の例では、Amazon DocumentDB クラスターの 1 つのインスタンスのインスタンスエンドポイントを示します。

```
sample-instance.123456789012.us-east-1.docdb.amazonaws.com:27017
```

次の接続文字列は、このインスタンスエンドポイントを使用した例です。

```
mongodb://username:password@sample-instance.123456789012.us-
east-1.docdb.amazonaws.com:27017
```

> **Note**
>
> プライマリまたはレプリカとしてのインスタンスのロールは、フェイルオーバーイベントが原因で変更される可能性があります。アプリケーションで、特定のインスタンスエンドポ

イントがプライマリインスタンスであることを前提にしないでください。本番稼働アプリケーションのインスタンスエンドポイントに接続することはお勧めしません。代わりに、クラスターエンドポイントを使用し、レプリカセットモード（「レプリカセットとしてのAmazon DocumentDB に接続する」を参照）でクラスターに接続することをお勧めします。インスタンスのフェイルオーバー優先度をより高度に制御する方法については、「Amazon DocumentDB クラスターの耐障害性について理解する」を参照してください。

クラスターのエンドポイントを見つける方法については、「インスタンスのエンドポイントの検索」を参照してください。

レプリカセットモード

レプリカセット名 `rs0` を指定して、レプリカセットモードで Amazon DocumentDB クラスターエンドポイントに接続できます。レプリカセットモードで接続すると、[読み取り保証]、[書き込み保証]、および [読み取り設定] のオプションを指定できます。詳細については、「読み込み整合性」を参照してください。

レプリカセットモードで接続する接続文字列の例を次に示します。

```
mongodb://username:password@sample-cluster.cluster-123456789012.us-
east-1.docdb.amazonaws.com:27017/?replicaSet=rs0
```

レプリカセットモードで接続するときは、Amazon DocumentDB クラスターはドライバーとクライアントに対してレプリカセットとして表示されます。Amazon DocumentDB クラスターに対して追加、削除したインスタンスは、レプリカセット設定に自動的に反映されます。

各 Amazon DocumentDB クラスターは、デフォルト名が `rs0` の 1 つのレプリカセットで構成されます。レプリカセットの名前を変更することはできません。

レプリカセットモードでのクラスターエンドポイントへの接続は、一般的な使用に推奨される方法です。

> **Note**
>
> Amazon DocumentDB クラスター内のすべてのインスタンスは、同じ TCP ポートで接続をリッスンします。

TLS サポート

TLS (Transport Layer Security) を使用した Amazon DocumentDB への接続方法については、「Encrypting data in transit」を参照してください。

Amazon DocumentDB のストレージ

Amazon DocumentDB のデータは クラスターボリューム に保存されます。これは、SSD (Solid State Drive) を使用する単一の仮想ボリュームです。クラスターボリュームはお客様のデータの 6 つのコピーで構成されます。これらのデータは、単一の AWS リージョンの複数のアベイラビリティーゾーン間で自動的にレプリケートされます。このレプリケーションにより、データの高い耐久性が保証され、データ損失の可能性が低くなります。また、データのコピーが他のアベイラビリティーゾーンにすでに存在するため、フェイルオーバー中のクラスターの可用性が高まります。これらのコピーは、継続して Amazon DocumentDB クラスター内のインスタンスに対するデータリクエストを処理できます。

データストレージに対する請求方法

Amazon DocumentDB は、データ量が増えるにつれて、クラスターボリュームのサイズを自動的に増やします。Amazon DocumentDB クラスターボリュームは最大サイズの 128 TiB まで拡張できますが、Amazon DocumentDBクラスターボリュームで使用する領域に対してのみ課金されます。Amazon DocumentDB 4.0 以降では、コレクションやインデックスの削除などでのデータを削除すると、割り当てられた領域全体が相応に減少します。したがって、不要になったコレクション、インデックス、データベースを削除することで、ストレージ料金を削減できます。Amazon DocumentDB バージョン 3.6 では、データを削除すると解放される領域をクラスターボリュームで再利用できますが、ボリューム自体のサイズが小さくなることはありません。バージョン 3.6 では、解放された領域が再利用されていても、コレクションまたはインデックスを削除してもストレージの変更が見られない場合があります。

> **Note**
>
> Amazon DocumentDB 3.6 では、ストレージコストは「高水準点 (high water mark)」 (任意の時点で Amazon DocumentDB クラスター用に割り当てられた最大量) に基づきます。大量の一時情報を作成したり、古い不要データの削除前に新規データを大量にロードする ETL プラクティスを避けることによってコストを管理できます。Amazon DocumentDB クラスターからデータを削除したことによって大量の割り当て済み未使用領域が発生した場合に、高水準点をリセットするには、`mongodump` や `mongorestore` などのツールを使用して論理データダンプを作成し、新しいクラスターに復元する必要があります。スナップショットを作成お

よび復元しても、割り当てられたストレージは削減されません。これは、基になるストレージの物理的なレイアウトが、復元されたスナップショットでも変更されないためです。

> **Note**
>
> `mongodump` や `mongorestore` などのユーティリティを使用すると、ストレージボリュームに対して読み書きするデータのサイズに応じた I/O 料金が発生します。

Amazon DocumentDB データストレージおよび I/O の料金に関する情報は、「Amazon DocumentDB (MongoDB 互換性) の料金] と [料金のよくある質問」を参照してください。

Amazon DocumentDB レプリケーション

Amazon DocumentDB クラスターでは、各レプリカインスタンスが独立したエンドポイントを公開します。これらのレプリカエンドポイントは、クラスターボリュームのデータへの読み取り専用アクセスを提供します。また、複数のレプリケートされたインスタンス経由で、データの読み取りワークロードをスケールできます。さらに、データ読み取りのパフォーマンスを向上させ、Amazon DocumentDB クラスター内のデータの可用性を高めるためにも有効です。Amazon DocumentDB レプリカはフェイルオーバーターゲットでもあり、Amazon DocumentDB クラスターのプライマリインスタンスに障害が発生した場合は迅速に昇格されます。

Amazon DocumentDB の信頼性

Amazon DocumentDB は、信頼性、耐久性、および耐障害性を持つように設計されています。(可用性を高めるため、Amazon DocumentDB クラスターを設定し、異なるアベイラビリティーゾーンに複数のレプリカインスタンスを存在させる必要があります。) Amazon DocumentDB には、信頼性の高いデータベースソリューションとなるいくつかの自動機能も含まれています。

ストレージの自動修復

Amazon DocumentDB では、データの複数のコピーを 3 つのアベイラビリティーゾーンに保持し、ストレージの障害によってデータが失われる可能性を最小限に抑えます。Amazon DocumentDB は、クラスターボリュームの障害を自動的に検出します。クラスターボリュームのセグメントで障害が発生すると、Amazon DocumentDB はすぐにそのセグメントを修復します。クラスターボリュームを構成する他のボリュームからのデータを使用して、修復されたセグメントのデータが最新であるようにします。その結果、Amazon DocumentDB ではデータ損失が回避され、インスタンス障害から回復するためのポイントインタイム復元を実行する必要性が低減します。

存続できるキャッシュのウォームアップ

Amazon DocumentDB はそのページキャッシュをデータベースとは別のプロセスで管理するため、ページキャッシュはデータベースとは無関係に存続できます。予期できないデータベース障害が発生した場合でも、ページキャッシュはメモリに保持されます。これにより、バッファプールはデータベースの再起動時に最新の状態にウォームアップされます。

クラッシュ回復

Amazon DocumentDB は、クラッシュからほぼ瞬時に回復し、アプリケーションデータを提供し続けるように設計されています。Amazon DocumentDB は、クラッシュ回復を並列スレッドで非同期に実行します。これにより、クラッシュのほぼ直後にデータベースを開き、使用できるようにします。

リソースガバナンス

Amazon DocumentDB は、ヘルスチェックなど、サービス内で重要なプロセスを実行するために必要なリソースを保護します。これを行うには、インスタンスのメモリ負荷が高い場合、Amazon DocumentDB はリクエストをスロットルします。その結果、一部のオペレーションは、メモリ負荷が低下するのを待つためにキューに入れられる場合があります。メモリ負荷が続くと、キューに入れられたオペレーションがタイムアウトすることがあります。次の CloudWatch メトリクスを使用して、メモリ不足によるサービススロットリングオペレーションを監視できます: `LowMemThrottleQueueDepth`、`LowMemThrottleMaxQueueDepth`、`LowMemNumOperationsThrottl` 詳細については、「CloudWatch によるAmazon DocumentDB のモニタリング」を参照してください。LowMem CloudWatch メトリクスの結果として、インスタンスに持続的なメモリ負荷が見られる場合は、インスタンスをスケールアップしてワークロードに追加のメモリを提供することをお勧めします。

読み込み設定のオプション

Amazon DocumentDB は、3 つのアベイラビリティーゾーンにわたってデータを 6 回レプリケートするクラウドネイティブな共有ストレージサービスを使用して、高レベルの耐久性を提供します。Amazon DocumentDB は、耐久性を実現するためにデータを複数のインスタンスにレプリケートすることに依存しません。クラスターのデータは、1 つのインスタンスが含まれているか、15 個のインスタンスが含まれているかにかかわらず、耐久性に優れています。

トピック

- 書き込みの耐久性
- 読み込みの分離

- 読み込み整合性
- 高可用性
- 読み取りスケーリング

書き込みの耐久性

Amazon DocumentDB は、独自の分散型の耐障害性がある自己修復機能を備えたストレージシステムを使用します。このシステムは、データの 6 つのコピー (V=6) を 3 つの AWS アベイラビリティーゾーンにレプリケートして、高可用性と耐久性を実現します。Amazon DocumentDB はデータを書き込む際、クライアントへの書き込みを確認する前に、すべての書き込みが大半のノードに永続的にレコードされることを確認します。3 ノードの MongoDB レプリカセットを実行している場合は、書き込み確認として `{w:3, j:true}` を使用すると、Amazon DocumentDB と比較したときに可能な限り最高の設定を得られます。

Amazon DocumentDB クラスターへの書き込みは、クラスターのライターインスタンスによって処理される必要があります。リーダーに書き込もうとすると、エラーが発生します。Amazon DocumentDB プライマリインスタンスからの確認済み書き込みは耐久性があり、ロールバックできません。Amazon DocumentDB はデフォルトで高い耐久性に優れており、耐久性に優れない書き込みオプションはサポートされていません。耐久性レベルを変更することはできません (これは、書き込みについてです)。Amazon DocumentDB は w=anything を無視し、事実上 w: 3 と j: true となります。この設定を減らすことはできません。

Amazon DocumentDB アーキテクチャではストレージとコンピューティングが分離されているため、単一インスタンスのクラスターは高い耐久性を持ちます。耐久性はストレージレイヤーで処理されます。その結果、1 つのインスタンスと 3 つのインスタンスを持つ Amazon DocumentDB クラスターは、同じレベルの耐久性を実現します。データの高い耐久性を保持しながら、具体的なユースケースに対してクラスターを設定できます。

Amazon DocumentDB クラスターへの書き込みは、1 つのドキュメント内でアトミックです。

Amazon DocumentDB は `wtimeout` オプションをサポートしておらず、値が指定された場合はエラーを返しません。プライマリ Amazon DocumentDB インスタンスへの書き込みは、無期限にブロックしないことが保証されます。

読み込みの分離

Amazon DocumentDB インスタンスからの読み取りでは、クエリが開始される前に、耐久性があるデータのみが返されます。読み込みで、クエリの実行が開始された後で変更されたデータが返されることはなく、どのような状況でもダーティーリードが可能になることもありません。

読み込み整合性

Amazon DocumentDB クラスターから読み取られたデータは耐久性があり、ロールバックされません。Amazon DocumentDB 読み取りの読み取り整合性は、リクエストまたは接続の読み取り設定を指定して変更できます。Amazon DocumentDB は、耐久性に優れない読み取りオプションをサポートしていません。

Amazon DocumentDB クラスターのプライマリインスタンスからの読み取りは、通常のオペレーション条件下では整合性が高く、書き込み後読み取りの整合性を実現します。書き込みとそれ以降の読み取りの間にフェイルオーバーイベントが発生した場合、システムは高い整合性を持たない読み取りを一時的に返すことがあります。リードレプリカからのすべての読み取りは、結果的に整合性を持ち、同じ順序でデータを返します。レプリカラグは多くの場合、100 ミリ秒未満です。

Amazon DocumentDB 読み取り設定

Amazon DocumentDB は、レプリカセットモードのクラスターエンドポイントからデータを読み取るときのみ、読み取り設定オプションをサポートします。読み取り設定オプションの設定により、MongoDB クライアントまたはドライバーが Amazon DocumentDB クラスターでインスタンスに読み取りリクエストをルーティングする方法に影響が生じます。特定のクエリ用に読み込み設定オプションを指定するか、MongoDB ドライバーの全般オプションとして指定できます。(読み込み設定オプションの設定方法については、クライアントまたはドライバーのドキュメントを参照してください。)

クライアントまたはドライバーがレプリカセットモードの Amazon DocumentDB クラスターエンドポイントに接続していない場合、読み取り設定の指定結果は定義されていません。

Amazon DocumentDB は、読み取り設定としての タグセット の設定はサポートしていません。

サポートされている読み込み設定オプション

- **`primary`** - `primary` 読み取り設定を指定すると、すべての読み取りがクラスターのプライマリインスタンスにルーティングされます。プライマリインスタンスが使用できない場合、読み込みオペレーションは失敗します。`primary` 読み込み設定により、書き込み後読み取りの整合性が得られ、高可用性および読み込みスケーリングよりも書き込み後読み取りの整合性を優先するユースケースに適しています。

  次の例では、`primary` 読み取り設定を指定します。

  ```
  db.example.find().readPref('primary')
  ```

- **primaryPreferred** - primaryPreferred 読み取り設定を指定すると、通常のオペレーションでは読み取りがプライマリインスタンスにルーティングされます。プライマリフェイルオーバーが発生した場合、クライアントはレプリカにリクエストをルーティングします。primaryPreferred 読み取り設定により、通常のオペレーションで書き込み後読み取り整合性と、フェイルオーバーイベント中の結果整合性のある読み込みが得られます。primaryPreferred 読み取り設定は、読み込みスケーリングよりも書き込み後読み取り整合性を優先するが、それでも高可用性を必要とするユースケースに適しています。

  次の例では、primaryPreferred 読み取り設定を指定します。

```
db.example.find().readPref('primaryPreferred')
```

- **secondary** - secondary 読み取り設定を指定すると、読み取りはレプリカのみにルーティングされ、プライマリインスタンスにルーティングされることはありません。クラスター内のレプリカインスタンスがない場合、読み込みリクエストは失敗します。secondary 読み取り設定では、結果整合性のある読み込みが得られ、高可用性および書き込み後の読み取り整合性よりもプライマリインスタンスの書き込みスループットを優先するユースケースに適しています。

  次の例では、secondary 読み取り設定を指定します。

```
db.example.find().readPref('secondary')
```

- **secondaryPreferred** - secondaryPreferred 読み取り設定を指定すると、1 つ以上のレプリカがアクティブになったときに、読み取りはリードレプリカにルーティングされます。クラスター内にアクティブなレプリカインスタンスがない場合、読み取りリクエストはプライマリインスタンスにルーティングされます。secondaryPreferred 読み取り設定では、読み取りがリードレプリカによって対応されたときに、結果整合性のある読み込みが得られます。これにより、読み取りがプライマリインスタンスによって処理されたときに (フェイルオーバーイベントがなければ)、書き込み後読み取り整合性が得られます。secondaryPreferred 読み取り設定は、書き込み後読み取り整合性よりも読み取りスケーリングや高可用性を優先するユースケースに適しています。

  次の例では、secondaryPreferred 読み取り設定を指定します。

```
db.example.find().readPref('secondaryPreferred')
```

- **`nearest`** - `nearest` 読み取り設定を指定すると、クライアントと Amazon DocumentDB クラスター内のすべてのインスタンス間で測定されたレイテンシーのみに基づいて、読み取りがルーティングされます。`nearest` 読み取り設定では、読み取りがリードレプリカによって対応されたときに、結果整合性のある読み込みが得られます。これにより、読み取りがプライマリインスタンスによって処理されたときに (フェイルオーバーイベントがなければ)、書き込み後読み取り整合性が得られます。`nearest` 読み取り設定は、書き込み後読み取り整合性と読み込みスケーリングよりも、可能な限り低い読み取りレイテンシーと高可用性の達成を優先するユースケースに適しています。

  次の例では、`nearest` 読み取り設定を指定します。

```
db.example.find().readPref('nearest')
```

高可用性

Amazon DocumentDB は、レプリカをプライマリインスタンスのフェイルオーバーターゲットとして使用して、可用性の高いクラスター設定をサポートします。プライマリインスタンスが失敗すると、Amazon DocumentDB レプリカが新しいプライマリとして昇格します。ここでは、プライマリインスタンスに対して行われた読み取り要求と書き込み要求が失敗して例外が発生するときに、短時間の中断が発生します。

Amazon DocumentDB クラスターにレプリカが含まれていない場合は、障害の発生時にプライマリインスタンスが再作成されます。ただし、Amazon DocumentDB レプリカの昇格は、プライマリインスタンスを再作成するよりもより高速です。したがって、1 つ以上の Amazon DocumentDB レプリカをフェイルオーバーターゲットとして作成することが推奨されます。

フェイルオーバーターゲットとして使用することを目的としたレプリカは、プライマリインスタンスと同じインスタンスのものである必要があります。これらは、プライマリとは別のアベイラビリティーゾーンでプロビジョニングする必要があります。フェイルオーバーターゲットとして設定されるレプリカを制御できます。高可用性のための Amazon DocumentDB の設定に関するベストプラクティスについては、「Amazon DocumentDB クラスターの耐障害性について理解する」を参照してください。

読み取りスケーリング

Amazon DocumentDB レプリカは、読み取りスケーリングに最適です。これらは、クラスターボリュームでの読み取りオペレーション専用です。つまり、レプリカは書き込みを処理しません。データのレプリケーションは、インスタンス間ではなくクラスターボリューム内で行われます。したがって、各レプリカのリソースはクエリの処理専用となり、データのレプリケートおよび書き込み用とはなりません。

アプリケーションがより多くの読み取り容量を必要とする場合、すぐにクラスターにレプリカを追加できます (通常は 10 分未満)。読み取り容量要件が減少した場合、不要なレプリカを削除できます。Amazon DocumentDB レプリカを使用すると、必要な読み取り容量のみのお支払いで済みます。

Amazon DocumentDB は、読み取り設定オプションの使用を通じて、クライアント側の読み取りスケーリングをサポートします。詳細については、「Amazon DocumentDB 読み取り設定」を参照してください。

TTL 削除

バックグラウンドプロセスによって達成された TTL インデックスエリアからの削除は、ベストエフォートであり、特定の期間内に保証されるものではありません。インスタンスサイズ、インスタンスリソースの使用率、ドキュメントサイズ、全体的なスループットなどの要因は、TTL 削除のタイミングに影響を与える可能性があります。

TTL モニタがドキュメントを削除すると、削除ごとに IO コストが発生するため、請求額が増加します。スループットと TTL 削除レートが増加する場合、IO 使用量の増加により、請求額が増加することが予想されます。

既存のコレクションに TTL インデックスを作成する場合、インデックスを作成する前に、期限切れのドキュメントをすべて削除する必要があります。現在の TTL 実装は、コレクション内のドキュメントのごく一部を削除するために最適化されています。これは、最初からコレクションでTTLが有効化されていた場合であり、一度に多数のドキュメントを削除する必要がある場合、必要以上に高い IOPS が発生する可能性があるからです。

TTL インデックスを作成してドキュメントを削除する代わりに、時間に基づいてドキュメントをコレクションにセグメント化し、ドキュメントが不要になったときにそれらのコレクションをドロップすることができます。例えば、週に 1 つのコレクションを作成し、IO コストをかけずにドロップできます。これは、TTL インデックスを使用するよりもコスト効率が大幅に向上します。

有料リソース

有料 Amazon DocumentDB リソースの特定

Amazon DocumentDB はフルマネージドデータベースサービスであるため、インスタンス、ストレージ、I/O、バックアップ、およびデータ転送が有料です。詳細については、「Amazon DocumentDB (MongoDB 互換) の料金」を参照してください。

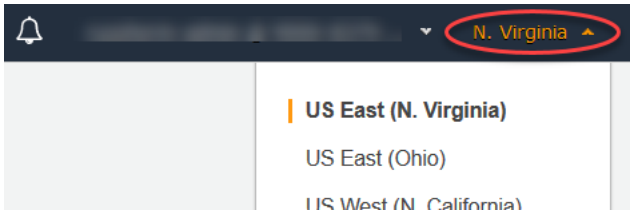
アカウント内の請求対象リソースを検出し、そのリソースを削除する場合は、 AWS Management Console または を使用できます AWS CLI。

の使用 AWS Management Console

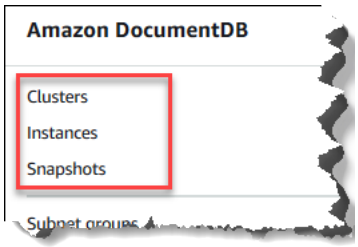
を使用すると AWS Management Console、特定の に対してプロビジョニングした Amazon DocumentDB クラスター、インスタンス、スナップショットを検出できます AWS リージョン。

クラスター、インスタンス、およびスナップショットを検出するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. デフォルトのリージョン以外のリージョンで請求対象リソースを検出するには、画面の右上隅で、検索 AWS リージョン する を選択します。



3. ナビゲーションペインで、該当する有料リソースのタイプ ([クラスター]、[インスタンス]、または [スナップショット]) を選択します。



4. そのリージョンについてプロビジョニングされたすべてのクラスター、インスタンス、またはスナップショットが右側のペインに表示されます。クラスター、インスタンス、およびスナップショットにたいして料金が発生します。

の使用 AWS CLI

を使用すると AWS CLI、特定の に対してプロビジョニングした Amazon DocumentDB クラスター、インスタンス、スナップショットを検出できます AWS リージョン。

クラスターおよびインスタンスを見つけるには

次のコードは、指定されたリージョンのすべてのクラスターとインスタンスをリスト表示しています。デフォルトのリージョンでクラスターとインスタンスを検索する場合は、`--region` パラメータを省略できます。

Example

Linux、macOS、Unix の場合:

```
aws docdb describe-db-clusters \
    --region us-east-1 \
    --query 'DBClusters[?Engine==`docdb`]' | \
       grep -e "DBClusterIdentifier" -e "DBInstanceIdentifier"
```

Windows の場合:

```
aws docdb describe-db-clusters ^
    --region us-east-1 ^
    --query 'DBClusters[?Engine==`docdb`]' | ^
       grep -e "DBClusterIdentifier" -e "DBInstanceIdentifier"
```

このオペレーションによる出力は、次のようになります。

```
"DBClusterIdentifier": "docdb-2019-01-09-23-55-38",
        "DBInstanceIdentifier": "docdb-2019-01-09-23-55-38",
        "DBInstanceIdentifier": "docdb-2019-01-09-23-55-382",
"DBClusterIdentifier": "sample-cluster",
"DBClusterIdentifier": "sample-cluster2",
```

スナップショットを見つけるには

次のコードは、指定されたリージョンのすべてのスナップショットをリスト表示しています。デフォルトのリージョンでスナップショットを検索する場合は、`--region` パラメータを省略できます。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-cluster-snapshots \
  --region us-east-1 \
  --query 'DBClusterSnapshots[?Engine==`docdb`].
[DBClusterSnapshotIdentifier,SnapshotType]'
```

Windows の場合:

```
aws docdb describe-db-cluster-snapshots ^
  --region us-east-1 ^
  --query 'DBClusterSnapshots[?Engine==`docdb`].
[DBClusterSnapshotIdentifier,SnapshotType]'
```

このオペレーションによる出力は、次のようになります。

```
[
    [
        "rds:docdb-2019-01-09-23-55-38-2019-02-13-00-06",
        "automated"
    ],
    [
        "test-snap",
        "manual"
    ]
]
```

削除する必要があるのは、manual スナップショットのみです。Automated スナップショットは、クラスターを削除すると削除されます。

不要な有料リソースの削除

クラスターを削除するには、まずクラスター内のインスタンスをすべて削除する必要があります。

- インスタンスを削除するには、「Amazon DocumentDB インスタンスの削除」を参照してください。

> **Important**
>
> クラスター内のインスタンスを削除しても、そのクラスターに関連付けられているストレージとバックアップの使用量に対して料金が発生します。すべての課金を停止するには、クラスターと手動スナップショットも削除する必要があります。

- クラスターを削除するには、「Amazon DocumentDB クラスターの削除」を参照してください。
- 手動スナップショットを削除するには、「クラススナップショットの削除」を参照してください。

ドキュメントデータベースとは

データモデルを正規化された行や列の観点から考えない開発者もいます。通常、アプリケーション層では、データは JSON ドキュメントとして表されます。これは、開発者にとって、データモデルをドキュメントとして考える方が直感的であるためです。

アプリケーションコードで使用するのと同じドキュメントモデル形式を使用してデータをデータベースに保持できるため、ドキュメントデータベースの人気が高まっています。ドキュメントデータベースは、柔軟でアジャイルな開発のための強力で直感的な API を提供します。

トピック

- ドキュメントデータベースのユースケース
- ドキュメントについて理解する
- ドキュメントでの作業

ドキュメントデータベースのユースケース

ユースケースにより、ドキュメントデータベースが必要か、またはデータを管理するためのその他の種類のデータベースが必要かが決まります。ドキュメントデータベースは、迅速で反復性のある開発のための柔軟なスキーマを必要とするワークロードに適しています。以下では、ドキュメントデータベースに大きな利点があるユースケースの例をいくつか示します。

トピック

- ユーザープロファイル
- リアルタイムのビッグデータ
- コンテンツ管理

ユーザープロファイル

ドキュメントデータベースには柔軟性が高いスキーマがあるため、属性とデータ値が異なるドキュメントを保存できます。ドキュメントデータベースは、さまざまなユーザーがさまざまな種類の情報を提供するオンラインプロファイルに対する実用的なソリューションです。ドキュメントデータベース

を使用して、各ユーザーに固有の属性のみを保存し、各ユーザーのプロファイルを効率的に保存することができます。

ユーザーが自分のプロファイル情報を追加または削除したとします。この場合、ドキュメントは、最近追加された属性とデータを含むか、新しく省略された属性とデータが省略された、更新されたバージョンに簡単に置き換えることができます。ドキュメントデータベースは、このようなレベルの個性や流動性を簡単に管理することができます。

リアルタイムのビッグデータ

従来、運用データから情報を抽出する能力は、運用データベースと分析データベースがそれぞれ異なる環境 (運用環境およびビジネス/レポート作成) で維持されているという事実によって妨げられていました。リアルタイムで動作情報を抽出できることは、競争の激しいビジネス環境において非常に重要です。ドキュメントデータベースを使用することで、企業は任意の送信元からの運用データを保存して管理し、同時に分析のため任意の BI エンジンにデータをフィードできます。2 つの環境を持つ必要はありません。

コンテンツ管理

コンテンツを効果的に管理するために、さまざまなソースからコンテンツを収集して集計し、お客様に提供する必要があります。ドキュメントデータベースは、柔軟性の高いスキーマがあるため、あらゆる種類のデータの収集および保存に最適です。それらを使用して、イメージ、コメント、ビデオなど、ユーザーが生成したコンテンツを含めて、新しい種類のコンテンツを作成し、組み込むことができます。

ドキュメントについて理解する

ドキュメントデータベースは、セミ構造化データをドキュメントとして格納するために使用されます。リレーショナルデータベースのように、それぞれが固有の固定構造を持つ複数のテーブルにわたってデータを正規化するのではなく、ドキュメントとして格納します。ドキュメントデータベースに保存されているドキュメントは、ネストされたキーと値のペアを使用して、ドキュメントの構造またはスキーマを提供します。ただし、さまざまな種類のドキュメントを同じドキュメントデータベースに保存できるため、形式が異なる類似したデータの処理要件を満たします。たとえば、各ドキュメントは自己記述型であるため、トピック「ドキュメントデータベースのドキュメントの例」で説明しているオンラインストアの JSON でエンコードされたドキュメントは、同じドキュメントデータベースに保存することができます。

トピック

- SQL 用語と非リレーショナル用語の比較

- シンプルなドキュメント
- 埋め込まれたドキュメント
- ドキュメントデータベースのドキュメントの例
- ドキュメントデータベースの正規化について

SQL 用語と非リレーショナル用語の比較

次の表は、SQL データベースによって使用される用語とドキュメントデータベース (MongoDB) で使用される用語を比較したものです。

SQL	MongoDB
[テーブル]	収集
行	ドキュメント
列	フィールド
プライマリキー	ObjectId
[Index] (インデックス)	[Index] (インデックス)
ビュー	ビュー
ネストされたテーブルまたはオブジェクト	埋め込まれたドキュメント
配列	配列

シンプルなドキュメント

ドキュメントデータベースのすべてのドキュメントは自己記述型です。このドキュメントでは、JSON に似た形式のドキュメントを使用しますが、エンコードの他の方法も使用できます。

単純ドキュメントには、ドキュメント内ですべてが同じレベルの 1 つ以上のフィールドがあります。次の例では、フィールド `SSN`、`LName`、`FName`、`DOB`、`Street`、`City`、`State-Province`、`PostalCode`、および `Country` はすべてドキュメント内の兄弟です。

```
{
```

```
   "SSN": "123-45-6789",
   "LName": "Rivera",
   "FName": "Martha",
   "DOB": "1992-11-16",
   "Street": "125 Main St.",
   "City": "Anytown",
   "State-Province": "WA",
   "PostalCode": "98117",
   "Country": "USA"
}
```

情報を単純ドキュメントに整理すると、各フィールドは個別に管理されます。ユーザーのアドレスを取得するには、Street、City、State-Province、PostalCode、および Country を個別のデータ項目として取得する必要があります。

埋め込まれたドキュメント

複合ドキュメントは、ドキュメント内に埋め込まれたドキュメントを作成してデータを整理します。埋め込まれたドキュメントでは、データをグループまたは個別のデータ項目として管理するために役立ちます。ケース別に、どちらか効率的なほうを選択できます。前の例を使用すると、メインドキュメントに Address ドキュメントを埋め込むことができます。この操作を行うと、次のドキュメント構造になります。

```
{
   "SSN": "123-45-6789",
   "LName": "Rivera",
   "FName": "Martha",
   "DOB": "1992-11-16",
   "Address":
   {
       "Street": "125 Main St.",
       "City": "Anytown",
       "State-Province": "WA",
       "PostalCode": "98117",
       "Country": "USA"
   }
}
```

ドキュメント内のデータは、個別のフィールド ("SSN":)、埋め込まれたドキュメント ("Address":)、または埋め込まれたドキュメントのメンバー ("Address":{"Street":}) としてアクセスできるようになりました。

ドキュメントデータベースのドキュメントの例

前に説明したように、ドキュメントデータベース内の各ドキュメントは自己記述型であるため、ドキュメントデータベース内のドキュメントの構造は、相互に異なる可能性があります。次の 2 つのドキュメント (1 つは書籍用、もう 1 つは定期刊行物用) は、構造的に異なります。ただし、両方とも同じドキュメントデータベースに存在できます。

書籍ドキュメントの例を次に示します。

```
{
    "_id" : "9876543210123",
    "Type": "book",
    "ISBN": "987-6-543-21012-3",
    "Author":
    {
        "LName":"Roe",
        "MI": "T",
        "FName": "Richard"
    },
    "Title": "Understanding Document Databases"
}
```

2 つの記事を含む定期刊行物の例を以下に示します。

```
{
    "_id" : "0123456789012",
    "Publication": "Programming Today",
    "Issue":
    {
        "Volume": "14",
        "Number": "09"
    },
    "Articles" : [
        {
            "Title": "Is a Document Database Your Best Solution?",
            "Author":
            {
                "LName": "Major",
                "FName": "Mary"
            }
        },
        {
```

```
            "Title": "Databases for Online Solutions",
            "Author":
            {
                "LName": "Stiles",
                "FName": "John"
            }
        }
    ],
    "Type": "periodical"
}
```

これら 2 つのドキュメントの構造を比較します。リレーショナルデータベースでは、別々の「定期刊行物」と「書籍」テーブル、または null の値として「刊行物」、「問題」、「記事」、「MI」など未使用のフィールドを持つ 1 つのテーブルが必要です。ドキュメントデータベースが半構造化され、各ドキュメントが独自の構造を定義しているため、それらの 2 つのドキュメントは null フィールドなしで同じドキュメントデータベースに共存できます。ドキュメントデータベースは、スパースデータの処理に適しています。

ドキュメントデータベースに対する開発により、迅速で反復的な開発が可能になります。これは、動的にドキュメントのデータ構造を変更でき、コレクション全体のスキーマを変更する必要がないためです。ドキュメントデータベースが適しているのは、アジャイル開発と動的に変化する環境です。

ドキュメントデータベースの正規化について

ドキュメントデータベースは正規化されていません。そのため、1 つのドキュメントで見つかったデータを別のドキュメントで繰り返すことができます。さらに、ドキュメント間で一部のデータに相違が発生する場合があります。たとえば、オンラインストアで購入を行い、購入のすべての詳細が 1 つのドキュメントに保存されるシナリオを考えてみます。このドキュメントは、たとえば次の JSON ドキュメントのようになります。

```
{
    "DateTime": "2018-08-15T12:13:10Z",
    "LName" : "Santos",
    "FName" : "Paul",
    "Cart" : [
        {
            "ItemId" : "9876543210123",
            "Description" : "Understanding Document Databases",
            "Price" : "29.95"
        },
        {
```

```
            "ItemId" : "0123456789012",
            "Description" : "Programming Today",
            "Issue": {
                "Volume": "14",
                "Number": "09"
            },
            "Price" : "8.95"
        },
        {
            "ItemId": "234567890-K",
            "Description": "Gel Pen (black)",
            "Price": "2.49"
        }
    ],
    "PaymentMethod" :
    {
        "Issuer" : "MasterCard",
        "Number" : "1234-5678-9012-3456"
    },
    "ShopperId" : "1234567890"
}
```

この情報は、トランザクションコレクションでドキュメントとして保存されます。後で、1 つのアイテムの購入を忘れていることに気付きます。そのため、再度同じストアにログオンし、別の購入を行います。これもトランザクションコレクションの別のドキュメントとして保存されます。

```
{
    "DateTime": "2018-08-15T14:49:00Z",
    "LName" : "Santos",
    "FName" : "Paul",
    "Cart" : [
        {
            "ItemId" : "2109876543210",
            "Description" : "Document Databases for Fun and Profit",
            "Price" : "45.95"
        }
    ],
    "PaymentMethod" :
    {
        "Issuer" : "Visa",
        "Number" : "0987-6543-2109-8765"
    },
    "ShopperId" : "1234567890"
```

```
}
```

この 2 つのドキュメント、つまり名前と買い物客 ID (同じクレジットカードを使用した場合はクレジットカード情報) の冗長性に注目してください。しかし、ストレージは安価であり、各ドキュメントは結合が必要ない単純なキーと値のクエリで迅速に取得できる単一のトランザクションを完全に記録するため、問題はありません。

また、2 つの文書の間には、クレジットカード情報という明らかな相違があります。これは明らかな相違であるのは、各購入で別のクレジットカードを使用した可能性が高いためです。各ドキュメントは、記録されるトランザクションについては正確です。

ドキュメントでの作業

ドキュメントデータベースとしての Amazon DocumentDB は、JSON データの保存、クエリ実行、インデックス作成を容易にします。Amazon DocumentDB において、コレクションはリレーショナルデータベースのテーブルに似ています。ただし、すべてのドキュメントに適用される単一のスキーマが存在しない点が異なります。コレクションでは、すべてのドキュメントを同じデータベースに保持しながら類似したドキュメントを一緒にグループ化でき、構造が類似している必要はありません。

前のセクションのドキュメント例を使用すると、`reading_material` および `office_supplies` のコレクションがあります。ソフトウェアは、ドキュメントが属するコレクションを適用する必要があります。

以下の例では、MongoDB API を使用して、ドキュメントを追加、クエリ、更新、および削除する方法を示します。

トピック

- ドキュメントの追加
- ドキュメントのクエリ
- ドキュメントの更新
- ドキュメントの削除

ドキュメントの追加

Amazon DocumentDB では、コレクションにドキュメントを最初に追加したときにデータベースが作成されます。次の例では、`example` という名前のコレクションを `test` データベースに作成しています。これは、クラスターに接続するときのデフォルトのデータベースです。最初のドキュメント

の挿入時に接続が暗黙で作成されるため、コレクション名のエラーチェックはありません。したがって、コレクション名にタイプミス (example の代わりに eexample など) があると、正しいコレクションの代わりに eexample コレクションが作成され、このコレクションにドキュメントが追加されます。エラーチェックはアプリケーションで処理する必要があります。

次の例では、MongoDB API を使用してドキュメントを追加します。

トピック

- 単一のドキュメントの追加
- 複数のドキュメントの追加

単一のドキュメントの追加

単一のドキュメントをコレクションに追加するには、コレクションに追加したドキュメントで `insertOne( {} )` オペレーションを使用します。

```
db.example.insertOne(
    {
        "Item": "Ruler",
        "Colors": ["Red","Green","Blue","Clear","Yellow"],
        "Inventory": {
            "OnHand": 47,
            "MinOnHand": 40
        },
        "UnitPrice": 0.89
    }
)
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "acknowledged" : true,
    "insertedId" : ObjectId("5bedafbcf65ff161707de24f")
}
```

複数のドキュメントの追加

複数のドキュメントをコレクションに追加するには、コレクションに追加したドキュメントのリストで `insertMany( [{},...,{}] )` オペレーションを使用します。この特定のリストのドキュメントのスキーマは異なりますが、すべてのドキュメントを同じコレクションに追加できます。

```
db.example.insertMany(
    [
        {
            "Item": "Pen",
            "Colors": ["Red","Green","Blue","Black"],
            "Inventory": {
                "OnHand": 244,
                "MinOnHand": 72
            }
        },
        {
            "Item": "Poster Paint",
            "Colors": ["Red","Green","Blue","Black","White"],
            "Inventory": {
                "OnHand": 47,
                "MinOnHand": 50
            }
        },
        {
            "Item": "Spray Paint",
            "Colors": ["Black","Red","Green","Blue"],
            "Inventory": {
                "OnHand": 47,
                "MinOnHand": 50,
                "OrderQnty": 36
            }
        }
    ]
)
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "acknowledged" : true,
    "insertedIds" : [
            ObjectId("5bedb07941ca8d9198f5934c"),
            ObjectId("5bedb07941ca8d9198f5934d"),
            ObjectId("5bedb07941ca8d9198f5934e")
    ]
}
```

ドキュメントのクエリ

ときどきオンラインストアのインベントリを検索し、販売商品を顧客が表示して購入できることを確認する必要があります。コレクションのクエリは、コレクションのすべてのドキュメントを対象とするか、特定の条件を満たすドキュメントのみを対象とするかにかかわらず、比較的簡単です。

ドキュメントのクエリを実行するには、find() オペレーションを使用します。この find() コマンドには、返すドキュメントを選択するために使用する条件を定義する単一のドキュメントパラメータがあります。find() からの出力は、改行のない 1 行のテキストとしてフォーマットされたドキュメントです。読みやすくするために出力ドキュメントをフォーマットするには、find().pretty() を使用します。このトピックのすべての例では、.pretty() を使用して出力を書式設定しています。

example と insertOne() の 2 つの演習で insertMany() コレクションに挿入した 4 つのドキュメントを使用してください。

トピック

- コレクション内のすべてのドキュメントの取得
- フィールドの値に一致するドキュメントの取得
- 埋め込みドキュメントに一致するドキュメントの取得
- 埋め込まれたドキュメントのフィールド値に一致するドキュメントの取得
- 配列に一致するドキュメントの取得
- 配列値に一致するドキュメントの取得
- 演算子を使用したドキュメントの取得

コレクション内のすべてのドキュメントの取得

コレクション内のすべてのドキュメントを取得するには、find() オペレーションで、空のクエリドキュメントを指定します。

以下のクエリは、example コレクション内のすべてのドキュメントを返します。

```
db.example.find( {} ).pretty()
```

フィールドの値に一致するドキュメントの取得

フィールドおよび値に一致するすべてのドキュメントを取得するには、find() オペレーションで、一致するフィールドおよび値を識別するクエリドキュメントを指定します。

上記のドキュメントを使用して、このクエリは、「Item」フィールドが「Pen」に等しいすべてのドキュメントを返します。

```
db.example.find( { "Item": "Pen" } ).pretty()
```

埋め込みドキュメントに一致するドキュメントの取得

埋め込みドキュメントに一致するすべてのドキュメントを検索するには、find() オペレーションで、埋め込みドキュメントの名前と、その埋め込みドキュメントのすべてのフィールドおよび値を識別するクエリドキュメントを指定します。

埋め込みドキュメントに一致するには、そのドキュメントの埋め込みドキュメントの名前が、クエリで指定した名前と同じであることが必要です。さらに、埋め込みドキュメントのフィールドおよび値も、クエリで指定したものと同じであることが必要です。

次のクエリでは、「Poster Paint」ドキュメントのみ返ります。これは、「Pen」の値は「OnHand」や「MinOnHand」とは異なり、「Spray Paint」には、クエリドキュメントよりもフィールドが 1 つ (OrderQnty) 多いためです。

```
db.example.find({"Inventory": {
    "OnHand": 47,
    "MinOnHand": 50 } } ).pretty()
```

埋め込まれたドキュメントのフィールド値に一致するドキュメントの取得

埋め込みドキュメントに一致するすべてのドキュメントを検索するには、find() オペレーションで、埋め込みドキュメントの名前と、その埋め込みドキュメントのすべてのフィールドおよび値を識別するクエリドキュメントを指定します。

上記のドキュメントでは、以下のようにクエリで「ドット表記」を使用して埋め込みドキュメントと目的のフィールドを指定しています。埋め込みドキュメント内に他のフィールドがある場合でも、これらに一致するすべてのドキュメントが返されます。クエリは「Poster Paint」と「Spray Paint」を返します。それらはいずれも、指定したフィールドおよび値と一致するためです。

```
db.example.find({"Inventory.OnHand": 47, "Inventory.MinOnHand": 50 }).pretty()
```

配列に一致するドキュメントの取得

配列に一致するすべてのドキュメントを検索するには、find() オペレーションで、目的の配列名とその配列内のすべての値を指定します。クエリは、配列値の名前および順序がクエリで指定したものと一致するすべてのドキュメントを返します。

以下のクエリは「Pen」のみを返します。「Poster Paint」では追加の色 (White) があり、「Spray Paint」では色の順序が異なるためです。

```
db.example.find( { "Colors": ["Red","Green","Blue","Black"] } ).pretty()
```

配列値に一致するドキュメントの取得

特定の属性配列を含むすべてのドキュメントを見つけるには、find() オペレーションで、目的の配列名と値を指定します。

```
db.example.find( { "Colors": "Red" } ).pretty()
```

上記のオペレーションは、すべてのドキュメントを返します。3 つのドキュメントのそれぞれで、Colors という配列があり、その配列のどこかに「Red」という値があるためです。値として「White」を指定すると、クエリからは「Poster Paint」のみが返されます。

演算子を使用したドキュメントの取得

以下のクエリは、「Inventory.OnHand」値が 50 未満のすべてのドキュメントを返します。

```
db.example.find(
        { "Inventory.OnHand": { $lt: 50 } } )
```

サポートされているクエリ演算子のリストについては、「クエリおよびプロジェクション演算子」を参照してください。

ドキュメントの更新

通常、ドキュメントは静的ではなく、アプリケーションワークフローの一部として更新されます。以下の例では、ドキュメントを更新する方法のいくつかを示します。

既存のドキュメントを更新するには、update() オペレーションを使用します。update() オペレーションには 2 つのドキュメントパラメータがあります。最初のドキュメントでは、更新するドキュメントを識別しています。2 番目のドキュメントでは、行う更新を指定しています。

既存のフィールド (シンプルなフィールド、配列、または埋め込みドキュメントは関係ありません) を更新する場合 フィールド名とその値を指定します。オペレーションの最後では、古いドキュメントのフィールドが新しいフィールドおよび値に置き換えられたようになります。

トピック

- 既存のフィールドの値の更新
- 新しいフィールドの追加
- 埋め込みドキュメントの置き換え
- 埋め込みドキュメントへの新しいフィールドの挿入
- ドキュメントからのフィールドの削除
- 複数のドキュメントからのフィールドの削除

既存のフィールドの値の更新

先ほど追加した次の 4 つのドキュメントを以下の更新オペレーションに使用します。

```
{
    "Item": "Ruler",
    "Colors": ["Red","Green","Blue","Clear","Yellow"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 40
    },
    "UnitPrice": 0.89
},
{
    "Item": "Pen",
    "Colors": ["Red","Green","Blue","Black"],
    "Inventory": {
        "OnHand": 244,
        "MinOnHand": 72
    }
},
{
    "Item": "Poster Paint",
    "Colors": ["Red","Green","Blue","Black","White"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 50
    }
```

```
},
{
    "Item": "Spray Paint",
    "Colors": ["Black","Red","Green","Blue"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 50,
        "OrderQnty": 36
    }
}
```

シンプルなフィールドを更新するには

シンプルなフィールドを更新するには、update() で $set を使用してフィールド名と新しい値を指定します。以下の例では、Item を「Pen」から「Gel Pen」に変更します。

```
db.example.update(
    { "Item" : "Pen" },
    { $set: { "Item": "Gel Pen" } }
)
```

このオペレーションの結果は、次のようになります。

```
{
    "Item": "Gel Pen",
    "Colors": ["Red","Green","Blue","Black"],
    "Inventory": {
        "OnHand": 244,
        "MinOnHand": 72
    }
}
```

配列を更新するには

以下の例では、既存の色の配列を、色のリストに Orange を含み White を含まない新しい配列に置き換えます。新しい色のリストは、update() オペレーションで指定された順序になります。

```
db.example.update(
    { "Item" : "Poster Paint" },
    { $set: { "Colors": ["Red","Green","Blue","Orange","Black"] } }
)
```

このオペレーションの結果は、次のようになります。

```
{
    "Item": "Poster Paint",
    "Colors": ["Red","Green","Blue","Orange","Black"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 50
    }
}
```

新しいフィールドの追加

1 つ以上の新しいフィールドを追加してドキュメントを変更するには、挿入先のドキュメントと、$set 演算子を使用して挿入する新しいフィールドと値を識別するクエリドキュメントで、update() オペレーションを使用します。

以下の例では、値 3.99 を含むフィールド UnitPrice を Spray Paints ドキュメントに追加します。値 3.99 は数字であり、文字列ではありません。

```
db.example.update(
    { "Item": "Spray Paint" },
    { $set: { "UnitPrice": 3.99 } }
)
```

このオペレーションの結果は、次の (JSON フォーマット) のようになります。

```
{
    "Item": "Spray Paint",
    "Colors": ["Black","Red","Green","Blue"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 50,
        "OrderQnty": 36
    },
    "UnitPrice": 3.99
}
```

埋め込みドキュメントの置き換え

埋め込みドキュメントを置き換えてドキュメントを変更するには、update() オペレーションで、埋め込みドキュメントと、$set 演算子を使用して挿入する新しいフィールドおよび値を識別するドキュメントを指定します。

以下のドキュメントがあるとします。

```
db.example.insert({
    "DocName": "Document 1",
    "Date": {
        "Year": 1987,
        "Month": 4,
        "Day": 18
    }
})
```

埋め込みドキュメントを置き換えるには

以下の例では、現在の Date ドキュメントを、フィールド Month と Day のみを含み Year を含まない新しいドキュメントに置き換えます。

```
db.example.update(
    { "DocName" : "Document 1" },
    { $set: { "Date": { "Month": 4, "Day": 18 } } }
)
```

このオペレーションの結果は、次のようになります。

```
{
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18
    }
}
```

埋め込みドキュメントへの新しいフィールドの挿入

埋め込みドキュメントにフィールドを追加するには

埋め込みドキュメントに 1 つ以上の新しいフィールドを追加してドキュメントを変更するには、update() オペレーションで、「ドット表記」を使用して、埋め込みドキュメントと、$set 演算子を使用して挿入する新しいフィールドおよび値を識別するドキュメントを指定します。

以下のドキュメントでは、コードに「ドット表記」を使用して Year フィールドと DoW フィールドを埋め込み Date ドキュメントに挿入し、Words フィールドを親ドキュメントに挿入しています。

```
{
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18
    }
}
```

```
db.example.update(
    { "DocName" : "Document 1" },
    { $set: { "Date.Year": 1987,
              "Date.DoW": "Saturday",
              "Words": 2482 } }
)
```

このオペレーションの結果は、次のようになります。

```
{
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    },
    "Words": 2482
}
```

ドキュメントからのフィールドの削除

ドキュメントからフィールドを削除することでドキュメントを変更するには、update() オペレーションで、フィールドを削除するドキュメントを識別するクエリドキュメントを指定し、$unset 演算子で、削除するフィールドを指定します。

以下の例では、上記のドキュメントから Words フィールドを削除します。

```
db.example.update(
    { "DocName" : "Document 1" },
    { $unset: { Words:1 } }
)
```

このオペレーションの結果は、次のようになります。

```
{
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    }
}
```

複数のドキュメントからのフィールドの削除

複数のドキュメントからフィールドを削除してドキュメントを変更するには、update( ) オペレーションを使用します ($unset 演算子を指定し、multi オプションを true に設定します)。

次の例では、コレクション例のすべてのドキュメントから Inventory フィールドを削除します。ドキュメントに Inventory フィールドがない場合、そのドキュメントに対して実行されるアクションはありません。multi: true が省略された場合、アクションは条件を満たす最初のドキュメントでのみ実行されます。

```
db.example.update(
    {},
    { $unset: { Inventory:1 } },
    { multi: true }
)
```

ドキュメントの削除

データベースからドキュメントを削除するには、remove( ) オペレーションを使用します。これにより、削除するドキュメントを指定します。以下のコードは example コレクションから「Gel Pen」を削除します。

```
db.example.remove( { "Item": "Gel Pen" } )
```

データベースからすべてのドキュメントを削除するには、空のクエリで remove()オペレーションを使用します。

```
db.example.remove( { } )
```

Amazon DocumentDB の使用開始

Amazon DocumentDB に接続して開始する方法は複数あります。このガイドは、ユーザーが当社の強力なドキュメントデータベースの使用を開始するための最も迅速、シンプルで簡単な方法です。このガイドではAWS CloudShell、 を使用して、 から直接 Amazon DocumentDB クラスターに接続してクエリを実行します AWS Management Console。 AWS 無料利用枠の対象となる新規のお客様は、Amazon DocumentDB と CloudShell を無料で使用できます。 AWS CloudShell 環境または Amazon DocumentDB クラスターが無料利用枠を超えるリソースを使用している場合は、それらのリソースの通常の AWS 料金が課金されます。このガイドでは、5 分以内に Amazon DocumentDB の使用を開始します。

> **Note**
>
> このガイドの手順は、Amazon DocumentDB と が AWS CloudShell 利用可能な Amazon DocumentDB インスタンスベースのクラスターを作成して接続するためのものです。
>
> - Amazon DocumentDB エラスティッククラスターを作成して接続する場合は、「Amazon DocumentDB エラスティッククラスターの開始方法」を参照してください。
> - AWS 中国リージョンにお住まいの場合は、「」を参照してくださいAmazon EC2 への自動接続。

トピック

- 前提条件
- ステップ 1: クラスターを作成する
- ステップ 2: クラスターに接続する
- ステップ 3: データを挿入してクエリする
- ステップ 4: 探索する

前提条件

最初の Amazon DocumentDB クラスターを作成する前に、以下の操作を行う必要があります。

Amazon Web Services (AWS) アカウントを作成する

Amazon DocumentDB を使用する前に、Amazon Web Services (AWS) アカウントを持っている必要があります。 AWS アカウントは無料です。使用しているサービスとリソースに対してのみ料金をお支払いいただきます。

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

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

   サインアップ手順の一環として、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。

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

必要な AWS Identity and Access Management (IAM) アクセス許可を設定します。

クラスター、インスタンス、クラスターパラメータグループなどの Amazon DocumentDB リソースを管理するには、 がリクエストの認証 AWS に使用できる認証情報が必要です。詳細については、「Amazon DocumentDB の Identity and Access Management」を参照してください。

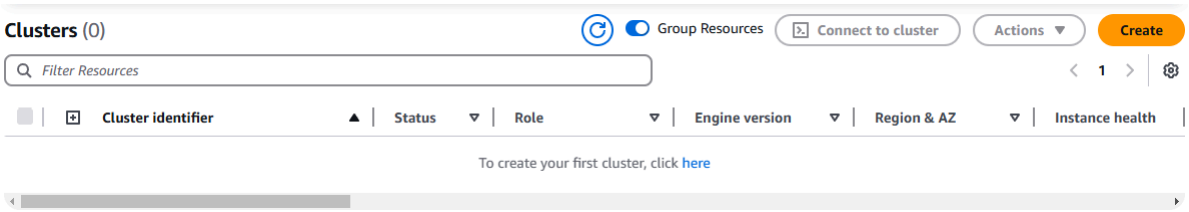
1. の検索バーで AWS Management Console、IAM と入力し、表示されるドロップダウンメニューで IAM を選択します。
2. IAM コンソールにアクセスしたら、ナビゲーションペインから [ユーザー] を選択します。
3. ユーザーネームを選択します。
4. [Add permissions] (アクセス許可の追加) をクリックします。
5. [ポリシーを直接アタッチする] を選択します。
6. 検索バーに `AmazonDocDBFullAccess` と入力し、検索結果に表示されたら、それを選択します。
7. [次へ] をクリックします。
8. [Add permissions] (アクセス許可の追加) をクリックします。

> **Note**
>
> AWS アカウントには、各リージョンにデフォルトの VPC が含まれます。Amazon VPC を使用する場合は、「Amazon VPC ユーザーガイド」の「Amazon VPC を作成する」トピックの手順を完了します。

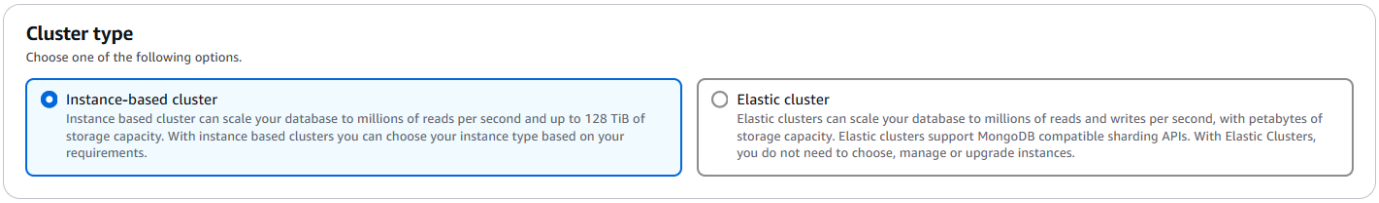
ステップ 1: クラスターを作成する

このステップでは、Amazon DocumentDB クラスターを作成します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. Amazon DocumentDB マネジメントコンソールで、[クラスター] の下にある [作成] を選択します。



3. Amazon DocumentDB クラスターの作成 ページのクラスタータイプ セクションで、インスタンスベースのクラスター (デフォルトオプション) を選択します。

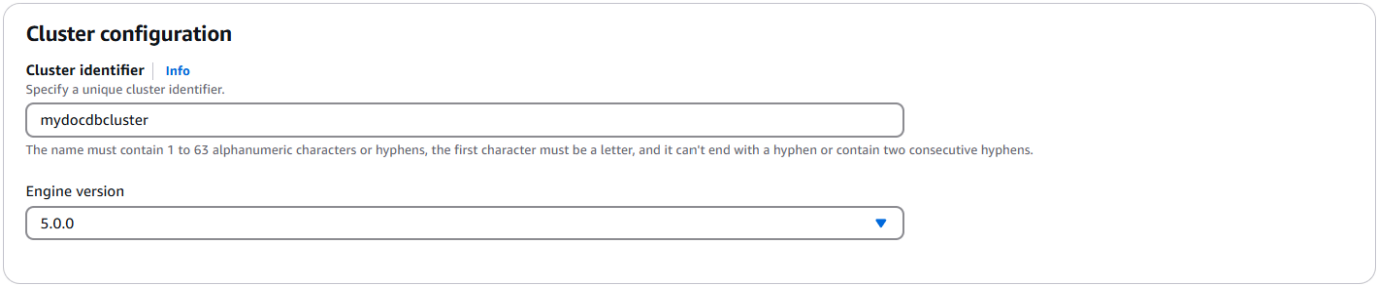


> **Note**
>
> このカテゴリのもう 1 つのオプションは Elastic クラスターです。Amazon DocumentDB エラスティッククラスターの詳細については、「」を参照してください。 Amazon DocumentDB のエラスティッククラスター

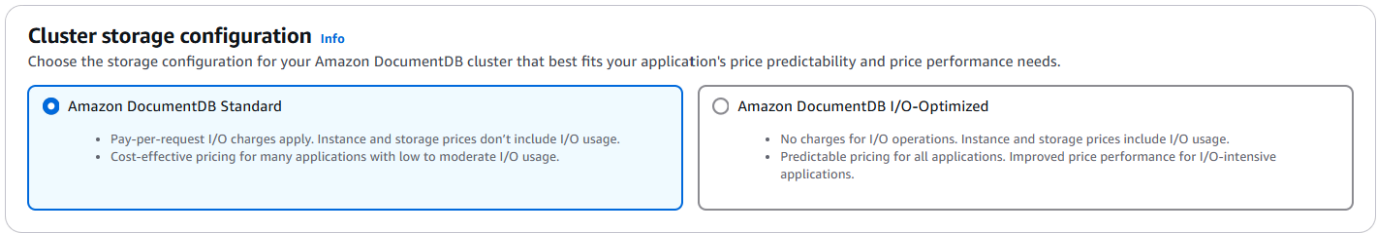
4. クラスター設定セクションで、次の操作を行います。

a. [クラスター識別子] には、**mydocdbcluster** などの一意の名称を入力します。コンソール操作では、入力方法に関係なくクラスター名のすべての文字が小文字に変換されることにご留意ください。

b. [エンジンバージョン] で、[5.0.0] を選択します。



5. [クラスターストレージ設定]セクションでは、[Amazon DocumentDB 標準] (デフォルトのオプションです) を選択します。



> **Note**
>
> このカテゴリのもう 1 つのオプションは、Amazon DocumentDB I/O 最適化です。いずれかのオプションの詳細については、「」を参照してください。 Amazon DocumentDB クラスターストレージ設定

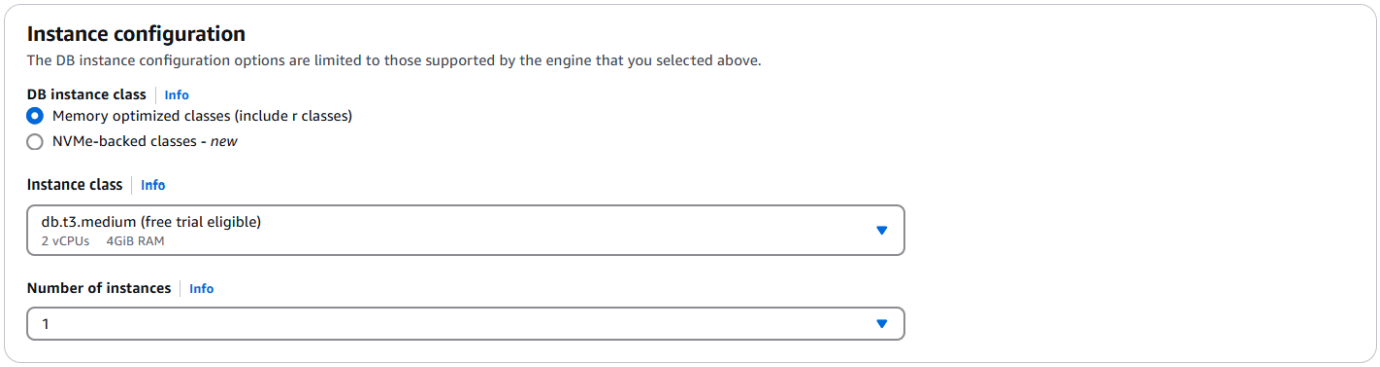
6. インスタンス設定 セクション:

a. DB インスタンスクラスの場合は、メモリ最適化クラス (r クラスを含む) を選択します (デフォルト) 。

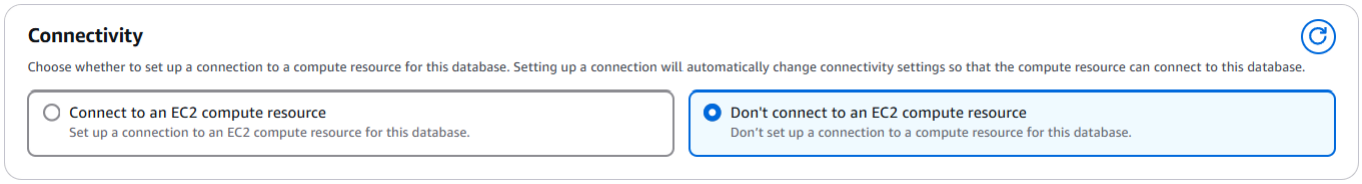
もう 1 つのインスタンスオプションは NVMe-backed クラスです。詳細については NVMe-backed インスタンスを参照してください。

b. [インスタンスクラス] で [t3.medium] を選択します。これは AWS 無料トライアルの対象となります。

c. [インスタンスの数] は、[1] を選択します。1 つのインスタンスを選択すると、コストを最小限に抑えることができます。これが本稼働システムの場合は、高可用性のために 3 つのインスタンスをプロビジョニングすることをお勧めします。

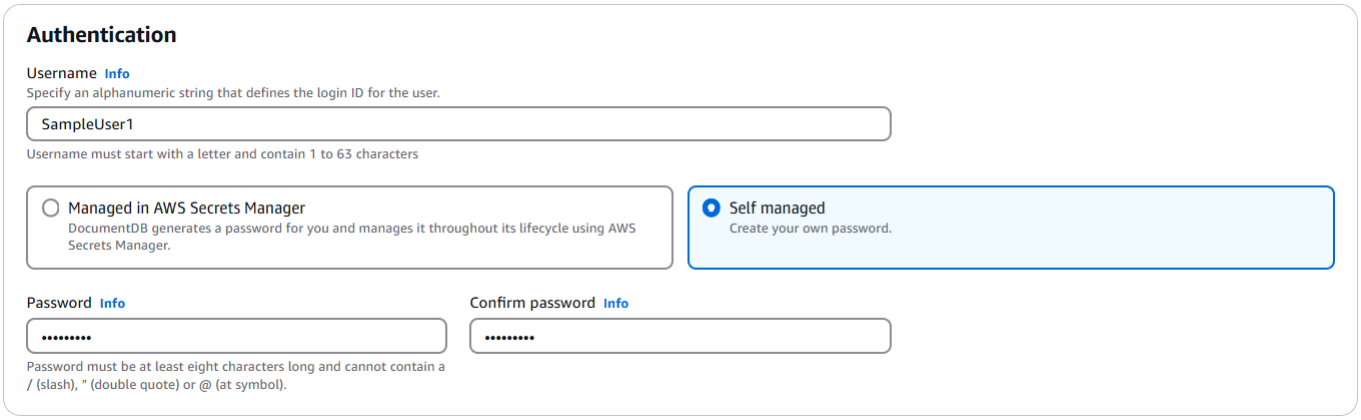


7. Connectivity セクションで、デフォルト設定のEC2 コンピューティングリソースに接続しない」のままにします。



8. 認証セクションで、プライマリユーザーのユーザー名を入力し、セルフマネージドを選択します。パスワードを入力して確認します。

   代わりに Managed in AWS Secrets Manager を選択した場合は、詳細については[Amazon DocumentDB と によるパスワード管理 AWS Secrets Manager]「」を参照してください。



9. 他のすべてのオプションはデフォルトのままにして、[クラスターの作成] を選択します。

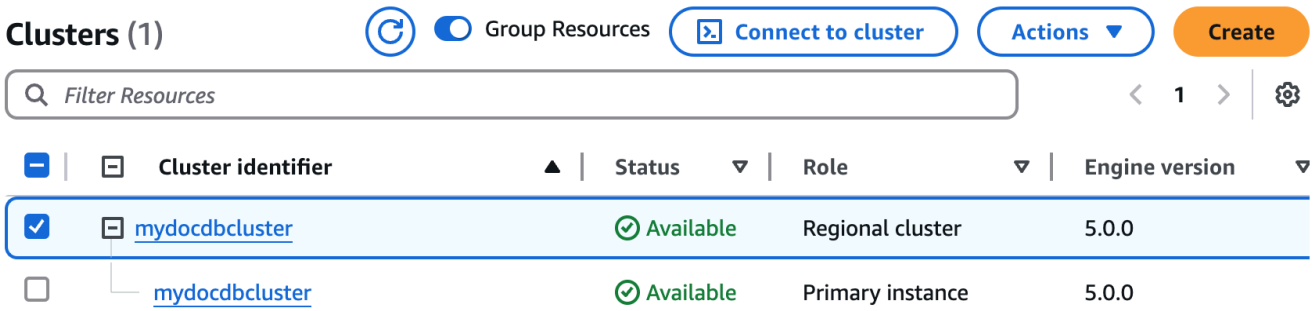
Amazon DocumentDB がクラスターをプロビジョニングしています。完了までに数分かかる場合もあります。

> **Note**
>
> クラスターのステータス値については、「Amazon DocumentDB のモニタリング」の章の「クラスターステータスの値」を参照してください。

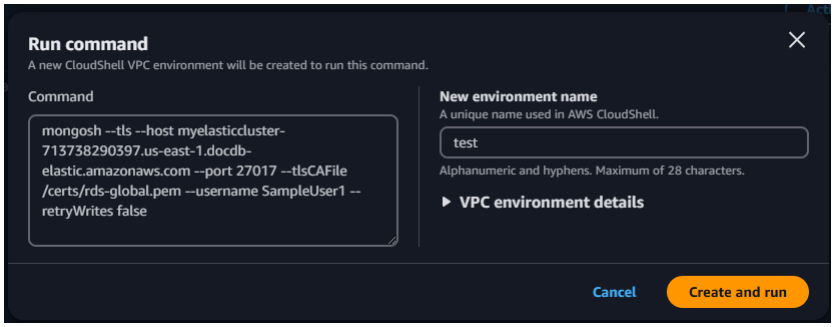
ステップ 2: クラスターに接続する

を使用して Amazon DocumentDB クラスターに接続します AWS CloudShell。

1. Amazon DocumentDB マネジメントコンソールの [クラスター] で、作成したクラスターを探します。クラスターの横にあるチェックボックスをクリックして選択します。



2. クラスターに接続する (アクションドロップダウンメニューの横にある) をクリックします。このボタンは、クラスターの横にあるチェックボックスをクリックした後にのみ有効になり、リージョンクラスターとプライマリインスタンス (複数可) の両方のステータスが使用可能と表示されます。CloudShell の [Run コマンド] 画面が表示されます。
3. New environment name フィールドに「test」などの一意の名前を入力し、Create and run をクリックします。VPC 環境の詳細は、Amazon DocumentDB データベースに自動的に設定されます。



4. プロンプトが表示されたら、ステップ 1: Amazon DocumentDB クラスターを作成する (サブステップ 7) で作成したパスワードを入力します。



パスワードを入力してプロンプトが になると`rs0 [direct: primary] <env-name>>`、Amazon DocumentDB クラスターに正常に接続されます。

> **Note**
>
> トラブルシューティングについては、「Amazon DocumentDB のトラブルシューティング」を参照してください。

ステップ 3: データを挿入してクエリする

クラスターに接続できたので、いくつかのクエリを実行して、ドキュメントデータベースの使用に慣れることができます。

1. 1 つのドキュメントを挿入するには、次のように入力します。

```
db.collection.insertOne({"hello":"DocumentDB"})
```

次の出力が返ります。

```
{
  acknowledged: true,
  insertedId: ObjectId('673657216bdf6258466b128c')
}
```

2. `findOne()` コマンドで書き込んだドキュメントを読み取ることができます (一つのドキュメントしか返さないため)。以下を入力します。

```
db.collection.findOne()
```

次の出力が返ります。

```
{ "_id" : ObjectId("5e401fe56056fda7321fbd67"), "hello" : "DocumentDB" }
```

3. さらにいくつかのクエリを実行するには、ゲームプロファイルのユースケースを検討してください。まず、「profiles」というタイトルのコレクションにエントリをいくつか挿入します。以下を入力します。

```
db.profiles.insertMany([{ _id: 1, name: 'Matt', status: 'active', level: 12, score:
 202 },
      { _id: 2, name: 'Frank', status: 'inactive', level: 2, score: 9 },
      { _id: 3, name: 'Karen', status: 'active', level: 7, score: 87 },
      { _id: 4, name: 'Katie', status: 'active', level: 3, score: 27 }
])
```

次の出力が返ります。

```
{ acknowledged: true, insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4 } }
```

4. find() コマンドを使用して、プロファイルコレクション内のすべてのドキュメントを返します。以下を入力します。

```
db.profiles.find()
```

ステップ 3 で入力したデータと一致する出力を取得します。

5. フィルターを使用して 1 つのドキュメントに対して1つのクエリを使用します。以下を入力します。

```
db.profiles.find({name: "Katie"})
```

次の出力が返ります。

```
{ "_id" : 4, "name" : "Katie", "status": "active", "level": 3, "score":27}
```

6. それでは、プロファイルを探して、それを findAndModify コマンドで変更してみましょう。次のコードを使用して、ユーザー Matt にさらに 10 ポイントを与えます。

```
db.profiles.findAndModify({
        query: { name: "Matt", status: "active"},
        update: { $inc: { score: 10 } }
    })
```

次のような出力が得られます (Mattのスコアはまだ上がっていないことに注意してください) 。

```
{
    [{_id : 1, name : 'Matt', status: 'active', level: 12, score: 202}]
```

7. 次のクエリを使用して、Matt のスコアが変更されたことを確認できます。

   ```
   db.profiles.find({name: "Matt"})
   ```

   次の出力が返ります。

   ```
   { "_id" : 1, "name" : "Matt", "status" : "active", "level" : 12, "score" : 212 }
   ```

ステップ 4: 探索する

お疲れ様でした。Amazon DocumentDB インスタンスベースのクラスターの開始方法ガイドが正常に完了しました。

次のステップ このデータベースを一般的な機能の一部を使用して、完全に活用する方法を学びましょう。

- Amazon DocumentDB イベントの管理
- スケーリング
- バックアップと復元

> **Note**
>
> この開始方法の演習で作成したクラスターは、削除しない限り、引き続きコストが発生します。手順については、以下を参照してください。Amazon DocumentDB クラスターを削除する。

を使用した Amazon DocumentDB クイックスタート AWS CloudFormation

このセクションには、 AWS CloudFormation を使用して Amazon DocumentDB (MongoDB 互換) をすぐに使い始めるための手順やその他の情報が含まれています。Amazon DocumentDB について一般的な情報は、「Amazon DocumentDB (MongoDB 互換) とは」を参照してください。

これらの手順では、 AWS CloudFormation テンプレートを使用して、デフォルトの Amazon VPC にクラスターとインスタンスを作成します。これらのリソースを自分で作成する方法については、「Amazon DocumentDB の使用開始」を参照してください。

> **Important**
>
> このテンプレートによって作成される AWS CloudFormation スタックは、Amazon DocumentDB のリソース (クラスターやインスタンスなど) や Amazon Elastic Compute Cloud (サブネットグループなど) など、複数のリソースを作成します。
> これらのリソースの一部は無料利用枠のリソースではありません。料金情報については、「Amazon DocumentDB の料金」と「Amazon EC2 の料金」を参照してください。スタックを終了したら削除して、以降そのスタックの料金が発生しないようにできます。

この AWS CloudFormation スタックはチュートリアルのみを目的としています。このテンプレートを本番環境で使用する場合は、より厳格な IAM ポリシーとセキュリティを使用することをお勧めします。リソースのセキュリティについては「Amazon VPC セキュリティ」および「Amazon EC2 ネットワークとセキュリティ」を参照してください。

トピック

- 前提条件
- Amazon DocumentDB AWS CloudFormation スタックを起動する
- Amazon DocumentDB クラスターにアクセスする
- 終了保護と削除保護

前提条件

Amazon DocumentDB クラスターを作成する前に、以下の操作を行う必要があります。

- デフォルトのAmazon VPC
- 必須の IAM アクセス権限

必要な IAM アクセス許可

以下のアクセス権限を使用すると、 AWS CloudFormation スタック用のリソースを作成できます。

AWS 管理ポリシー

- `AWSCloudFormationReadOnlyAccess`
- `AmazonDocDBFullAccess`

追加の IAM アクセス権限

次のポリシーは、この AWS CloudFormation スタックを作成および削除するために必要な追加のアクセス許可の概要を示しています。

次の例では、各`###############`をリソースの情報に置き換えます。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DocDBPermissions",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBCluster",
                "rds:DeleteDBCluster",
                "rds:ModifyDBCluster",
                "rds:DescribeDBClusters",
                "rds:CreateDBInstance",
                "rds:DeleteDBInstance",
                "rds:ModifyDBInstance",
                "rds:DescribeDBInstances",
                "rds:CreateDBSubnetGroup",
                "rds:DeleteDBSecurityGroup",
                "rds:DescribeDBSubnetGroups"
            ],
```

```
            "Resource": [
                "arn:aws:rds:{AWS_REGION}:{AWS_ACCOUNT_ID}:cluster:*",
                "arn:aws:rds:{AWS_REGION}:{AWS_ACCOUNT_ID}:db:*",
                "arn:aws:rds:{AWS_REGION}:{AWS_ACCOUNT_ID}:subgrp:*"
            ]
        },
        {
            "Sid": "EC2NetworkingPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateSecurityGroup",
                "ec2:DeleteSecurityGroup",
                "ec2:DescribeSecurityGroups",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets"
            ],
            "Resource": [
                "arn:aws:ec2:{AWS_REGION}:{AWS_ACCOUNT_ID}:security-group/*",
                "arn:aws:ec2:{AWS_REGION}:{AWS_ACCOUNT_ID}:vpc/*",
                "arn:aws:ec2:{AWS_REGION}:{AWS_ACCOUNT_ID}:subnet/*"
            ]
        },
        {
            "Sid": "EC2DescribePermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAvailabilityZones"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchLogsPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:{AWS_ACCOUNT_ID}:log-group:/aws/docdb/*",
```

```
                "arn:aws:logs:us-east-1:{AWS_ACCOUNT_ID}:log-group:/aws/docdb/
*:log-stream:*"
            ]
        },
        {
            "Sid": "KMSPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:CreateKey",
                "kms:DescribeKey",
                "kms:EnableKey",
                "kms:ListKeys",
                "kms:PutKeyPolicy"
            ],
            "Resource": "arn:aws:kms:us-east-1:{AWS_ACCOUNT_ID}:key/*"
        },
        {
            "Sid": "KMSEncryption",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:{AWS_ACCOUNT_ID}:key/*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "rds.us-east-1.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "IAMServiceLinkedRole",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
rds.amazonaws.com/AWSServiceRoleForRDS",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "rds.amazonaws.com"
                }
            }
```

```
        }
    ]
}
```

Amazon EC2 キーペア

AWS CloudFormation スタックを作成するリージョンで使用可能なキーペア (および PEM ファイル) が必要です。キーペアを作成する必要がある場合は、「Amazon EC2 ユーザーガイド」の「Amazon EC2 を使用したキーペアの作成」を参照してください。

Amazon DocumentDB AWS CloudFormation スタックを起動する

このセクションでは、Amazon DocumentDB AWS CloudFormation スタックを起動して設定する方法について説明します。

1. AWS Management Console で にサインインしますhttps://console.aws.amazon.com/。
2. 次の表に、各 AWS リージョン Amazon DocumentDB スタックテンプレートを示します。スタックを起動する の起動スタックを選択します。 AWS リージョン

リージョン	テンプレートを表示	デザイナーで表示	起動する
米国東部 (オハイオ)	テンプレートを表示	デザイナーで表示	Launch Stack
米国東部 (バージニア北部)	テンプレートを表示	デザイナーで表示	Launch Stack
米国西部 (オレゴン)	テンプレートを表示	デザイナーで表示	Launch Stack
アジアパシフィック (ムンバイ)	テンプレートを表示	デザイナーで表示	Launch Stack
アジアパシフィック (ソウル)	テンプレートを表示	デザイナーで表示	Launch Stack
アジアパシフィック (シンガポール)	テンプレートを表示	デザイナーで表示	Launch Stack

リージョン	テンプレートを表示	デザイナーで表示	起動する
アジアパシフィック (シドニー)	テンプレートを表示	デザイナーで表示	Launch Stack
アジアパシフィック (東京)	テンプレートを表示	デザイナーで表示	Launch Stack
カナダ (中部)	テンプレートを表示	デザイナーで表示	Launch Stack
欧州 (フランクフルト)	テンプレートを表示	デザイナーで表示	Launch Stack
欧州 (アイルランド)	テンプレートを表示	デザイナーで表示	Launch Stack
欧州 (ロンドン)	テンプレートを表示	デザイナーで表示	Launch Stack
欧州 (パリ)	テンプレートを表示	デザイナーで表示	Launch Stack

3. [スタックの作成] - 選択した Amazon DocumentDB テンプレートについて説明します。すべてのスタックは、スタックに含める AWS リソースに関する設定を含む JSON ファイルまたは YAML ファイルであるテンプレートに基づいています。上記のテンプレートからスタックを起動することを選択した場合、テンプレートは AWS リージョン 選択した の Amazon DocumentDB スタックを作成するように既に設定されています。

   AWS CloudFormation スタックを起動すると、Amazon DocumentDB クラスターの削除保護はデフォルトで無効になります。クラスターの削除保護を有効にする場合は、次のステップを実行します。それ以外の場合は、[次へ] を選択して次のステップに進みます。

   Amazon DocumentDB クラスターの削除保護を有効にするには

   1. スタック作成 ページの右下隅から、デザイナーで表示 を選択します。
   2. コンソールの結果の AWS CloudFormation デザイナーページで、統合された JSON および YAML エディタを使用してテンプレートを変更します。`Resources` セクションまでスクロールし、次のように変更して `DeletionProtection` を含めます。 AWS CloudFormation デザイナーの使用の詳細については、AWS CloudFormation 「デザイナーとは」を参照してください。

JSON:

```
"Resources": {
    "DBCluster": {
        "Type": "AWS::DocDB::DBCluster",
        "DeletionPolicy": "Delete",
        "Properties": {
            "DBClusterIdentifier": {
                "Ref": "DBClusterName"
            },
            "MasterUsername": {
                "Ref": "MasterUser"
            },
            "MasterUserPassword": {
                "Ref": "MasterPassword"
            },
            "DeletionProtection": "true"
        }
    },
```

YAML:

```
Resources:
  DBCluster:
    Type: 'AWS::DocDB::DBCluster'
    DeletionPolicy: Delete
    Properties:
      DBClusterIdentifier: !Ref DBClusterName
      MasterUsername: !Ref MasterUser
      MasterUserPassword: !Ref MasterPassword
      DeletionProtection: 'true'
```

3. ページの左上隅にある [スタックの作成]
   ()
   を選択して変更を保存し、これらの変更を有効にしてスタックを作成します。
4. 変更を保存すると、[スタックの作成] ページにリダイレクトされます。
5. [次へ] を選択して続行します。

4. [スタックの詳細をの指定] - テンプレートのスタック名とパラメータを入力します。パラメータはテンプレートで定義されます。パラメータを使用すると、スタックの作成時または更新時にカスタム値を入力できます。

- [スタック名] に、スタックの名前を入力するか、指定された名前を受け入れます。スタック名には、文字 (A ~ Z および a ~ z)、数字 (0 ~ 9)、およびダッシュ (-) を使用できます。
- [パラメータ] に、次の詳細を入力します。
  - DBClusterName - Amazon DocumentDB クラスターの名前を入力するか、表示された名前を受け入れます。

    クラスターの名前付けの制約:
    - 長さは [1 ～ 63] 個の英字、数字、またはハイフンです。
    - 1 字目は英字である必要があります。
    - ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
    - Amazon RDS、Neptune、Amazon DocumentDB のすべてのクラスターで AWS アカウント、リージョンごとに一意である必要があります。
  - DBInstanceClass - ドロップダウンリストから、Amazon DocumentDB クラスターのインスタンスクラスを選択します。
  - DBInstanceName - Amazon DocumentDB インスタンスの名前を入力するか、提供された名前を受け入れます。

    インスタンスの名前付けの制約:
    - 長さは [1 ～ 63] 個の英字、数字、またはハイフンです。
    - 1 字目は英字である必要があります。
    - ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
    - Amazon RDS、Neptune、Amazon DocumentDB のすべてのインスタンスで AWS アカウント、リージョンごとに一意である必要があります。
  - マスターパスワード - データベース管理者アカウントのパスワード。
  - マスターユーザー - データベース管理者アカウントのユーザー名。MasterUser は英数字で始まる必要があり、英数字のみ使用できます。

  [次へ] を選択して変更を保存し、続行します。

5. [スタックオプションの設定] - スタックのタグ、アクセス権限、および追加オプションを設定します。
   - [タグ] - スタック内のリソースに適用するタグ (キーと値のペア) を指定します。スタックごとに最大 50 個の一意のタグを追加できます。

- [アクセス権限] - オプションです。IAM ロールを選択して、 AWS CloudFormation がスタック内のリソースを作成、変更、または削除する方法を明示的に定義します。ロールを選択しない場合、 はユーザーの認証情報に基づいてアクセス許可 AWS CloudFormation を使用します。サービスロールを指定する前に、サービスロールを割り当てるアクセス権限 (`iam:PassRole`) があるかどうかを確認します。`iam:PassRole` アクセス権限は、どのロールを使用できるかを指定します。

  > **Note**
  >
  > サービスロールを指定すると、 は常にそのロールをそのスタックで実行されるすべてのオペレーション AWS CloudFormation に使用します。このスタックでオペレーションを実行するアクセス権限を持つ他のユーザーは、割り当てるアクセス権限がない場合でも、このロールを使用できます。ユーザーが持つべきではないアクセス権限がロールに含まれる場合、ユーザーのアクセス権限を非意図的にエスカレーションできてしまいます。ロールが最小権限を付与していることを確認します。

- [高度なオプション] - 以下の高度なオプションを設定することができます。

  - [スタックポリシー] - オプションです。スタックの更新中、意図しない更新から保護するリソースを定義します。デフォルトでは、スタックの更新時にすべてのリソースを更新できます。

    スタックポリシーを JSON として直接入力するか、スタックポリシーを含む JSON ファイルをアップロードすることができます。詳細については、「スタックのリソースが更新されないようにする」を参照してください。

  - [ロールバック構成] - オプションです。スタックの作成および更新時に がモニタリング AWS CloudFormation する CloudWatch Logs アラームを指定します。オペレーションがアラームしきい値を超えた場合、 はアラームしきい値を AWS CloudFormation ロールバックします。

  - [通知オプション] - オプションです。Simple Notification System (SNS) のトピックを指定します。

  - [スタック作成オプション] - オプションです。以下のオプションを指定できます。

    - [失敗時のロールバック] - スタックの作成に失敗した場合、スタックをロールバックするかどうかを指定します。

    - [タイムアウト] - スタック作成がタイムアウトするまでの時間 (分)です。

    - [終了保護] - スタックが誤って削除されるのを防ぎます。

> **Note**
>
> AWS CloudFormation 終了保護は、Amazon DocumentDB の削除保護の概念とは異なります。詳細については、「終了保護と削除保護」を参照してください。

[次へ] を選択して続行します。

6. [<stack-name> の確認] - スタックテンプレート、詳細、設定オプションを確認します。ページの下部にある quick-create link を開いて、これと同じ基本設定のスタックを作成することもできます。

   - [作成] を選択してスタックを作成します。
   - または、[変更セットの作成] を選択することもできます。変更セットは、スタックを作成する前に、このスタックがどのように設定されるかをプレビューするものです。これにより、変更セットを実行する前にさまざまな設定を調べることができます。

Amazon DocumentDB クラスターにアクセスする

AWS CloudFormation スタックが完了したら、Amazon EC2 インスタンスを使用して Amazon DocumentDB クラスターに接続できます。SSH を使用して Amazon EC2 インスタンスに接続する方法については、「Amazon EC2 ユーザーガイド」の「Linux インスタンスへの接続」を参照してください。

接続後、Amazon DocumentDB の使用に関する情報を含む、次のセクションを参照してください。

- ステップ 3: データを挿入してクエリする
- Amazon DocumentDB リソースの管理
- Amazon DocumentDB のモニタリング

終了保護と削除保護

Amazon DocumentDB のベストプラクティスとして、削除保護と終了保護を有効にすることをお勧めします。CloudFormation の終了保護は、Amazon DocumentDB の削除保護機能とは明確に異なる機能です。

- 終了保護 - CloudFormation スタックの終了保護を有効にすることによって、スタックが誤って削除されるのを防ぐことができます。終了保護を有効にした状態でスタックを削除しようとすると、削除は失敗し、スタックは変更されません。CloudFormation を使用してスタックを作成すると、終了保護はデフォルトで無効になります。終了保護は、スタック作成時に有効にすることができます。詳細については、AWS CloudFormation 「スタックオプションの設定」を参照してください。
- 削除保護 - Amazon DocumentDB は、クラスターに対して削除保護を有効にする機能も提供します。削除保護が有効になっている Amazon DocumentDB クラスターを削除しようとすると、削除は失敗し、クラスターは変更されません。削除保護を有効にする AWS CLIと、Amazon DocumentDB AWS Management Console、、および CloudFormation からの偶発的な削除から保護されます。Amazon DocumentDB クラスターの削除保護の有効化、および無効化の詳細については 削除保護 を参照してください。

Amazon DocumentDB の MongoDB との互換性

Amazon DocumentDB は、MongoDB 4.0 や MongoDB 5.0 などの MongoDB 互換性をサポートしています。MongoDB 互換性とは、現在お使いの MongoDB データベースで使用しているアプリケーション、ドライバ、ツールの大部分を Amazon DocumentDB でもほとんど変更することなく使用できることを意味します。ここでは、Amazon DocumentDB の新機能、利用開始、移行パス、機能的な違いなど、Amazon DocumentDB の MongoDB との互換性について知っておくべきことをすべて説明します。

トピック

- MongoDB 5.0 との互換性
- MongoDB 4.0 の互換

MongoDB 5.0 との互換性

トピック

- Amazon DocumentDB 5.0 の新機能
- Amazon DocumentDB 5.0 の使用開始
- Amazon DocumentDB 5.0 へのアップグレードまたは移行
- 機能的な違い

Amazon DocumentDB 5.0 の新機能

Amazon DocumentDB 5.0 では、ストレージ制限やクライアント側のフィールドレベルの暗号化など、新機能が導入されています。Amazon DocumentDB 5.0 で導入された主要な機能をいくつか紹介します。新機能の全リストを見るには、「リリースノート」 を参照してください。

- すべてのインスタンスベースの Amazon DocumentDB クラスターとシャードベースのエラスティッククラスターのストレージ制限を 128 TiB に増大。
- Amazon DocumentDB 5.0 (エンジンバージョン 3.0.775) の導入
  - MongoDB 5.0 API ドライバーのサポート
  - クライアント側のフィールドレベル暗号化 (FLE) のサポート。Amazon DocumentDB クラスターにデータを書き込む前に、クライアント側でフィールドを暗号化できるようになりました。詳細については、「クライアント側のフィールドレベルの暗号化」を参照してください。

- 新しい集約演算子: `$dateAdd`、`$dateSubtract`
- `$elemMatch` 演算子付きのインデックスをサポートします。その結果、`$elemMatch` を持つクエリではインデックススキャンが行われます。

Amazon DocumentDB は、すべての MongoDB 5.0 機能をサポートしているわけではありません。Amazon DocumentDB 5.0 を構築する際、お客様から最も多く構築するよう求められた特徴と機能から逆算していきました。今後もお客様の要望をもとに、MongoDB 5.0 の機能を追加していく予定です。サポートされている API の最新リストについては、「Amazon DocumentDB でサポートされている MongoDB API、オペレーション、およびデータ型」 を参照してください。

Amazon DocumentDB 5.0 の使用開始

Amazon DocumentDB 5.0 の使用を開始するには、入門ガイド を参照してください。新しい Amazon DocumentDB 5.0 クラスターは、 AWS Management Console または AWS SDK AWS CLI、または を使用して作成できます AWS CloudFormation。Amazon DocumentDB に接続する場合、MongoDB 5.0 以降と互換性のある MongoDB ドライバまたはユーティリティを使用する必要があります。

> **Note**
>
> AWS SDK AWS CLI、、または を使用する場合 AWS CloudFormation、エンジンバージョンはデフォルトで 5.0.0 になります。Amazon DocumentDB 4.0 クラスターを新規に作成する場合はパラメータ `engineVersion = 4.0.0` を、あるいは Amazon DocumentDB 3.6 クラスターを新規に作成する場合は`engineVersion = 3.6.0` を明示的に指定する必要があります。特定の Amazon DocumentDB クラスターでは、 AWS CLI を使用してクラスターのバージョンを確認する`describe-db-clusters`か、Amazon DocumentDB マネジメントコンソールを使用して特定のクラスターのエンジンバージョン番号を表示できます。

Amazon DocumentDB 5.0 は クラスターの `r6g` と `t4.medium` のインスタンスタイプなど、Amazon EC2 Graviton2 プロセッサーをサポートし、サポートされているすべてのリージョンで利用できます。料金の詳細については、「Amazon DocumentDB (MongoDB 互換) の料金」 を参照してください。

Amazon DocumentDB 5.0 へのアップグレードまたは移行

MongoDB 3.6 または MongoDB 4.0 から Amazon DocumentDB 5.0 への移行は、AWS DMS または mongodump、mongorestore、mongoimport および mongoexport のようなユーティリティを利用して行うことができます。移行の手順については、「を使用した Amazon DocumentDB クラスターのアップグレード AWS Database Migration Service」 を参照してください。

機能的な違い

Amazon DocumentDB 4.0 と 5.0 の機能的な違い

Amazon DocumentDB 5.0 のリリースでは、Amazon DocumentDB 4.0 と Amazon DocumentDB 5.0 の機能的な違いがあります。

- バックアップビルトインロールが `serverStatus` をサポートするようになりました。アクション - バックアップロールを持つデベロッパーおよびアプリケーションは、Amazon DocumentDB クラスターの状態に関する統計を収集できます。
- `replSetGetConfig` 出力では `SecondaryDelaySecs` フィールドが `slaveDelay` を置き換えます。
- hello コマンドは `isMaster` を置き換えます - hello は Amazon DocumentDB クラスターの役割を説明するドキュメントを返します。
- Amazon DocumentDB 5.0 では、最初のネストレベルで `$elemMatch` 演算子によるインデックススキャンをサポートするようになりました。インデックススキャンは、クエリのみのフィルタに 1 つのレベルの `$elemMatch` フィルタがある場合はサポートされますが、ネストされた `$elemMatch` クエリが含まれる場合はサポートされません。

  たとえば、Amazon DocumentDB 5.0 では、ネストされたレベルに `$elemMatch` 演算子を含めても、Amazon DocumentDB 4.0 のように値を返しません。

```
db.foo.insert(
[
    {a: {b: 5}},
    {a: {b: [5]}},
    {a: {b: [3, 7]}},
    {a: [{b: 5}]},
    {a: [{b: 3}, {b: 7}]},
    {a: [{b: [5]}]},
    {a: [{b: [3, 7]}]},
    {a: [[{b: 5}]]},
```

```
    {a: [[{b: 3}, {b: 7}]]},
    {a: [[{b: [5]}]]},
    {a: [[{b: [3, 7]}]]}
]);

// DocumentDB 5.0
> db.foo.find({a: {$elemMatch: {b: {$elemMatch: {$lt: 6, $gt: 4}}}}}, {_id: 0})
{ "a" : [ { "b" : [ 5 ] } ] }

// DocumentDB 4.0
> db.foo.find({a: {$elemMatch: {b: {$elemMatch: {$lt: 6, $gt: 4}}}}}, {_id: 0})
{ "a" : [ { "b" : [ 5 ] } ] }
{ "a" : [ [ { "b" : [ 5 ] } ] ] }
```

- Amazon DocumentDB 4.0 では、「$」プロジェクションはすべてのドキュメントとすべてのフィールドを返します。Amazon DocumentDB 5.0 では、「$」プロジェクションを使用する find コマンドは、「$」プロジェクションと一致したフィールドのみを含むクエリパラメータに一致するドキュメントを返します。
- Amazon DocumentDB 5.0 では、`$regex` および `$options` クエリパラメータを指定した find コマンドは、「`$regex` と `$options` の両方にオプションを設定することはできません」というエラーを返します。
- Amazon DocumentDB 5.0 では、以下の場合に `$indexOfCP` は「-1」を返します。
  - 文字列式内に部分文字列がない場合、または
  - 開始値が終了値より大きい場合、または
  - 開始値がは文字列のバイト長より大きい場合。
- Amazon DocumentDB 4.0 では、開始位置が文字列の終了またはバイト長よりも大きい数値の場合、`$indexOfCP` は「0」を返します。
- Amazon DocumentDB 5.0 では、たとえば `{"_id.nestedField" : 1}` のような `_id fields` の射影演算は、投影されたフィールドのみを含むドキュメントが返されます。これに対し、Amazon DocumentDB 4.0 の場合、入れ子にされたフィールドプロジェクトコマンドではどのドキュメントも除外しません。

MongoDB 4.0 の互換

トピック

- Amazon DocumentDB 4.0 機能
- Amazon DocumentDB 4.0 の使用開始

- Amazon DocumentDB 4.0 へのアップグレードまたは移行
- 機能的な違い

Amazon DocumentDB 4.0 機能

Amazon DocumentDB 4.0 では、ACID トランザクションや変更ストリームの改良など、多くの新機能が導入されました。Amazon DocumentDB 4.0 で導入された主要な機能をいくつかを以下に要約します。機能の全リストを見るには、「リリースノート」 を参照してください。

- ACID トランザクション: Amazon DocumentDB は、複数のドキュメント、ステートメント、コレクション、およびデータベース間でトランザクションを実行する機能をサポートするようになりました。トランザクションは、Amazon DocumentDB クラスター内の 1 つまたは複数のドキュメントに対し、アトミックで一貫性、分離、耐久性 (ACID) のあるオペレーションを実行できるようにすることで、アプリケーション開発を簡素化します。詳細については、Amazon DocumentDB でのトランザクション を参照してください。
- 変更ストリーム: これで、クラスターレベル (`client.watch()` または `mongo.watch()`) とデータベース (`db.watch()`) で変更ストリームを開くことができるようになりました 。`startAtOperationTime` をクリックして変更ストリームカーソルを開き、最後に、変更ストリームの保持期間を 7 日 (以前は 24 時間) に延長できます。詳細については、「Amazon DocumentDB を用いた変更ストリームの使用」を参照してください。
- AWS Database Migration Service (AWS DMS): AWS DMS を使用してMongoDB 4.0 ワークロードを Amazon DocumentDB に移行できるようになりました。 は、MongoDB 4.0 ソース、Amazon DocumentDB 4.0 ターゲット、Amazon DocumentDB 3.6 ソースをサポートし、Amazon DocumentDB 3.6 と 4.0 の間でアップグレードを実行 AWS DMS できるようになりました。詳細については、AWS DMS ドキュメント を参照してください。
- パフォーマンスとインデックス作成: `$lookup` で、インデックスを使用できるようになり、1 フィールドまたは 1 フィールドと `_id` フィールドを含む投影による検索クエリは、インデックスから直接提供でき、コレクション (カバークエリ) から読み取る必要がなくなり、`hint()` と `findAndModify` をする機能、`$addToSet` のパフォーマンスの最適化、および全体的なインデックスサイズを小さくするための改良が行われました。詳細については、「リリースノート」を参照してください。
- 演算子: Amazon DocumentDB 4.0 では、いくつかの新しい集計演算子がサポートされるようになりました: `$ifNull`、`$replaceRoot`、`$setIsSubset`、`$setIntersection`、`$setUnion`、`$setEquals`。Amazon DocumentDB でサポートされている MongoDB API、オペレーション、およびデータ型 でサポートされるすべての MongoDB API、オペレーション、およびデータタイプを確認できます。

- ロールベースのアクセスコントロール(RBAC): `ListCollection` と `ListDatabase` の両方のコマンドで、オプションとして `authorizedCollections` と `authorizedDatabases` パラメータを使用することができるようになり、ユーザーが `listCollections` と `listDatabase` ロールを必要とすることなくアクセス許可を持つコレクションとデータベースを一覧表示できるようになりました。また、`KillCursor` ロールを必要とすることなく自分のカーソルを強制終了させる機能もあります。

Amazon DocumentDB は、すべての MongoDB 4.0 機能をサポートしているわけではありません。Amazon DocumentDB 4.0 を構築する際、お客様から最も多く構築するよう求められた特徴と機能から逆算していきました。今後もお客様の要望をもとに、MongoDB 4.0 の機能を追加していく予定です。例えば、Amazon DocumentDB 4.0 では、現在、MongoDB 4.0 で導入されていた型変換演算子または文字列演算子をサポートしていません。サポートされている API の最新リストについては、Amazon DocumentDB でサポートされている MongoDB API、オペレーション、およびデータ型を参照してください。

Amazon DocumentDB 4.0 の使用開始

Amazon DocumentDB 4.0 の使用を開始するには、入門ガイド を参照してください。新しい Amazon DocumentDB 4.0 クラスターは、 AWS Management Console または AWS SDK AWS CLI、または を使用して作成できます AWS CloudFormation。Amazon DocumentDB に接続する場合、MongoDB 4.0 以降と互換性のある MongoDB ドライバまたはユーティリティを使用する必要があります。

> **Note**
>
> AWS SDK AWS CLI、、または を使用する場合 AWS CloudFormation、エンジンバージョンはデフォルトで 5.0.0 になります。Amazon DocumentDB 4.0 クラスターを新規に作成する場合はパラメータ `engineVersion = 4.0.0` を、あるいは Amazon DocumentDB 3.6 クラスターを新規に作成する場合は`engineVersion = 3.6.0` を明示的に指定する必要があります。特定の Amazon DocumentDB クラスターでは、 AWS CLI を使用してクラスターのバージョンを確認する`describe-db-clusters`か、Amazon DocumentDB マネジメントコンソールを使用して特定のクラスターのエンジンバージョン番号を表示できます。

Amazon DocumentDB 4.0 は `r5`、`r6g`、`t3.medium` および、クラスターの `t4g.medium` インスタンスタイプをサポートし、サポートされているすべてのリージョンで利用できます。Amazon

DocumentDB 4.0 を使用しても追加コストは発生しません。料金の詳細については、Amazon DocumentDB (MongoDB 互換) の料金 を参照してください。

Amazon DocumentDB 4.0 へのアップグレードまたは移行

MongoDB 3.6 または MongoDB 4.0 から Amazon DocumentDB 4.0 への移行は、AWS DMS または mongodump、mongorestore、mongoimport および mongoexport のようなユーティリティを利用して行うことができます。同様に、同じツールを使用して Amazon DocumentDB 3.6 から Amazon DocumentDB 4.0 にアップグレードできます。移行の手順については、を使用した Amazon DocumentDB クラスターのアップグレード AWS Database Migration Service を参照してください。

機能的な違い

Amazon DocumentDB 3.6 と 4.0 の機能的な違い

Amazon DocumentDB 4.0のリリースに伴い、Amazon DocumentDB 3.6と Amazon DocumentDB 4.0 には機能上の違いがあります。

- ネストドキュメントの投影: Amazon DocumentDB 3.6 は、投影法を適用するときに、ネストドキュメントの最初のフィールドを考慮します。ただし、Amazon DocumentDB 4.0 はサブドキュメントを解析し、各サブドキュメントにも投影法を適用します。たとえば、投影が `"a.b.c": 1` の場合、両方のバージョンでの動作は同じです。ただし、投影が `{a:{b:{c:1}}}` だとすると、Amazon DocumentDB 3.6 は投影を 'a' にのみ適用し、'b' や 'c' には適用しません。
- **`minKey`** と **`maxKey`** 動作: Amazon DocumentDB 4.0 では、`{x:{$gt:MaxKey}}` では何も返しませんが、`{x:{$lt:MaxKey}}` ではすべてを返します。
- ドキュメント比較の相違: サブドキュメント内 (例: b の `{"_id" :1, "a" :{"b":1}}`)の 異なるタイプ (double、int、long) の数値比較では、数値データ型とドキュメントの各レベルにわたって一貫した出力が提供されるようになりました。

Amazon DocumentDB 4.0 と MongoDB 4.0 の機能的な違い

以下は、Amazon DocumentDB 4.0 と MongoDB 4.0 の機能的な違いです。

- パスに空のキーがある場合のルックアップ: コレクションに、配列内 (例: `{"x" : [ { "" : 10 }, { "b" : 20 } ]}`) に空のキーを持つドキュメントが含まれている場合、そしてクエリで使用されるキーが空の文字列 (例: `x.`) で終わる場合は、Amazon DocumentDB は配列内のすべてのドキュメントを走査するのでそのドキュメントを返しますが、一方 MongoDB はそのドキュメントを返しません。

- パスの**$setOnInsert** に沿った **$**: フィールド演算子 `$setOnInsert` は Amazon DocumentDBのパスの `$` との組み合わせでは動作しませんが、これは MongoDB 4.0 と同じです。

Amazon DocumentDB でのトランザクション

Amazon DocumentDB (MongoDB 互換) は、ACID トランザクションを含む MongoDB 4.0 の互換性をサポートしています。複数のドキュメント、ステートメント、コレクション、およびデータベース間でトランザクションを実行できます。トランザクションは、Amazon DocumentDB クラスター内の 1 つまたは複数のドキュメントに対し、原始的、一貫性、分離、耐久性 (ACID) オペレーションを実行できるようにすることで、アプリケーション開発を簡素化します。トランザクションの一般的なユースケースには、財務処理、注文の履行と管理、マルチプレイヤーゲームの構築などがあります。

トランザクションを有効にするために、追加料金はかかりません。支払いは、トランザクションの一部として消費した読み書きの IO に対してのみ支払います。

トピック

- 要件
- ベストプラクティス
- 制限
- モニタリングと診断
- トランザクション分離レベル
- ユースケース
- サポートされているコマンド
- サポート対象外の機能
- セッション
- トランザクションエラー

要件

トランザクション機能を使用するには、次の要件を満たしている必要があります。

- Amazon DocumentDB 4.0 エンジンを使用する必要があります。
- MongoDB 4.0 以降と互換性のあるドライバを使用する必要があります。

ベストプラクティス

Amazon DocumentDB でトランザクションを最も多く利用できるようにするためのベストプラクティスをいくつか紹介します。

- トランザクションの完了後は、必ずトランザクションをコミットまたは中断してください。トランザクションが不完全な状態のままにすると、データベースリソースが結びつき、書き込みの競合が発生する可能性があります。
- トランザクションは、必要なコマンド数を最小限に抑えることをお勧めします。複数のステートメントを持つトランザクションを複数の小さなトランザクションに分割できる場合は、タイムアウトの可能性を減らすためにこれを行うことをお勧めします。長時間実行される読み取りではなく、短いトランザクションを作成することを常に目指してください。

制限

- Amazon DocumentDB は、トランザクション内のカーソルをサポートしていません。
- Amazon DocumentDB はトランザクションで新しいコレクションを作成できず、存在しないコレクションに対してクエリ/更新することはできません。
- ドキュメントレベルの書き込みロックには 1 分間のタイムアウトが適用されますが、これはユーザーによって設定できません。
- Amazon DocumentDB では、再試行可能な書き込み、再試行可能なコミット、および再試行可能な中止コマンドはサポートされていません。例外: mongo シェル を使用している場合は、どのコード文字列にも `retryWrites=false` コマンドを含めないでください。デフォルトでは、再試行可能な書き込みは無効になっています。`retryWrites=false` を含めると、通常の読み取りコマンドでは失敗する可能性があります。
- 各 Amazon DocumentDB インスタンスには、一度にインスタンスでオープンされる同時トランザクションの数に上限があります。制限事項については、「インスタンス制限」を参照してください。
- 特定のトランザクションでは、トランザクションログのサイズは 32 MB 未満である必要があります。
- Amazon DocumentDB はトランザクション内で `count()` をサポートしていますが、すべてのドライバがこの機能をサポートしているわけではありません。別の方法として、`countDocuments()` API を使用して、count クエリをクライアント側の集計クエリに変換します。

- トランザクションの実行制限は 1 分で、セッションには 30 分のタイムアウトがあります。トランザクションがタイムアウトすると、トランザクションは中止され、既存のトランザクションのセッション内で後続のコマンドが発行されると、次のエラーが表示されます。

```
WriteCommandError({
"ok" : 0,
"operationTime" : Timestamp(1603491424, 627726),
"code" : 251,
"errmsg" : "Given transaction number 0 does not match any in-progress transactions."
}
```

モニタリングと診断

Amazon DocumentDB 4.0 でのトランザクションのサポートにより、トランザクションのモニタリングに役立つ追加の CloudWatch メトリックスが追加されました。

新しい CloudWatch メトリクス

- `DatabaseTransactions`: 1 分間に実行された未処理のトランザクションの数。
- `DatabaseTransactionsAborted`: 1 分間に中断された未処理のトランザクションの数。
- `DatabaseTransactionsMax`: 1 分間にインスタンスで開かれるトランザクションの最大数。
- `TransactionsAborted`: 1 分間にインスタンスで中断されたトランザクションの数。
- `TransactionsCommitted`: 1 分間にインスタンスでコミットされたトランザクションの数。
- `TransactionsOpen`: 1 分間で実行されるインスタンスで開かれるトランザクションの数。
- `TransactionsOpenMax`: 1 分間にインスタンスで開いているトランザクションの最大数。
- `TransactionsStarted`: 1 分間にインスタンスで開始されたトランザクションの数。

> **Note**
>
> Amazon DocumentDB の CloudWatch メトリックスの詳細については、Amazon DocumentDB と CloudWatch のモニタリング。

さらに、新しいフィールドが `currentOp` `lsid`、`transactionThreadId`、`idle transaction` のための新しい状態、および `serverStatus` トランザクション:

currentActive、currentInactive、 currentOpen、totalAborted、totalCommitted、および totalStarted に追加されました。

トランザクション分離レベル

トランザクションを開始するときは、以下の例writeConcernに示すように、 readConcernと の両方を指定できます。

```
mySession.startTransaction({readConcern: {level: 'snapshot'}, writeConcern:
{w: 'majority'}});
```

readConcern を使用する場合、Amazon DocumentDB はデフォルトでスナップショットアイソレーションをサポートしています。readConcern に local、available、majority が指定された場合、Amazon DocumentDB は readConcern レベルをスナップショットにアップグレードします。Amazon DocumentDB は線形化可能な readConcern をサポートしていません。このような読み取り懸念を指定すると、エラーが発生します。

writeConcern を使用する場合、Amazon DocumentDB はデフォルトで多数数をサポートしており、3 つの AZ にわたってデータの 4 つのコピーが永続化されると、書き込みクォーラムが達成されます。低い writeConcern が指定されている場合、Amazon DocumentDB は writeConcern を過半数にアップグレードします。さらに、Amazon DocumentDB の書き込みはすべてジャーナル処理され、ジャーナリングを無効にすることはできません。

ユースケース

このセクションでは、トランザクションの 2 つのユースケース、マルチステートメントとマルチコレクションについて説明します。

マルチステートメントトランザクション

Amazon DocumentDB トランザクションはマルチステートメントです。つまり、明示的なコミットまたはロールバックを使用して、複数のステートメントにまたがるトランザクションを記述できます。単一のアトミック操作として、insert、update、delete、および findAndModify のアクションをグループ分けできます。

マルチステートメントトランザクションの一般的なユースケースは、デビットクレジットトランザクションです。例えば、友達に服のお金を借りたとします。アカウントから 500 ドルのデビット (引き

出し) を実行し、友達のアカウントに 500 ドル (入金) をクレジットする必要があります。そのオペレーションを実行するには、1 つのトランザクション内でデビットオペレーションとクレジットオペレーションの両方を実行して、アトミック性を確保します。これにより、500 ドルがアカウントから引き落とされ、友達のアカウントに入金されないシナリオが防止されます。このユースケースは次のようになります。

```
// *** Transfer $500 from Alice to Bob inside a transaction: Success Scenario***
// Setup bank account for Alice and Bob. Each have $1000 in their account

var databaseName = "bank";
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var bankDB = session.getDatabase(databaseName);
var accountColl = bankDB[collectionName];
accountColl.drop();

accountColl.insert({name: "Alice", balance: 1000});
accountColl.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountColl.find({"name": "Alice"}).next().balance;
var newAliceBalance = aliceBalance - amountToTransfer;
accountColl.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountColl.find({"name": "Alice"}).next().balance;

// add $500 to Bob's account
var bobBalance = accountColl.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountColl.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountColl.find({"name": "Bob"}).next().balance;

session.commitTransaction();

accountColl.find();

// *** Transfer $500 from Alice to Bob inside a transaction: Failure Scenario***

// Setup bank account for Alice and Bob. Each have $1000 in their account
```

```
var databaseName = "bank";
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var bankDB = session.getDatabase(databaseName);
var accountColl = bankDB[collectionName];
accountColl.drop();

accountColl.insert({name: "Alice", balance: 1000});
accountColl.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountColl.find({"name": "Alice"}).next().balance;
var newAliceBalance = aliceBalance - amountToTransfer;
accountColl.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountColl.find({"name": "Alice"}).next().balance;

session.abortTransaction();
```

マルチコレクショントランザクション

また、トランザクションはマルチコレクションです。つまり、単一のトランザクション内および複数のコレクションにわたって複数の操作を実行するために使用できます。これにより、データの整合性が保たれ、データの整合性が維持されます。コマンドを単一の <> としてコミットする場合、トランザクションはすべてまたは何も実行されません。その場合、トランザクションはすべて成功するか、すべて失敗します。

以下は、マルチコレクショントランザクションの例で、複数ステートメントトランザクションの例と同じシナリオとデータを使用します。

```
// *** Transfer $500 from Alice to Bob inside a transaction: Success Scenario***

// Setup bank account for Alice and Bob. Each have $1000 in their account
var amountToTransfer = 500;
var collectionName = "account";

var session = db.getMongo().startSession({causalConsistency: false});
```

```
var accountCollInBankA = session.getDatabase("bankA")[collectionName];
var accountCollInBankB = session.getDatabase("bankB")[collectionName];

accountCollInBankA.drop();
accountCollInBankB.drop();

accountCollInBankA.insert({name: "Alice", balance: 1000});
accountCollInBankB.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountCollInBankA.find({"name": "Alice"}).next().balance;
var newAliceBalance = aliceBalance - amountToTransfer;
accountCollInBankA.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountCollInBankA.find({"name": "Alice"}).next().balance;

// add $500 to Bob's account
var bobBalance = accountCollInBankB.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountCollInBankB.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountCollInBankB.find({"name": "Bob"}).next().balance;

session.commitTransaction();

accountCollInBankA.find(); // Alice holds $500 in bankA
accountCollInBankB.find(); // Bob holds $1500 in bankB

// *** Transfer $500 from Alice to Bob inside a transaction: Failure Scenario***

// Setup bank account for Alice and Bob. Each have $1000 in their account
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var accountCollInBankA = session.getDatabase("bankA")[collectionName];
var accountCollInBankB = session.getDatabase("bankB")[collectionName];

accountCollInBankA.drop();
accountCollInBankB.drop();

accountCollInBankA.insert({name: "Alice", balance: 1000});
accountCollInBankB.insert({name: "Bob", balance: 1000});
```

```
session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountCollInBankA.find({"name": "Alice"}).next().balance;
var newAliceBalance = aliceBalance - amountToTransfer;
accountCollInBankA.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountCollInBankA.find({"name": "Alice"}).next().balance;

// add $500 to Bob's account
var bobBalance = accountCollInBankB.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountCollInBankB.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountCollInBankB.find({"name": "Bob"}).next().balance;

session.abortTransaction();

accountCollInBankA.find(); // Alice holds $1000 in bankA
accountCollInBankB.find(); // Bob holds $1000 in bankB
```

コールバック API のトランザクション API の例

コールバック API は 4.2 以降のドライバでのみ使用できます。

Javascript

次のコードは、Javascript で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// *** Transfer $500 from Alice to Bob inside a transaction: Success ***
// Setup bank account for Alice and Bob. Each have $1000 in their account
var databaseName = "bank";
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var bankDB = session.getDatabase(databaseName);
var accountColl = bankDB[collectionName];
accountColl.drop();

accountColl.insert({name: "Alice", balance: 1000});
accountColl.insert({name: "Bob", balance: 1000});
```

```
session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountColl.find({"name": "Alice"}).next().balance;
assert(aliceBalance >= amountToTransfer);
var newAliceBalance = aliceBalance - amountToTransfer;
accountColl.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountColl.find({"name": "Alice"}).next().balance;
assert.eq(newAliceBalance, findAliceBalance);

// add $500 to Bob's account
var bobBalance = accountColl.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountColl.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountColl.find({"name": "Bob"}).next().balance;
assert.eq(newBobBalance, findBobBalance);

session.commitTransaction();

accountColl.find();
```

Node.js

次のコードは、Node.js で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// Node.js callback API:

const bankDB = await mongoclient.db("bank");
var accountColl = await bankDB.createCollection("account");
var amountToTransfer = 500;

const session = mongoclient.startSession({causalConsistency: false});
await accountColl.drop();

await accountColl.insertOne({name: "Alice", balance: 1000}, { session });
await accountColl.insertOne({name: "Bob", balance: 1000}, { session });

const transactionOptions = {
    readConcern: { level: 'snapshot' },
    writeConcern: { w: 'majority' }
    };
```

```
// deduct $500 from Alice's account
var aliceBalance = await accountColl.findOne({name: "Alice"}, {session});
assert(aliceBalance.balance >= amountToTransfer);
var newAliceBalance = aliceBalance - amountToTransfer;
session.startTransaction(transactionOptions);
await accountColl.updateOne({name: "Alice"}, {$set: {balance: newAliceBalance}},
 {session });
await session.commitTransaction();
aliceBalance = await accountColl.findOne({name: "Alice"}, {session});
assert(newAliceBalance == aliceBalance.balance);

// add $500 to Bob's account
var bobBalance = await accountColl.findOne({name: "Bob"}, {session});
var newBobBalance = bobBalance.balance + amountToTransfer;
session.startTransaction(transactionOptions);
await accountColl.updateOne({name: "Bob"}, {$set: {balance: newBobBalance}},
 {session });
await session.commitTransaction();
bobBalance = await accountColl.findOne({name: "Bob"}, {session});
assert(newBobBalance == bobBalance.balance);
```

C#

次のコードは、C# で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// C# Callback API

var dbName = "bank";
var collName = "account";
var amountToTransfer = 500;

using (var session = client.StartSession(new ClientSessionOptions{CausalConsistency
 = false}))
{
    var bankDB = client.GetDatabase(dbName);
    var accountColl = bankDB.GetCollection<BsonDocument>(collName);
    bankDB.DropCollection(collName);
    accountColl.InsertOne(session, new BsonDocument { {"name", "Alice"}, {"balance",
 1000 } });
    accountColl.InsertOne(session, new BsonDocument { {"name", "Bob"}, {"balance",
 1000 } });
```

```
   // start transaction
   var transactionOptions = new TransactionOptions(
           readConcern: ReadConcern.Snapshot,
           writeConcern: WriteConcern.WMajority);
   var result = session.WithTransaction(
       (sess, cancellationtoken) =>
       {
           // deduct $500 from Alice's account
           var aliceBalance = accountColl.Find(sess,
Builders<BsonDocument>.Filter.Eq("name",
"Alice")).FirstOrDefault().GetValue("balance");
           Debug.Assert(aliceBalance >= amountToTransfer);
           var newAliceBalance = aliceBalance.AsInt32 - amountToTransfer;
           accountColl.UpdateOne(sess, Builders<BsonDocument>.Filter.Eq("name",
"Alice"),
                                    Builders<BsonDocument>.Update.Set("balance",
newAliceBalance));
           aliceBalance = accountColl.Find(sess,
Builders<BsonDocument>.Filter.Eq("name",
"Alice")).FirstOrDefault().GetValue("balance");
           Debug.Assert(aliceBalance == newAliceBalance);

           // add $500 from Bob's account
           var bobBalance = accountColl.Find(sess,
Builders<BsonDocument>.Filter.Eq("name",
"Bob")).FirstOrDefault().GetValue("balance");
           var newBobBalance = bobBalance.AsInt32 + amountToTransfer;
           accountColl.UpdateOne(sess, Builders<BsonDocument>.Filter.Eq("name",
"Bob"),
                                    Builders<BsonDocument>.Update.Set("balance",
newBobBalance));
           bobBalance = accountColl.Find(sess,
Builders<BsonDocument>.Filter.Eq("name",
"Bob")).FirstOrDefault().GetValue("balance");
           Debug.Assert(bobBalance == newBobBalance);

           return "Transaction committed";
       }, transactionOptions);
  // check values outside of transaction
   var aliceNewBalance = accountColl.Find(Builders<BsonDocument>.Filter.Eq("name",
"Alice")).FirstOrDefault().GetValue("balance");
   var bobNewBalance = accountColl.Find(Builders<BsonDocument>.Filter.Eq("name",
"Bob")).FirstOrDefault().GetValue("balance");
   Debug.Assert(aliceNewBalance ==  500);
```

```
    Debug.Assert(bobNewBalance ==  1500);
}
```

Ruby

次のコードは、Ruby で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// Ruby Callback API

dbName = "bank"
collName = "account"
amountToTransfer = 500

session = client.start_session(:causal_consistency=> false)
bankDB = Mongo::Database.new(client, dbName)
accountColl = bankDB[collName]
accountColl.drop()

accountColl.insert_one({"name"=>"Alice", "balance"=>1000})
accountColl.insert_one({"name"=>"Bob", "balance"=>1000})

    # start transaction
    session.with_transaction(read_concern: {level: :snapshot}, write_concern:
 {w: :majority}) do
        # deduct $500 from Alice's account
        aliceBalance = accountColl.find({"name"=>"Alice"}, :session=>
 session).first['balance']
        assert aliceBalance >= amountToTransfer
        newAliceBalance = aliceBalance - amountToTransfer
        accountColl.update_one({"name"=>"Alice"}, { "$set" =>
 {"balance"=>newAliceBalance} }, :session=> session)
        aliceBalance = accountColl.find({"name"=>>"Alice"}, :session=>
 session).first['balance']
        assert_equal(newAliceBalance, aliceBalance)

        # add $500 from Bob's account
        bobBalance = accountColl.find({"name"=>"Bob"}, :session=>
 session).first['balance']
        newBobBalance = bobBalance + amountToTransfer
        accountColl.update_one({"name"=>"Bob"}, { "$set" =>
 {"balance"=>newBobBalance} }, :session=> session)
```

```
        bobBalance = accountColl.find({"name"=>"Bob"}, :session=>
 session).first['balance']
        assert_equal(newBobBalance, bobBalance)
    end

   # check results outside of transaction
    aliceBalance = accountColl.find({"name"=>"Alice"}).first['balance']
    bobBalance = accountColl.find({"name"=>"Bob"}).first['balance']
    assert_equal(aliceBalance, 500)
    assert_equal(bobBalance, 1500)

session.end_session
```

Go

次のコードは、Go で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// Go - Callback API
type Account struct {
    Name string
    Balance  int
}

ctx := context.TODO()

dbName := "bank"
collName := "account"
amountToTransfer := 500

session, err := client.StartSession(options.Session().SetCausalConsistency(false))
assert.NilError(t, err)
defer session.EndSession(ctx)

bankDB := client.Database(dbName)
accountColl := bankDB.Collection(collName)
accountColl.Drop(ctx)

_, err = accountColl.InsertOne(ctx, bson.M{"name" : "Alice", "balance":1000})
_, err = accountColl.InsertOne(ctx, bson.M{"name" : "Bob", "balance":1000})

transactionOptions := options.Transaction().SetReadConcern(readconcern.Snapshot()).
```

```
 SetWriteConcern(writeconcern.New(writeconcern.WMajority()))
_, err = session.WithTransaction(ctx, func(sessionCtx mongo.SessionContext)
 (interface{}, error) {
    var result Account
    // deduct $500 from Alice's account
    err = accountColl.FindOne(sessionCtx, bson.M{"name": "Alice"}).Decode(&result)
    aliceBalance := result.Balance
    newAliceBalance := aliceBalance - amountToTransfer
    _, err = accountColl.UpdateOne(sessionCtx, bson.M{"name": "Alice"},
 bson.M{"$set": bson.M{"balance": newAliceBalance}})
    err = accountColl.FindOne(sessionCtx, bson.M{"name": "Alice"}).Decode(&result)
    aliceBalance = result.Balance
    assert.Equal(t, aliceBalance, newAliceBalance)

    // add $500 to Bob's account
    err = accountColl.FindOne(sessionCtx, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance := result.Balance
    newBobBalance := bobBalance + amountToTransfer
    _, err = accountColl.UpdateOne(sessionCtx, bson.M{"name": "Bob"}, bson.M{"$set":
 bson.M{"balance": newBobBalance}})
    err = accountColl.FindOne(sessionCtx, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance = result.Balance
    assert.Equal(t, bobBalance, newBobBalance)

    if err != nil {
        return nil, err
    }
    return "transaction committed", err
}, transactionOptions)

// check results outside of transaction
var result Account
err = accountColl.FindOne(ctx, bson.M{"name": "Alice"}).Decode(&result)
aliceNewBalance := result.Balance
err = accountColl.FindOne(ctx, bson.M{"name": "Bob"}).Decode(&result)
bobNewBalance := result.Balance
assert.Equal(t, aliceNewBalance, 500)
assert.Equal(t, bobNewBalance, 1500)
```

Java

次のコードは、Java で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// Java (sync) - Callback API
MongoDatabase bankDB = mongoClient.getDatabase("bank");
MongoCollection accountColl = bankDB.getCollection("account");
accountColl.drop();
int amountToTransfer = 500;

// add sample data
accountColl.insertOne(new Document("name", "Alice").append("balance", 1000));
accountColl.insertOne(new Document("name", "Bob").append("balance", 1000));

TransactionOptions txnOptions = TransactionOptions.builder()
        .readConcern(ReadConcern.SNAPSHOT)
        .writeConcern(WriteConcern.MAJORITY)
        .build();
ClientSessionOptions sessionOptions =
 ClientSessionOptions.builder().causallyConsistent(false).build();
try ( ClientSession clientSession = mongoClient.startSession(sessionOptions) ) {
    clientSession.withTransaction(new TransactionBody<Void>() {
        @Override
        public Void execute() {
            // deduct $500 from Alice's account
            List<Document> documentList = new ArrayList<>();
            accountColl.find(clientSession, new Document("name",
 "Alice")).into(documentList);
            int aliceBalance = (int) documentList.get(0).get("balance");
            int newAliceBalance = aliceBalance - amountToTransfer;

            accountColl.updateOne(clientSession, new Document("name", "Alice"), new
 Document("$set", new Document("balance", newAliceBalance)));

            // check Alice's new balance
            documentList = new ArrayList<>();
            accountColl.find(clientSession, new Document("name",
 "Alice")).into(documentList);
            int updatedBalance = (int) documentList.get(0).get("balance");
            Assert.assertEquals(updatedBalance, newAliceBalance);

            // add $500 to Bob's account
            documentList = new ArrayList<>();
            accountColl.find(clientSession, new Document("name",
 "Bob")).into(documentList);
            int bobBalance = (int) documentList.get(0).get("balance");
            int newBobBalance = bobBalance + amountToTransfer;
```

```
            accountColl.updateOne(clientSession, new Document("name", "Bob"), new
 Document("$set", new Document("balance", newBobBalance)));

            // check Bob's new balance
            documentList = new ArrayList<>();
            accountColl.find(clientSession, new Document("name",
 "Bob")).into(documentList);
            updatedBalance = (int) documentList.get(0).get("balance");
            Assert.assertEquals(updatedBalance, newBobBalance);

            return null;
        }
    }, txnOptions);
}
```

C

次のコードは、C で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// Sample Code for C with Callback

#include <bson.h>
#include <mongoc.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>

typedef struct {
    int64_t balance;
    bson_t *account;
    bson_t *opts;
    mongoc_collection_t *collection;
} ctx_t;

bool callback_session (mongoc_client_session_t *session, void *ctx, bson_t **reply,
 bson_error_t *error)
{
    bool r = true;
    ctx_t *data = (ctx_t *) ctx;
    bson_t local_reply;
    bson_t *selector = data->account;
```

```
    bson_t *update = BCON_NEW ("$set", "{", "balance", BCON_INT64 (data->balance),
 "}");

    mongoc_collection_update_one (data->collection, selector, update, data->opts,
 &local_reply, error);

    *reply = bson_copy (&local_reply);
    bson_destroy (&local_reply);
    bson_destroy (update);
    return r;
}

void test_callback_money_transfer(mongoc_client_t* client, mongoc_collection_t*
 collection, int amount_to_transfer){

    bson_t reply;
    bool r = true;
    const bson_t *doc;
    bson_iter_t iter;
    ctx_t alice_ctx;
    ctx_t bob_ctx;
    bson_error_t error;

    // find query
    bson_t *alice_query = bson_new ();
    BSON_APPEND_UTF8(alice_query, "name", "Alice");

    bson_t *bob_query = bson_new ();
    BSON_APPEND_UTF8(bob_query, "name", "Bob");

    // create session
    // set causal consistency to false
    mongoc_session_opt_t *session_opts = mongoc_session_opts_new ();
    mongoc_session_opts_set_causal_consistency (session_opts, false);
    // start the session
    mongoc_client_session_t *client_session = mongoc_client_start_session (client,
 session_opts, &error);

    // add session to options
    bson_t *opts = bson_new();
    mongoc_client_session_append (client_session, opts, &error);

    // deduct 500 from Alice
    // find account balance of Alice
```

```
   mongoc_cursor_t *cursor = mongoc_collection_find_with_opts (collection,
alice_query, NULL, NULL);
   mongoc_cursor_next (cursor, &doc);
   bson_iter_init (&iter, doc);
   bson_iter_find (&iter, "balance");
   int64_t alice_balance = (bson_iter_value (&iter))->value.v_int64;
   assert(alice_balance >= amount_to_transfer);
   int64_t new_alice_balance = alice_balance - amount_to_transfer;

   // set variables which will be used by callback function
   alice_ctx.collection = collection;
   alice_ctx.opts = opts;
   alice_ctx.balance = new_alice_balance;
   alice_ctx.account = alice_query;

   // callback
   r = mongoc_client_session_with_transaction (client_session, &callback_session,
NULL, &alice_ctx, &reply, &error);
   assert(r);

   // find account balance of Alice after transaction
   cursor = mongoc_collection_find_with_opts (collection, alice_query, NULL, NULL);
   mongoc_cursor_next (cursor, &doc);
   bson_iter_init (&iter, doc);
   bson_iter_find (&iter, "balance");
   alice_balance = (bson_iter_value (&iter))->value.v_int64;
   assert(alice_balance == new_alice_balance);
   assert(alice_balance == 500);

       // add 500 to bob's balance
   // find account balance of Bob
   cursor = mongoc_collection_find_with_opts (collection, bob_query, NULL, NULL);
   mongoc_cursor_next (cursor, &doc);
   bson_iter_init (&iter, doc);
   bson_iter_find (&iter, "balance");
   int64_t bob_balance = (bson_iter_value (&iter))->value.v_int64;
   int64_t new_bob_balance = bob_balance + amount_to_transfer;

   bob_ctx.collection = collection;
   bob_ctx.opts = opts;
   bob_ctx.balance = new_bob_balance;
   bob_ctx.account = bob_query;

   // set read & write concern
```

```
    mongoc_read_concern_t *read_concern = mongoc_read_concern_new ();
    mongoc_write_concern_t *write_concern = mongoc_write_concern_new ();
    mongoc_transaction_opt_t *txn_opts = mongoc_transaction_opts_new ();

    mongoc_write_concern_set_w(write_concern, MONGOC_WRITE_CONCERN_W_MAJORITY);
    mongoc_read_concern_set_level(read_concern, MONGOC_READ_CONCERN_LEVEL_SNAPSHOT);
    mongoc_transaction_opts_set_write_concern (txn_opts, write_concern);
    mongoc_transaction_opts_set_read_concern (txn_opts, read_concern);

    // callback
    r = mongoc_client_session_with_transaction (client_session, &callback_session,
 txn_opts, &bob_ctx, &reply, &error);
    assert(r);

 // find account balance of Bob after transaction
    cursor = mongoc_collection_find_with_opts (collection, bob_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    bob_balance = (bson_iter_value (&iter))->value.v_int64;
    assert(bob_balance == new_bob_balance);
    assert(bob_balance == 1500);

    // cleanup
    bson_destroy(alice_query);
    bson_destroy(bob_query);
    mongoc_client_session_destroy(client_session);
    bson_destroy(opts);
    mongoc_transaction_opts_destroy(txn_opts);
    mongoc_read_concern_destroy(read_concern);
    mongoc_write_concern_destroy(write_concern);
    mongoc_cursor_destroy(cursor);
    bson_destroy(doc);
}
int main(int argc, char* argv[]) {
    mongoc_init ();
    mongoc_client_t* client = mongoc_client_new (<connection uri>);
    bson_error_t error;

    // connect to bank db
    mongoc_database_t *database = mongoc_client_get_database (client, "bank");
    // access account collection
    mongoc_collection_t* collection = mongoc_client_get_collection(client, "bank",
 "account");
```

```
    // set amount to transfer
    int64_t amount_to_transfer = 500;
    // delete the collection if already existing
    mongoc_collection_drop(collection, &error);

    // open Alice account
    bson_t *alice_account = bson_new ();
    BSON_APPEND_UTF8(alice_account, "name", "Alice");
    BSON_APPEND_INT64(alice_account, "balance", 1000);

    // open Bob account
    bson_t *bob_account = bson_new ();
    BSON_APPEND_UTF8(bob_account, "name", "Bob");
    BSON_APPEND_INT64(bob_account, "balance", 1000);

    bool r = true;

    r = mongoc_collection_insert_one(collection, alice_account, NULL, NULL, &error);
    if (!r) {printf("Error encountered:%s", error.message);}
    r = mongoc_collection_insert_one(collection, bob_account, NULL, NULL, &error);
    if (!r) {printf("Error encountered:%s", error.message);}

    test_callback_money_transfer(client, collection, amount_to_transfer);

}
```

Python

次のコードは、Python で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// Sample Python code with callback api

import pymongo

def callback(session, balance, query):
    collection.update_one(query, {'$set': {"balance": balance}}, session=session)

client = pymongo.MongoClient(<connection uri>)
rc_snapshot = pymongo.read_concern.ReadConcern('snapshot')
wc_majority = pymongo.write_concern.WriteConcern('majority')
```

```
# To start, drop and create an account collection and insert balances for both Alice
 and Bob
collection = client.get_database("bank").get_collection("account")
collection.drop()
collection.insert_one({"_id": 1, "name": "Alice", "balance": 1000})
collection.insert_one({"_id": 2, "name": "Bob", "balance": 1000})

amount_to_transfer = 500

# deduct 500 from Alice's account
alice_balance = collection.find_one({"name": "Alice"}).get("balance")
assert alice_balance >= amount_to_transfer
new_alice_balance = alice_balance - amount_to_transfer

with client.start_session({'causalConsistency':False}) as session:
    session.with_transaction(lambda s: callback(s, new_alice_balance, {"name":
 "Alice"}), read_concern=rc_snapshot, write_concern=wc_majority)

updated_alice_balance = collection.find_one({"name": "Alice"}).get("balance")
assert updated_alice_balance == new_alice_balance

# add 500 to Bob's account
bob_balance = collection.find_one({"name": "Bob"}).get("balance")
assert bob_balance >= amount_to_transfer
new_bob_balance = bob_balance + amount_to_transfer

with client.start_session({'causalConsistency':False}) as session:
    session.with_transaction(lambda s: callback(s, new_bob_balance, {"name":
 "Bob"}), read_concern=rc_snapshot, write_concern=wc_majority)

updated_bob_balance = collection.find_one({"name": "Bob"}).get("balance")
assert updated_bob_balance == new_bob_balance
```

コア API のトランザクション API の例

Javascript

次のコードは、Javascript で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// *** Transfer $500 from Alice to Bob inside a transaction: Success ***
// Setup bank account for Alice and Bob. Each have $1000 in their account
```

```
var databaseName = "bank";
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var bankDB = session.getDatabase(databaseName);
var accountColl = bankDB[collectionName];
accountColl.drop();

accountColl.insert({name: "Alice", balance: 1000});
accountColl.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountColl.find({"name": "Alice"}).next().balance;
assert(aliceBalance >= amountToTransfer);
var newAliceBalance = aliceBalance - amountToTransfer;
accountColl.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountColl.find({"name": "Alice"}).next().balance;
assert.eq(newAliceBalance, findAliceBalance);

// add $500 to Bob's account
var bobBalance = accountColl.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountColl.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountColl.find({"name": "Bob"}).next().balance;
assert.eq(newBobBalance, findBobBalance);

session.commitTransaction();

accountColl.find();
```

C#

次のコードは、C# で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// C# Core API

public void TransferMoneyWithRetry(IMongoCollection<bSondocument> accountColl,
 IClientSessionHandle session)
{
```

```
    var amountToTransfer = 500;

    // start transaction
   var transactionOptions = new TransactionOptions(
                readConcern: ReadConcern.Snapshot,
                writeConcern: WriteConcern.WMajority);
    session.StartTransaction(transactionOptions);
   try
    {
        // deduct $500 from Alice's account
        var aliceBalance = accountColl.Find(session,
Builders<bSondocument>.Filter.Eq("name",
"Alice")).FirstOrDefault().GetValue("balance");
        Debug.Assert(aliceBalance >= amountToTransfer);
        var newAliceBalance = aliceBalance.AsInt32 - amountToTransfer;
        accountColl.UpdateOne(session, Builders<bSondocument>.Filter.Eq("name",
"Alice"),
                                Builders<bSondocument>.Update.Set("balance",
newAliceBalance));
        aliceBalance = accountColl.Find(session,
Builders<bSondocument>.Filter.Eq("name",
"Alice")).FirstOrDefault().GetValue("balance");
        Debug.Assert(aliceBalance == newAliceBalance);

        // add $500 from Bob's account
        var bobBalance = accountColl.Find(session,
Builders<bSondocument>.Filter.Eq("name",
"Bob")).FirstOrDefault().GetValue("balance");
        var newBobBalance = bobBalance.AsInt32 + amountToTransfer;
        accountColl.UpdateOne(session, Builders<bSondocument>.Filter.Eq("name",
"Bob"),
                                Builders<bSondocument>.Update.Set("balance",
newBobBalance));
        bobBalance = accountColl.Find(session,
Builders<bSondocument>.Filter.Eq("name",
"Bob")).FirstOrDefault().GetValue("balance");
        Debug.Assert(bobBalance == newBobBalance);

    }
    catch (Exception e)
    {
        session.AbortTransaction();
        throw;
    }
```

```
    session.CommitTransaction();
 }

}
public void DoTransactionWithRetry(MongoClient client)
{
    var dbName = "bank";
    var collName = "account";
    using (var session = client.StartSession(new
 ClientSessionOptions{CausalConsistency = false}))
    {
        try
        {
            var bankDB = client.GetDatabase(dbName);
            var accountColl = bankDB.GetCollection<bSondocument>(collName);
            bankDB.DropCollection(collName);
            accountColl.InsertOne(session, new BsonDocument { {"name", "Alice"},
 {"balance", 1000 } });
            accountColl.InsertOne(session, new BsonDocument { {"name", "Bob"},
 {"balance", 1000 } });

            while(true) {
                try
                {
                        TransferMoneyWithRetry(accountColl, session);
                        break;
                }
                catch (MongoException e)
                {
                    if(e.HasErrorLabel("TransientTransactionError"))
                    {
                        continue;
                    }
                    else
                    {
                        throw;
                    }
                }
            }

            // check values outside of transaction
```

```
            var aliceNewBalance =
 accountColl.Find(Builders<bSondocument>.Filter.Eq("name",
 "Alice")).FirstOrDefault().GetValue("balance");
            var bobNewBalance =
 accountColl.Find(Builders<bSondocument>.Filter.Eq("name",
 "Bob")).FirstOrDefault().GetValue("balance");
            Debug.Assert(aliceNewBalance ==  500);
            Debug.Assert(bobNewBalance ==  1500);
        }
        catch (Exception e)
        {
            Console.WriteLine("Error running transaction: " + e.Message);
        }
    }
}
```

Ruby

次のコードは、Ruby で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
# Ruby Core API

def transfer_money_w_retry(session, accountColl)
    amountToTransfer = 500

    session.start_transaction(read_concern: {level: :snapshot}, write_concern:
 {w: :majority})
    # deduct $500 from Alice's account
    aliceBalance = accountColl.find({"name"=>"Alice"}, :session=>
 session).first['balance']
    assert aliceBalance >= amountToTransfer
    newAliceBalance = aliceBalance - amountToTransfer
    accountColl.update_one({"name"=>"Alice"}, { "$set" =>
 {"balance"=>newAliceBalance} }, :session=> session)
    aliceBalance = accountColl.find({"name"=>"Alice"}, :session=>
 session).first['balance']
    assert_equal(newAliceBalance, aliceBalance)

    # add $500 to Bob's account
    bobBalance = accountColl.find({"name"=>"Bob"}, :session=>
 session).first['balance']
    newBobBalance = bobBalance + amountToTransfer
```

```
    accountColl.update_one({"name"=>"Bob"}, { "$set" =>
 {"balance"=>newBobBalance} }, :session=> session)
    bobBalance = accountColl.find({"name"=>"Bob"}, :session=>
 session).first['balance']
    assert_equal(newBobBalance, bobBalance)

    session.commit_transaction

end

def do_txn_w_retry(client)
     dbName = "bank"
    collName = "account"

    session = client.start_session(:causal_consistency=> false)
    bankDB = Mongo::Database.new(client, dbName)
    accountColl = bankDB[collName]
    accountColl.drop()

    accountColl.insert_one({"name"=>"Alice", "balance"=>1000})
    accountColl.insert_one({"name"=>"Bob", "balance"=>1000})

    begin
        transferMoneyWithRetry(session, accountColl)
        puts "transaction committed"
    rescue Mongo::Error => e
        if e.label?('TransientTransactionError')
            retry
        else
            puts "transaction failed"
            raise
        end
    end

    # check results outside of transaction
    aliceBalance = accountColl.find({"name"=>"Alice"}).first['balance']
    bobBalance = accountColl.find({"name"=>"Bob"}).first['balance']
    assert_equal(aliceBalance, 500)
    assert_equal(bobBalance, 1500)

end
```

Go

次のコードは、Go で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// Go - Core API
type Account struct {
    Name string
    Balance  int
}

func transferMoneyWithRetry(sessionContext mongo.SessionContext, accountColl
 *mongo.Collection, t *testing.T) error {
    amountToTransfer := 500

    transactionOptions :=
 options.Transaction().SetReadConcern(readconcern.Snapshot()).

 SetWriteConcern(writeconcern.New(writeconcern.WMajority()))
    if err := sessionContext.StartTransaction(transactionOptions); err != nil {
        panic(err)
    }

    var result Account
    // deduct $500 from Alice's account
    err := accountColl.FindOne(sessionContext, bson.M{"name":
 "Alice"}).Decode(&result)
    aliceBalance := result.Balance
    newAliceBalance := aliceBalance - amountToTransfer
    _, err = accountColl.UpdateOne(sessionContext, bson.M{"name": "Alice"},
 bson.M{"$set": bson.M{"balance": newAliceBalance}})
    if err != nil {
        sessionContext.AbortTransaction(sessionContext)
    }
    err = accountColl.FindOne(sessionContext, bson.M{"name":
 "Alice"}).Decode(&result)
    aliceBalance = result.Balance
    assert.Equal(t, aliceBalance, newAliceBalance)

    // add $500 to Bob's account
    err = accountColl.FindOne(sessionContext, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance := result.Balance
    newBobBalance := bobBalance + amountToTransfer
```

```
    _, err = accountColl.UpdateOne(sessionContext, bson.M{"name": "Bob"},
 bson.M{"$set": bson.M{"balance": newBobBalance}})
    if err != nil {
        sessionContext.AbortTransaction(sessionContext)
    }
    err = accountColl.FindOne(sessionContext, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance = result.Balance
    assert.Equal(t, bobBalance, newBobBalance)

    err = sessionContext.CommitTransaction(sessionContext)
    return err
}

func doTransactionWithRetry(t *testing.T) {
    ctx := context.TODO()

    dbName := "bank"
    collName := "account"
    bankDB := client.Database(dbName)
    accountColl := bankDB.Collection(collName)

    client.UseSessionWithOptions(ctx, options.Session().SetCausalConsistency(false),
 func(sessionContext mongo.SessionContext) error {
        accountColl.Drop(ctx)
        accountColl.InsertOne(sessionContext, bson.M{"name" : "Alice",
 "balance":1000})
        accountColl.InsertOne(sessionContext, bson.M{"name" : "Bob",
 "balance":1000})
        for {
            err := transferMoneyWithRetry(sessionContext, accountColl, t)
            if err == nil {
                println("transaction committed")
                return nil
            }
            if mongoErr := err.(mongo.CommandError);
 mongoErr.HasErrorLabel("TransientTransactionError") {
                continue
            }
            println("transaction failed")
            return err
        }
    })

    // check results outside of transaction
```

```
    var result Account
    accountColl.FindOne(ctx, bson.M{"name": "Alice"}).Decode(&result)
    aliceBalance := result.Balance
    assert.Equal(t, aliceBalance, 500)
    accountColl.FindOne(ctx, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance := result.Balance
    assert.Equal(t, bobBalance, 1500)
}
```

Java

次のコードは、Java で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// Java (sync) - Core API

public void transferMoneyWithRetry() {
   // connect to server
    MongoClientURI mongoURI = new MongoClientURI(uri);
    MongoClient mongoClient = new MongoClient(mongoURI);

    MongoDatabase bankDB = mongoClient.getDatabase("bank");
    MongoCollection accountColl = bankDB.getCollection("account");
    accountColl.drop();

   // insert some sample data
    accountColl.insertOne(new Document("name", "Alice").append("balance", 1000));
    accountColl.insertOne(new Document("name", "Bob").append("balance", 1000));

    while (true) {
        try {
            doTransferMoneyWithRetry(accountColl, mongoClient);
            break;
        } catch (MongoException e) {
            if (e.hasErrorLabel(MongoException.TRANSIENT_TRANSACTION_ERROR_LABEL)) {
                continue;
            } else {
                throw e;
            }
        }
    }
}
```

```
public void doTransferMoneyWithRetry(MongoCollection accountColl, MongoClient
 mongoClient) {
    int amountToTransfer = 500;

   TransactionOptions txnOptions = TransactionOptions.builder()
      .readConcern(ReadConcern.SNAPSHOT)
      .writeConcern(WriteConcern.MAJORITY)
      .build();
    ClientSessionOptions sessionOptions =
 ClientSessionOptions.builder().causallyConsistent(false).build();
    try ( ClientSession clientSession = mongoClient.startSession(sessionOptions) ) {
        clientSession.startTransaction(txnOptions);

        // deduct $500 from Alice's account
        List<Document> documentList = new ArrayList<>();
        accountColl.find(clientSession, new Document("name",
 "Alice")).into(documentList);
        int aliceBalance = (int) documentList.get(0).get("balance");
        Assert.assertTrue(aliceBalance >= amountToTransfer);
        int newAliceBalance = aliceBalance - amountToTransfer;
        accountColl.updateOne(clientSession, new Document("name", "Alice"), new
 Document("$set", new Document("balance", newAliceBalance)));

        // check Alice's new balance
        documentList = new ArrayList<>();
        accountColl.find(clientSession, new Document("name",
 "Alice")).into(documentList);
        int updatedBalance = (int) documentList.get(0).get("balance");
        Assert.assertEquals(updatedBalance, newAliceBalance);

        // add $500 to Bob's account
        documentList = new ArrayList<>();
        accountColl.find(clientSession, new Document("name",
 "Bob")).into(documentList);
        int bobBalance = (int) documentList.get(0).get("balance");
        int newBobBalance = bobBalance + amountToTransfer;
        accountColl.updateOne(clientSession, new Document("name", "Bob"), new
 Document("$set", new Document("balance", newBobBalance)));

        // check Bob's new balance
        documentList = new ArrayList<>();
        accountColl.find(clientSession, new Document("name",
 "Bob")).into(documentList);
        updatedBalance = (int) documentList.get(0).get("balance");
```

```
        Assert.assertEquals(updatedBalance, newBobBalance);

        // commit transaction
        clientSession.commitTransaction();
    }
}
// Java (async) -- Core API
public void transferMoneyWithRetry() {
    // connect to the server
    MongoClient mongoClient = MongoClients.create(uri);

    MongoDatabase bankDB = mongoClient.getDatabase("bank");
    MongoCollection accountColl = bankDB.getCollection("account");
    SubscriberLatchWrapper<Void> dropCallback = new SubscriberLatchWrapper<>();
    mongoClient.getDatabase("bank").drop().subscribe(dropCallback);
    dropCallback.await();

    // insert some sample data
    SubscriberLatchWrapper<InsertOneResult> insertionCallback = new
 SubscriberLatchWrapper<>();
    accountColl.insertOne(new Document("name", "Alice").append("balance",
 1000)).subscribe(insertionCallback);
    insertionCallback.await();

    insertionCallback = new SubscriberLatchWrapper<>();
    accountColl.insertOne(new Document("name", "Bob").append("balance",
 1000)).subscribe(insertionCallback);;
    insertionCallback.await();

    while (true) {
        try {
            doTransferMoneyWithRetry(accountColl, mongoClient);
            break;
        } catch (MongoException e) {
            if (e.hasErrorLabel(MongoException.TRANSIENT_TRANSACTION_ERROR_LABEL)) {
                continue;
            } else {
                throw e;
            }
        }
    }
}
```

```
public void doTransferMoneyWithRetry(MongoCollection accountColl, MongoClient
 mongoClient) {
    int amountToTransfer = 500;

    // start the transaction
    TransactionOptions txnOptions = TransactionOptions.builder()
            .readConcern(ReadConcern.SNAPSHOT)
            .writeConcern(WriteConcern.MAJORITY)
            .build();
    ClientSessionOptions sessionOptions =
 ClientSessionOptions.builder().causallyConsistent(false).build();

    SubscriberLatchWrapper<ClientSession> sessionCallback = new
 SubscriberLatchWrapper<>();
    mongoClient.startSession(sessionOptions).subscribe(sessionCallback);
    ClientSession session = sessionCallback.get().get(0);
    session.startTransaction(txnOptions);

    // deduct $500 from Alice's account
    SubscriberLatchWrapper<Document> findCallback = new SubscriberLatchWrapper<>();
    accountColl.find(session, new Document("name",
 "Alice")).first().subscribe(findCallback);
    Document documentFound = findCallback.get().get(0);
    int aliceBalance = (int) documentFound.get("balance");
    int newAliceBalance = aliceBalance - amountToTransfer;

    SubscriberLatchWrapper<UpdateResult> updateCallback = new
 SubscriberLatchWrapper<>();
    accountColl.updateOne(session, new Document("name",
 "Alice"), new Document("$set", new Document("balance",
 newAliceBalance))).subscribe(updateCallback);
    updateCallback.await();

    // check Alice's new balance
    findCallback = new SubscriberLatchWrapper<>();
    accountColl.find(session, new Document("name",
 "Alice")).first().subscribe(findCallback);
    documentFound = findCallback.get().get(0);
    int updatedBalance = (int) documentFound.get("balance");
    Assert.assertEquals(updatedBalance, newAliceBalance);

    // add $500 to Bob's account
    findCallback = new SubscriberLatchWrapper<>();
```

```
    accountColl.find(session, new Document("name",
 "Bob")).first().subscribe(findCallback);
    documentFound = findCallback.get().get(0);
    int bobBalance = (int) documentFound.get("balance");
    int newBobBalance = bobBalance + amountToTransfer;

    updateCallback = new SubscriberLatchWrapper<>();
    accountColl.updateOne(session, new Document("name", "Bob"), new Document("$set",
 new Document("balance", newBobBalance))).subscribe(updateCallback);
    updateCallback.await();

    // check Bob's new balance
    findCallback = new SubscriberLatchWrapper<>();
    accountColl.find(session, new Document("name",
 "Bob")).first().subscribe(findCallback);
    documentFound = findCallback.get().get(0);
    updatedBalance = (int) documentFound.get("balance");
    Assert.assertEquals(updatedBalance, newBobBalance);

    // commit the transaction
    SubscriberLatchWrapper<Void> transactionCallback = new
 SubscriberLatchWrapper<>();
    session.commitTransaction().subscribe(transactionCallback);
    transactionCallback.await();
}

public class SubscriberLatchWrapper<T> implements Subscriber<T> {

    /**
     * A Subscriber that stores the publishers results and provides a latch so can
 block on completion.
     *
     * @param <T> The publishers result type
     */
    private final List<T> received;
    private final List<RuntimeException> errors;
    private final CountDownLatch latch;
    private volatile Subscription subscription;
    private volatile boolean completed;

    /**
     * Construct an instance
     */
    public SubscriberLatchWrapper() {
```

```
        this.received = new ArrayList<>();
        this.errors = new ArrayList<>();
        this.latch = new CountDownLatch(1);
    }

    @Override
    public void onSubscribe(final Subscription s) {
        subscription = s;
        subscription.request(Integer.MAX_VALUE);
    }

    @Override
    public void onNext(final T t) {
        received.add(t);
    }

    @Override
    public void onError(final Throwable t) {
        if (t instanceof RuntimeException) {
            errors.add((RuntimeException) t);
        } else {
            errors.add(new RuntimeException("Unexpected exception", t));
        }
        onComplete();
    }

    @Override
    public void onComplete() {
        completed = true;
        subscription.cancel();
        latch.countDown();
    }

    /**
     * Get received elements
     *
     * @return the list of received elements
     */
    public List<T> getReceived() {
        return received;
    }

    /**
     * Get received elements.
```

```
     *
     * @return the list of receive elements
     */
    public List<T> get() {
        return await().getReceived();
    }

    /**
     * Await completion or error
     *
     * @return this
     */
    public SubscriberLatchWrapper<T> await() {
        subscription.request(Integer.MAX_VALUE);
        try {
            if (!latch.await(300, TimeUnit.SECONDS)) {
                throw new MongoTimeoutException("Publisher onComplete timed out for
 300 seconds");
            }
        } catch (InterruptedException e) {
            throw new MongoInterruptedException("Interrupted waiting for
 observeration", e);
        }
        if (!errors.isEmpty()) {
            throw errors.get(0);
        }
        return this;
    }

    public boolean getCompleted() {
        return this.completed;
    }

    public void close() {
        subscription.cancel();
        received.clear();
    }
}
```

C

次のコードは、C で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// Sample C code with core session

bool core_session(mongoc_client_session_t *client_session, mongoc_collection_t*
 collection, bson_t *selector, int64_t balance){
    bool r = true;
    bson_error_t error;
    bson_t *opts = bson_new();
    bson_t *update = BCON_NEW ("$set", "{", "balance", BCON_INT64 (balance), "}");

    // set read & write concern
    mongoc_read_concern_t *read_concern = mongoc_read_concern_new ();
    mongoc_write_concern_t *write_concern = mongoc_write_concern_new ();
    mongoc_transaction_opt_t *txn_opts = mongoc_transaction_opts_new ();

    mongoc_write_concern_set_w(write_concern, MONGOC_WRITE_CONCERN_W_MAJORITY);
    mongoc_read_concern_set_level(read_concern, MONGOC_READ_CONCERN_LEVEL_SNAPSHOT);
    mongoc_transaction_opts_set_write_concern (txn_opts, write_concern);
    mongoc_transaction_opts_set_read_concern (txn_opts, read_concern);

    mongoc_client_session_start_transaction (client_session, txn_opts, &error);
    mongoc_client_session_append (client_session, opts, &error);

    r = mongoc_collection_update_one (collection, selector, update, opts, NULL,
 &error);

    mongoc_client_session_commit_transaction (client_session, NULL, &error);
    bson_destroy (opts);
    mongoc_transaction_opts_destroy(txn_opts);
    mongoc_read_concern_destroy(read_concern);
    mongoc_write_concern_destroy(write_concern);
    bson_destroy (update);
    return r;
}

void test_core_money_transfer(mongoc_client_t* client, mongoc_collection_t*
 collection, int amount_to_transfer){

    bson_t reply;
    bool r = true;
    const bson_t *doc;
    bson_iter_t iter;
    bson_error_t error;
```

```
    // find query
    bson_t *alice_query = bson_new ();
    BSON_APPEND_UTF8(alice_query, "name", "Alice");

    bson_t *bob_query = bson_new ();
    BSON_APPEND_UTF8(bob_query, "name", "Bob");

    // create session
    // set causal consistency to false
    mongoc_session_opt_t *session_opts = mongoc_session_opts_new ();
    mongoc_session_opts_set_causal_consistency (session_opts, false);
    // start the session
    mongoc_client_session_t *client_session = mongoc_client_start_session (client,
session_opts, &error);

    // add session to options
    bson_t *opts = bson_new();
    mongoc_client_session_append (client_session, opts, &error);

    // deduct 500 from Alice
    // find account balance of Alice
    mongoc_cursor_t *cursor = mongoc_collection_find_with_opts (collection,
alice_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    int64_t alice_balance = (bson_iter_value (&iter))->value.v_int64;
    assert(alice_balance >= amount_to_transfer);
    int64_t new_alice_balance = alice_balance - amount_to_transfer;

    // core
    r = core_session (client_session, collection, alice_query, new_alice_balance);
    assert(r);

    // find account balance of Alice after transaction
    cursor = mongoc_collection_find_with_opts (collection, alice_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    alice_balance = (bson_iter_value (&iter))->value.v_int64;
    assert(alice_balance == new_alice_balance);
    assert(alice_balance == 500);

    // add 500 to Bob's balance
```

```
    // find account balance of Bob
    cursor = mongoc_collection_find_with_opts (collection, bob_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    int64_t bob_balance = (bson_iter_value (&iter))->value.v_int64;
    int64_t new_bob_balance = bob_balance + amount_to_transfer;

    //core
    r = core_session (client_session, collection, bob_query, new_bob_balance);
    assert(r);

    // find account balance of Bob after transaction
    cursor = mongoc_collection_find_with_opts (collection, bob_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    bob_balance = (bson_iter_value (&iter))->value.v_int64;
    assert(bob_balance == new_bob_balance);
    assert(bob_balance == 1500);

    // cleanup
    bson_destroy(alice_query);
    bson_destroy(bob_query);
    mongoc_client_session_destroy(client_session);
    bson_destroy(opts);
    mongoc_cursor_destroy(cursor);
    bson_destroy(doc);
}

int main(int argc, char* argv[]) {
    mongoc_init ();
    mongoc_client_t* client = mongoc_client_new (<connection uri>);
    bson_error_t error;

    // connect to bank db
    mongoc_database_t *database = mongoc_client_get_database (client, "bank");
    // access account collection
    mongoc_collection_t* collection = mongoc_client_get_collection(client, "bank",
 "account");
    // set amount to transfer
    int64_t amount_to_transfer = 500;
    // delete the collection if already existing
    mongoc_collection_drop(collection, &error);
```

```
    // open Alice account
    bson_t *alice_account = bson_new ();
    BSON_APPEND_UTF8(alice_account, "name", "Alice");
    BSON_APPEND_INT64(alice_account, "balance", 1000);

    // open Bob account
    bson_t *bob_account = bson_new ();
    BSON_APPEND_UTF8(bob_account, "name", "Bob");
    BSON_APPEND_INT64(bob_account, "balance", 1000);

    bool r = true;

    r = mongoc_collection_insert_one(collection, alice_account, NULL, NULL, &error);
    if (!r) {printf("Error encountered:%s", error.message);}
    r = mongoc_collection_insert_one(collection, bob_account, NULL, NULL, &error);
    if (!r) {printf("Error encountered:%s", error.message);}

    test_core_money_transfer(client, collection, amount_to_transfer);

}
```

Scala

次のコードは、Scala で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// Scala Core API
def transferMoneyWithRetry(sessionObservable: SingleObservable[ClientSession] ,
 database: MongoDatabase ): Unit = {
    val accountColl = database.getCollection("account")
    var amountToTransfer = 500

    var transactionObservable: Observable[ClientSession] =
 sessionObservable.map(clientSession => {
    clientSession.startTransaction()

    // deduct $500 from Alice's account
    var aliceBalance = accountColl.find(clientSession, Document("name" ->
 "Alice")).await().head.getInteger("balance")
    assert(aliceBalance >= amountToTransfer)
    var newAliceBalance = aliceBalance - amountToTransfer
```

```
    accountColl.updateOne(clientSession, Document("name" -> "Alice"),
 Document("$set" -> Document("balance" -> newAliceBalance))).await()
    aliceBalance = accountColl.find(clientSession, Document("name" ->
 "Alice")).await().head.getInteger("balance")
    assert(aliceBalance == newAliceBalance)

    // add $500 to Bob's account
    var bobBalance = accountColl.find(clientSession, Document("name" ->
 "Bob")).await().head.getInteger("balance")
    var newBobBalance = bobBalance + amountToTransfer
    accountColl.updateOne(clientSession, Document("name" -> "Bob"), Document("$set"
 -> Document("balance" -> newBobBalance))).await()
    bobBalance = accountColl.find(clientSession, Document("name" ->
 "Bob")).await().head.getInteger("balance")
    assert(bobBalance == newBobBalance)

    clientSession
    })

    transactionObservable.flatMap(clientSession =>
 clientSession.commitTransaction()).await()
}

def doTransactionWithRetry(): Unit = {
    val client: MongoClient = MongoClientWrapper.getMongoClient()
    val database: MongoDatabase = client.getDatabase("bank")
    val accountColl = database.getCollection("account")
    accountColl.drop().await()

    val sessionOptions =
 ClientSessionOptions.builder().causallyConsistent(false).build()
    var  sessionObservable: SingleObservable[ClientSession] =
 client.startSession(sessionOptions)
    accountColl.insertOne(Document("name" -> "Alice", "balance" -> 1000)).await()
    accountColl.insertOne(Document("name" -> "Bob", "balance" -> 1000)).await()

    var retry = true
    while (retry) {
        try {
        transferMoneyWithRetry(sessionObservable, database)
        println("transaction committed")
        retry = false
        }
        catch {
```

```
        case e: MongoException if
 e.hasErrorLabel(MongoException.TRANSIENT_TRANSACTION_ERROR_LABEL) => {
            println("retrying transaction")
        }
        case other: Throwable => {
            println("transaction failed")
            retry = false
            throw other

        }
        }
    }

    // check results outside of transaction
    assert(accountColl.find(Document("name" ->
 "Alice")).results().head.getInteger("balance") == 500)
    assert(accountColl.find(Document("name" ->
 "Bob")).results().head.getInteger("balance") == 1500)

    accountColl.drop().await()

}
```

Python

次のコードは、Python で Amazon DocumentDB トランザクション API を使用する方法を示しています。

```
// Sample Python code with Core api

import pymongo

client = pymongo.MongoClient(<connection_string>)
rc_snapshot = pymongo.read_concern.ReadConcern('snapshot')
wc_majority = pymongo.write_concern.WriteConcern('majority')

# To start, drop and create an account collection and insert balances for both Alice
 and Bob
collection = client.get_database("bank").get_collection("account")
collection.drop()
collection.insert_one({"_id": 1, "name": "Alice", "balance": 1000})
collection.insert_one({"_id": 2, "name": "Bob", "balance": 1000})
```

```
amount_to_transfer = 500

# deduct 500 from Alice's account
alice_balance = collection.find_one({"name": "Alice"}).get("balance")
assert alice_balance >= amount_to_transfer
new_alice_balance = alice_balance - amount_to_transfer

with client.start_session({'causalConsistency':False}) as session:
    session.start_transaction(read_concern=rc_snapshot, write_concern=wc_majority)
    collection.update_one({"name": "Alice"}, {'$set': {"balance":
 new_alice_balance}}, session=session)
    session.commit_transaction()

updated_alice_balance = collection.find_one({"name": "Alice"}).get("balance")
assert updated_alice_balance == new_alice_balance

# add 500 to Bob's account
bob_balance = collection.find_one({"name": "Bob"}).get("balance")
assert bob_balance >= amount_to_transfer
new_bob_balance = bob_balance + amount_to_transfer

with client.start_session({'causalConsistency':False}) as session:
    session.start_transaction(read_concern=rc_snapshot, write_concern=wc_majority)
    collection.update_one({"name": "Bob"}, {'$set': {"balance": new_bob_balance}},
 session=session)
    session.commit_transaction()

updated_bob_balance = collection.find_one({"name": "Bob"}).get("balance")
assert updated_bob_balance == new_bob_balance
```

サポートされているコマンド

コマンド	サポート
abortTransaction	あり
commitTransaction	あり
endSessions	あり
killSession	あり

コマンド	サポート
killAllSession	はい
killAllSessionsByPattern	いいえ
refreshSessions	なし
startSession	あり

サポート対象外の機能

方法	ステージまたはコマンド
db.collection.aggregate()	$collStats $currentOp $indexStats $listSessions $out
db.collection.count() db.collection.countDocuments()	$where $near $nearSphere
db.collection.insert()	insert は、既存のコレクションに対して実行されない場合はサポートされません。このメソッドは、既存のコレクションを対象とする場合にサポートされます。

セッション

MongoDB セッションは、再試行可能な書き込み、因果一貫性、トランザクション、およびデータベース間の操作の管理をサポートするために使用されるフレームワークです。セッションが作成されると、論理セッション識別子 (LSID) がクライアントによって生成され、サーバーにコマンドを送信するときに、そのセッション内のすべてのオペレーションにタグを付けるために使用されます。

Amazon DocumentDB は、トランザクションを有効にするためのセッションの使用をサポートしていますが、因果整合性や再試行可能な書き込みはサポートしていません。

Amazon DocumentDB 内でトランザクションを利用する場合、トランザクションはセッション内から `session.startTransaction()` API とセッションは、一度に 1 つのトランザクションをサポートします。同様に、トランザクションは commit (`session.commitTransaction()`) または中止 (`session.abortTransaction()`) API。

因果整合性

因果整合性により、単一のクライアントセッション内でクライアントが書き込み後の読み取り整合性、モノアトミックな読み取り/書き込みが読み取りに従うことが保証され、これらの保証はプライマリだけでなく、クラスタ内のすべてのインスタンスに適用されます。Amazon DocumentDB は因果一貫性をサポートしておらず、次のステートメントではエラーが発生します。

```
var mySession = db.getMongo().startSession();
var mySessionObject = mySession.getDatabase('test').getCollection('account');

mySessionObject.updateOne({"_id": 2}, {"$inc": {"balance": 400}});
//Result:{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

mySessionObject.find()
//Error: error: {
//         "ok" : 0,
//         "code" : 303,
//         "errmsg" : "Feature not supported: 'causal consistency'",
//         "operationTime" : Timestamp(1603461817, 493214)
//}

mySession.endSession()
```

セッション内で因果一貫性を無効にできます。そうすることで、セッションフレームワークを利用できますが、リードの因果一貫性の保証は提供されません。Amazon DocumentDB を使用する場合、

プライマリからの読み取りは書き込み後の読み取り整合性になり、レプリカインスタンスからの読み取りは最終的に一貫します。トランザクションは、セッションを利用するための主要なユースケースです。

```
var mySession = db.getMongo().startSession({causalConsistency: false});
var mySessionObject = mySession.getDatabase('test').getCollection('account');

mySessionObject.updateOne({"_id": 2}, {"$inc": {"balance": 400}});
//Result:{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

mySessionObject.find()
//{ "_id" : 1, "name" : "Bob", "balance" : 100 }
//{ "_id" : 2, "name" : "Alice", "balance" : 1700 }
```

再試行可能な書き込み

再試行可能な書き込みは、ネットワークエラーが発生したとき、またはクライアントがプライマリを検出できない場合に、クライアントが書き込み操作を再試行する機能です。Amazon DocumentDB では、再試行可能な書き込みはサポートされていないため、無効にする必要があります。接続文字列のコマンド (`retryWrites=false`) で無効にできます。

> **Note**
>
> 例外: mongo シェル を使用している場合は、どのコード文字列にも `retryWrites=false` コマンドを含めないでください。デフォルトでは、再試行可能な書き込みは無効になっています。`retryWrites=false` を含めると、通常の読み取りコマンドでは失敗する可能性があります。

トランザクションエラー

トランザクションを使用する場合、トランザクション番号が進行中のトランザクションと一致しないことを示すエラーが発生するシナリオがあります。

このエラーは、少なくとも 2 つの異なるシナリオで生成できます。

- 1 分間のトランザクションタイムアウト後。
- インスタンスの再起動後 (パッチ適用、クラッシュリカバリなど)、トランザクションが正常にコミットされた場合でも、このエラーを受け取る可能性があります。インスタンスの再起動中、デー

タベースは正常に完了したトランザクションと中止されたトランザクションの違いを判断できません。つまり、トランザクションの完了状態があいまいとなります。

このエラーを処理する最善の方法は、トランザクションの更新を冪等化することです。例えば、$set インクリメント/デクリメント操作の代わりにミューテーター。以下を参照してください。

```
{ "ok" : 0,
"operationTime" : Timestamp(1603938167, 1),
"code" : 251,
"errmsg" : "Given transaction number 1 does not match any in-progress transactions."
}
```

Amazon DocumentDB のベストプラクティス

Amazon DocumentDB(MongoDB との互換性あり)を使用するためのベストプラクティスを説明します。新しいベストプラクティスが確認されると、このセクションは更新されます。

トピック

- 基本運用ガイドライン
- インスタンスのサイズ指定
- インデックスの使用
- セキュリティに関するベストプラクティス
- コスト最適化
- メトリクスを使用したパフォーマンスの問題の特定
- TTL および時系列ワークロード
- 移行
- クラスターパラメータグループの使用
- 集約パイプラインクエリ
- batchInsert および batchUpdate

基本運用ガイドライン

Amazon DocumentDB の使用の際に順守すべき基本的な運用ガイドラインを次に示します。Amazon DocumentDB のサービスレベルアグリーメント (SLA) では、次のガイドラインに順守する必要があります。

- 2 つ以上の Amazon DocumentDB インスタンスで構成されるクラスターを 2 つの AWS アベイラビリティーゾーンにデプロイします。本番ワークロードの場合、3 つ以上の Amazon DocumentDB インスタンスで構成されるクラスターを 3 つの アベイラビリティーゾーンにデプロイすることを推奨します。
- 指定されたサービスの制限内でサービスを使用します。詳細については、「Amazon DocumentDB のクォータと制限事項」を参照してください。
- メモリ、CPU、接続、およびストレージの使用状況をモニタリングします。Amazon CloudWatch では、使用パターンが変更されたり、デプロイの最大容量に近づいたりすると、通知するように設定できます。この通知は、システムのパフォーマンスと可用性を維持するために役立ちます。

- 最大ストレージ容量に近づいたら、インスタンスをスケールアップする。アプリケーションからの予期しない需要増加に対応できる十分なコンピューティングリソース (RAM、CPU など) を使用してインスタンスをプロビジョニングする必要があります。
- 復旧ポイントの目標に合わせて、バックアップ保持期間を設定します。
- クラスターのフェイルオーバーをテストすることで、そのプロセスでユースケースにかかる時間を把握します。詳細については、「Amazon DocumentDB フェイルオーバー」を参照してください。
- Amazon DocumentDB クラスターをクラスターエンドポイント (「Amazon DocumentDB エンドポイント」を参照) にレプリカセットモード (「レプリカセットとしての Amazon DocumentDB に接続する」を参照) で接続し、アプリケーションへのフェイルオーバーによる影響を最小限に抑えます。
- アプリケーションの読み込み整合性要件に合わせて読み取り性能を最大化する、ドライバー読み込み設定を選択します。`secondaryPreferred` 読み込み設定を使用すると、レプリカの読み取りを有効にし、プライマリインスタンスを解放してより多くの作業を行うことができます。詳細については、「読み込み設定のオプション」を参照してください。
- ネットワークエラーやデータベースエラーが発生した場合にも回復できるようにアプリケーションを設計します。ドライバーのエラーメカニズムを使用して、一時エラーと永続エラーを区別します。適切な場合には、エクスポネンシャルバックオフメカニズムを使用して一時エラーを再試行します。再試行ロジックを実装する場合は、アプリケーションがデータの整合性を考慮するようにします。
- すべての本稼働用クラスターや貴重なデータがあるすべてのクラスターに対して、クラスターの削除保護を有効にします。Amazon DocumentDB 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します。でリソースをデプロイする場合は AWS CloudFormation、終了保護を有効にします。詳細については、「終了保護と削除保護」を参照してください。
- Amazon DocumentDB クラスターを作成する場合、 `--engine-version`はオプションのパラメータであり、デフォルトで最新のメジャーエンジンバージョンになります。現在のメジャーエンジンバージョンは 5.0.0 です。新しいメジャーエンジンバージョンがリリースされると、 のデフォルトのエンジンバージョン`--engine-version`が更新され、最後のメジャーエンジンバージョンが反映されます。そのため、本稼働ワークロード、特にスクリプト、オートメーション、または AWS CloudFormation テンプレートに依存するワークロードの場合は、目的のメジャーバージョン`--engine-version`に を明示的に指定することをお勧めします。

インスタンスのサイズ指定

Amazon DocumentDB でインスタンスサイズを選択する最も重要な側面の 1 つは、キャッシュの RAM 容量です。Amazon DocumentDB は RAM の 3 分の 1 を独自のサービス用に使用しており、キャッシュに使用できるのはインスタンス RAM の 3 分の 2 のみです。そのため、Amazon DocumentDB のパフォーマンスのベストプラクティスとして、ワーキングセット (データやインデックスなど) をメモリに収めるのに十分な RAM を備えたインスタンスタイプを選択することを推奨します。インスタンスを適切なサイズに設定すると、全体的なパフォーマンスを最適化し、I/O コストを最小限に抑えることができます。

アプリケーションのワーキングセットがメモリに収まるかどうかを判断するには、Amazon CloudWatch を使用して、負荷がかかっているクラスター内のインスタンスごとに `BufferCacheHitRatio` をモニタリングします。

CloudWatch の `BufferCacheHitRatio` メトリクスは、インスタンスのメモリキャッシュから提供されたデータおよびインデックスの (ストレージボリュームに対する) 割合を測定します。一般的に、ワーキングセットメモリからのデータの読み取りは、ストレージボリュームからの読み取りよりも高速でコスト効率が高いため、`BufferCacheHitRatio` をできるだけ高い値に維持します。`BufferCacheHitRatio` はできるだけ 100% に近づけることが理想ですが、達成可能な最大値はアプリケーションのアクセスパターンやパフォーマンス要件によって異なります。可能な限り高い `BufferCacheHitRatio` を維持するために、インデックスとワーキングデータセットをメモリに収められるだけの十分な RAM を使用して、クラスター内のインスタンスをプロビジョニングすることを推奨します。

インデックスがメモリに収まらない場合は、`BufferCacheHitRatio` 値が小さくなります。ディスクからの継続的な読み取りは、追加の I/O コストが発生するため、メモリからの読み取りよりパフォーマンスが優れているとは言えません。`BufferCacheHitRatio` の比率が予想よりも小さい場合は、クラスターのインスタンスサイズをスケールアップして、ワーキングセットデータがメモリに収まるように RAM を大きくします。インスタンスクラスをスケールアップした結果 `BufferCacheHitRatio` が大幅に増加した場合、アプリケーションのワーキングセットがメモリに収まりません。スケーリングオペレーション後に `BufferCacheHitRatio` が劇的に増加しなくなるまで、継続してスケールアップします。インスタンスのメトリクスのモニタリングについては、「Amazon DocumentDB のメトリクス」を参照してください。

ワークロードやレイテンシーの要件に応じて、定常状態の使用中にアプリケーションの `BufferCacheHitRatio` 値が高くなることは許容されますが、コレクション全体をスキャンする必要がある分析クエリがインスタンスで実行されるため、定期的に `BufferCacheHitRatio` の急減が発生します。これらの定期的な `BufferCacheHitRatio` の急減は、後続のクエリでストレージボ

リュームからバッファキャッシュにワーキングセットデータを再入力する場合に、レイテンシーの増加として現れることがあります。ワークロードを本稼働用環境にデプロイする前にパフォーマンス特性と **BufferCacheHitRatio** を理解するために、まず代表的な本番稼働用ワークロードを使用して本番稼働前の環境でワークロードをテストすることを推奨します。

BufferCacheHitRatio はインスタンス固有のメトリクスであるため、プライマリインスタンスとレプリカインスタンス間での読み取りの分散方法に応じて、同じクラスター内のインスタンスが異なる BufferCacheHitRatio 値を持つ場合があります。運用ワークロードが、分析クエリの実行後にワーキングセットキャッシュの再入力に伴う定期的なレイテンシーの増加に対応できない場合は、分析クエリのバッファキャッシュから通常のワークロードのバッファキャッシュを分離する必要があります。BufferCacheHitRatio の完全な分離を達成するには、運用クエリをプライマリインスタンスに転送し、分析クエリをレプリカインスタンスにのみ転送します。また、部分的な分離を達成するには、分析クエリを特定のレプリカインスタンスに転送できます。ただし、このレプリカでは通常のクエリも一部実行されるため、影響を受ける可能性があります。

適切な BufferCacheHitRatio 値は、ユースケースとアプリケーションの要件によって異なります。このメトリクスには 1 つの最適値や最小値はありません。コストとパフォーマンスの観点から、BufferCacheHitRatio 値の一時的な低下に伴うトレードオフを許容できるかどうかはお客様の判断次第です。

インデックスの使用

インデックスの構築

Amazon DocumentDB にデータをインポートする場合は、大きなデータセットをインポートする前にインデックスを作成する必要があります。Amazon DocumentDB インデックスツール を使用すると、実行中の MongoDB インスタンスや mongodump ディレクトリからインデックスを抽出し、Amazon DocumentDB クラスターでそれらのインデックスを作成できます。移行の詳細については、Amazon DocumentDB への移行を参照してください。

インデックスの選択性

インデックスの作成は、重複する値の数がコレクション内のドキュメントの総数の 1% 未満のフィールドに制限することを推奨します。例えば、コレクションに 100,000 個のドキュメントが含まれている場合、同じ値が発生する回数が 1000 回以下のフィールドにのみインデックスを作成します。

一意の値が多数ある (つまり、濃度が高い) インデックスを選択すると、フィルター処理によって返されるドキュメントの数が少なくなるため、インデックススキャン中のパフォーマンスが向上しま

す。高濃度インデックスの例は一意のインデックスです。これにより、等式の述語が最大で 1 つのドキュメントを返すことが保証されます。低濃度の例としては、ブール型フィールドのインデックスと、曜日別のインデックスなどがあります。パフォーマンスが低下するため、低濃度のインデックスは、データベースのクエリオプティマイザによって選択される可能性が低くなります。同時に、低濃度のインデックスは、ディスク領域や I/O などのリソースを消費し続けます。経験則として、標準値の頻度がコレクション全体のサイズの 1% 以下のフィールドのインデックスをターゲットにしてください。

さらに、よくフィルターとして使用されるフィールドに対してのみインデックスを作成し、未使用のインデックスを定期的に検索することを推奨します。詳細については、「インデックスの使用状況を分析し、未使用のインデックスを特定する方法」を参照してください。

インデックスがデータの書き込みに与える影響

インデックスを使用すると、コレクション内のすべてのドキュメントをスキャンする必要がなくなるためクエリのパフォーマンスが向上しますが、このメリットにはトレードオフがあります。コレクションのインデックスごとに、ドキュメントが挿入、更新、または削除されるたびに、データベースはコレクションを更新し、コレクションの各インデックスにフィールドを書き込む必要があります。例えば、コレクションに 9 つのインデックスがある場合、データベースはクライアントへのオペレーションを承認する前に 10 回の書き込みを実行する必要があります。したがって、インデックスが増えるたびに、書き込みレイテンシーと I/O が多くなり、全体的なストレージ使用率が高まります。

クラスターインスタンスは、すべてのワーキングセットメモリを保持するために適切なサイズにする必要があります。これにより、インデックスページをストレージボリュームから継続的に読み取る必要がなくなるため、パフォーマンスに悪影響を及ぼし、I/O コストが高くなります。詳細については、「インスタンスのサイズ指定」を参照してください。

最高のパフォーマンスを得るには、コレクション内のインデックスの数を最小限に抑え、一般的なクエリのパフォーマンスを高めるために必要なインデックスのみを追加します。ワークロードはさまざまですが、ガイドラインとして、コレクションあたりのインデックス数を 5 つ以下に抑えることを推奨します。

欠落しているインデックスの識別

ベストプラクティスとして、欠落しているインデックスを定期的に特定することを推奨します。詳細については、「欠落しているインデックスを特定する方法」を参照してください。

使用されていないインデックスの識別

ベストプラクティスとして、未使用のインデックスを定期的に特定し削除することを推奨します。詳細については、「インデックスの使用状況を分析し、未使用のインデックスを特定する方法」を参照してください。

セキュリティに関するベストプラクティス

セキュリティのベストプラクティスとして、 AWS Identity and Access Management (IAM) アカウントを使用して Amazon DocumentDB API オペレーション、特に Amazon DocumentDB リソースを作成、変更、削除するオペレーションへのアクセスを制御する必要があります。そのようなリソースには、クラスター、セキュリティグループ、およびパラメータグループなどがあります。また、IAMを使用して、クラスターのバックアップや復元など、一般的な管理操作を実行するアクションも制御します。IAM ロールを作成する際には、最小特権の原則を採択します。

- ロールベースのアクセスコントロールを使用して、最小特権を適用します。
- Amazon DocumentDB のリソースを管理する各ユーザーにそれぞれ IAM アカウントを割り当てます。Amazon DocumentDB リソースの管理に AWS アカウント ルートユーザーを使用しないでください。お客様を含めて全員に IAM ユーザーを作成します。
- それぞれの職務遂行に最低限必要な一連のアクセス権限を各 IAM ユーザーに付与します。
- IAM グループを使用して、複数のユーザーのアクセス許可を効果的に管理します。IAM の詳細については、IAM ユーザーガイドを参照してください。IAM のベストプラクティスの詳細については、「IAM のベストプラクティス」を参照してください。
- IAM 認証情報のローテーションを定期的に行います。
- Amazon DocumentDB の AWS シークレットを自動的にローテーションするように Secrets Manager を設定します。詳細については、AWS Secrets Manager ユーザーガイドの「Secrets Manager シークレットのローテーション」およびAmazon DocumentDB のシークレットのローテーション」を参照してください。 AWS
- それぞれの職務遂行に最低限必要な一連のアクセス権限を各 Amazon DocumentDB ユーザーに付与します。詳細については、「ロールベースのアクセス制御 (RBAC) を使用したデータベースへのアクセス」を参照してください。
- Transport Layer Security (TLS) を使用して、転送中のデータを暗号化し AWS KMS 、保管中のデータを暗号化します。

コスト最適化

次のベストプラクティスは、Amazon DocumentDB を使用する際のコストの管理と最小化に役立ちます。料金に関する情報は、「Amazon DocumentDB (MongoDB との互換性あり) の料金 と Amazon DocumentDB (MongoDB との互換性あり) のよくある質問」を参照してください。

- 該当月の予想請求額の 50% と 75% のしきい値で請求アラートを作成します。請求アラートの作成の詳細については、「請求アラームの作成」を参照してください。
- Amazon DocumentDB のアーキテクチャは、ストレージとコンピューティングを分離するため、単一インスタンスクラスターでも高い耐久性を備えています。クラスターストレージボリュームは、3 つのアベイラビリティーゾーンにわたって 6 つの方法でデータをレプリケートすることにより、クラスター内のインスタンス数に関係なく、きわめて高い耐久性を実現します。一般的な本稼働クラスターには、高可用性を実現するために 3 つ以上のインスタンスがあります。ただし、高可用性を必要としない場合は、単一インスタンス開発クラスターを使用してコストを最適化できます。
- 開発およびテストシナリオでは、不要になったらクラスターを停止し、開発が再開されたらクラスターを起動します。詳細については、「Amazon DocumentDB クラスターの停止と開始」を参照してください。
- TTL ストリームと変更ストリームでは、どちらもデータの書き込み、読み取り、削除時に I/O が発生します。これらの機能を有効にしているが、アプリケーションで使用していない場合、機能を無効にすると、コスト削減に役立ちます。

メトリクスを使用したパフォーマンスの問題の特定

トピック

- パフォーマンスメトリクスの表示
- CloudWatch アラームの設定
- パフォーマンスメトリクスの評価
- CloudWatch メトリクスで Amazon DocumentDB インスタンスの使用状況を評価する
- クエリのチューニング

リソース不足やその他の一般的なボトルネックに起因するパフォーマンスの問題を特定するために、Amazon DocumentDB クラスターに適用されるメトリクスをモニタリングできます。

パフォーマンスメトリクスの表示

さまざまな時間範囲の平均値、最大値、最小値を表示するには、パフォーマンスメトリクスを定期的に監視します。これは、いつパフォーマンスが低下しているかを特定するうえで有効です。また、特定のメトリクスしきい値に対して Amazon CloudWatch アラームを設定すると、しきい値に達した場合に警告されるように設定できます。

パフォーマンスの問題を解決するために重要なのは、システムのベースラインパフォーマンスを理解することです。新しいクラスターをセットアップし、一般的なワークロードで実行したら、すべてのパフォーマンスメトリクスの平均値、最大値、最小値をさまざまな間隔 (例: 1 時間、24 時間、1 週間、2 週間) で取得します。これにより、正常な状態を把握することができます。それにより、オペレーションのピークおよびオフピークの時間帯を比較して、得られた情報から、いつパフォーマンスが標準レベルを下回っているかを特定できます。

パフォーマンスメトリクスは、 AWS Management Console または を使用して表示できます AWS CLI。詳細については、「CloudWatch データの表示」を参照してください。

CloudWatch アラームの設定

CloudWatch アラームを設定するには、CloudWatch ユーザーガイドの「Amazon CloudWatch アラームの使用」を参照してください。

パフォーマンスメトリクスの評価

インスタンスには、さまざまなカテゴリのメトリクスがあります。許容値を決定する方法は、メトリクスによって異なります。

CPU

- CPU使用率 - コンピュータの処理能力のうち、使用されている容量の割合。

メモリ

- 空きメモリ — インスタンスで使用可能な RAM の容量。
- スワップ領域 — インスタンスが使用しているスワップスペースの量 (メガバイト単位)。

入力/出力オペレーション

- 読み込み IOPS、書き込み IOPS — 1 秒あたりのディスク読み取りまたは書き込み操作の平均数。

- 読み取りレイテンシー、書き込みレイテンシ — 1 秒あたりのディスク読み取りまたは書き込み操作の平均時間 (ミリ秒単位)。
- 読み取りスループット、書き込みスループット — 1 秒あたりのディスク読み取りまたは書き込みデータの平均量 (メガバイト単位)。
- ディスクキューの深さ — ディスクへの書き込みまたはディスクからの読み取りを待っている I/O オペレーションの数。

ネットワークトラフィック

- ネットワーク受信スループット、ネットワーク送信スループット — インスタンスに対する送信または受信ネットワークトラフィックの1 秒あたりのレート (メガバイト単位)。

データベース接続

- DB接続数 — インスタンスに接続されたクライアントセッションの数。

一般的に、パフォーマンスメトリクスの許容値は、ベースラインがどのようになっているか、アプリケーションによって何が実行されているかによって異なります。ベースラインからの一貫した差異またはトレンドになっている差異を調べます。

メトリクスのタイプごとの推奨事項とアドバイスは次のとおりです。

- CPU の高消費量 - CPU消費量が高い値になっていても、アプリケーションの目標 (スループットや並行処理など) に沿った想定値であれば、妥当である場合があります。CPU 使用率が一貫して 80% を超える場合は、インスタンスのスケールアップを検討してください。
- RAM の高消費量 — `FreeableMemory` メトリクスがインスタンスメモリの合計の 10% を頻繁に下回る場合は、インスタンスのスケールアップを検討してください。DocumentDB インスタンスで高いメモリプレッシャーが発生している場合に何が起こるかについての詳細は、「Amazon DocumentDB リソースガバナンス」を参照してください。
- スワップの使用量 — このメトリクスは 0 またはほぼ 0 のままにする必要があります。スワップの使用量が多い場合は、インスタンスのスケールアップを検討してください。
- ネットワークトラフィック — ネットワークトラフィックについてシステム管理者に問い合わせて、ドメインネットワークとインターネット接続の想定スループットを把握してください。スループットが一貫して想定よりも低い場合は、ネットワークトラフィックを調べます。
- データベース接続数 — ユーザー接続数が多いことが、インスタンスのパフォーマンスと応答時間の低下とに関連している場合は、データベース接続数の制限を検討してください。インスタンスの

最適なユーザー接続数は、インスタンスのクラスと実行中のオペレーションの複雑さによって異なります。パフォーマンスメトリクスに問題がある場合、パフォーマンスを向上させるためにまず実行できることの 1 つは、使用頻度とコストの最も高いクエリをチューニングして、システムリソースへの負荷が下がるかどうかを確認することです。

クエリをチューニングしても問題が解決しない場合は、その問題に関連するリソース(CPU、RAM、ディスク容量、ネットワーク帯域幅、I/O 容量)を増加できる Amazon DocumentDB インスタンスクラスにアップグレードすることを検討してください。

CloudWatch メトリクスで Amazon DocumentDB インスタンスの使用状況を評価する

CloudWatch メトリクスを使用して、インスタンスのスループットを監視し、インスタンスクラスがアプリケーションに十分なリソースを提供しているかどうかを確認できます。インスタンスクラスの制限については、インスタンス制限 を表示し、お使いのインスタンスクラスの仕様を確認して、ネットワークパフォーマンスをチェックしてください。

インスタンスの使用量がインスタンスクラスの限度に近い場合、パフォーマンスが低下し始める可能性があります。CloudWatch メトリクスによってこの状況を確認できるので、より大きなインスタンスクラスへの手動スケールアップを計画できます。

以下の CloudWatch メトリクス値を組み合わせて、インスタンスクラスの限界に近づいているかどうかを確認します。

- NetworkThroughput - Amazon DocumentDB クラスター内の各インスタンスについて、各クライアントによって送受信されたネットワークスループットの量。このスループット値には、クラスター内のインスタンスとクラスターストレージボリューム間のネットワークトラフィックは含まれません。
- StorageNetworkThroughput — Amazon DocumentDB クラスター内の各インスタンスによって Amazon DocumentDB クラスターストレージボリュームに送受信されるネットワークスループットの量。

NetworkThroughput を StorageNetworkThroughput に加えると、Amazon DocumentDB クラスター内の各インスタンスと Amazon DocumentDB クラスターストレージボリュームとの間で送受信されたネットワークスループット量が得られます。インスタンスのインスタンスクラス限界は、この 2 つのメトリクスの合計よりも大きい必要があります。

以下のメトリクスを使用して、送受信時のクライアントアプリケーションからのネットワークトラフィックの詳細を確認できます。

- NetworkReceiveThroughput – Amazon DocumentDB クラスター内の各インスタンスがクライアントから受信したネットワークスループットの量。クラスター内のインスタンスとクラスターストレージボリューム間のネットワークトラフィックは、このスループットに含まれません。
- NetworkTransmitThroughput – Amazon DocumentDB クラスター内の各インスタンスが各クライアントに送信したネットワークスループットの量。クラスター内のインスタンスとクラスターストレージボリューム間のネットワークトラフィックは、このスループットに含まれません。
- StorageNetworkReceiveThroughput – Amazon DocumentDB クラスター内の各インスタンスがクラスターストレージボリュームから受信したネットワークスループットの量。
- StorageNetworkTransmitThroughput – Amazon DocumentDB クラスター内の各インスタンスがクラスターストレージボリュームに送信したネットワークスループットの量。

これらすべてのメトリクスを合計して、ネットワーク使用量とインスタンスクラスの限界との比較を評価します。インスタンスクラス限界は、これらのメトリクスの合計よりも大きい必要があります。

ネットワーク限度とインスタンスによる CPU 使用率は相互に関係しています。ネットワークのスループットが増加すると、CPU 使用率も増加します。CPU とネットワークの使用状況を監視すると、リソースがどのくらい枯渇しているのか、またなぜ枯渇しているのかについての情報が得られます。

ネットワークの使用量を最小限に抑えるには、次のことを検討してください。

- より大きなインスタンスクラスを使用します。
- 書き込みリクエストをバッチに分割して、トランザクション全体を減らします。
- 読み取り専用ワークロードを読み取り専用インスタンスに転送します。
- 未使用のインデックスをすべて削除します。

クエリのチューニング

クラスターのパフォーマンスを向上させるには、大量のリソースを消費する使用頻度の最も高いクエリをチューニングして、実行コストを下げることを推奨します。

プロファイラーを使用して(Amazon DocumentDB オペレーションのプロファイリング を参照)、クラスターで実行されたオペレーションの実行時間と詳細を記録できます。プロファイラーは、クラス

ターで最も遅いオペレーションをモニタリングし、個々のクエリパフォーマンスとクラスター全体のパフォーマンスを向上させるのに役立ちます。

explain コマンドを使用して、特定のクエリのクエリプランを分析する方法を学習することもできます。この情報を使用してクエリまたは基礎となるコレクションを変更して、クエリのパフォーマンスを向上させます(例えば、インデックスの追加)。

TTL および時系列ワークロード

TTL インデックスの有効期限が切れたことによるドキュメントの削除は、ベストエフォートプロセスです。ドキュメントが特定の期間内に削除されることは保証されません。インスタンスサイズ、インスタンスリソース使用率、ドキュメントサイズ、全体的なスループット、インデックスの数、およびインデックスとワーキングセットがメモリに収まるかどうかなどの要因はすべて、期限切れのドキュメントが TTL プロセスによって削除されるタイミングに影響します。

TTL モニターがドキュメントを削除すると、削除ごとに I/O コストが発生するため、請求が増加します。スループットおよび TTL 削除のレートが増加した場合、I/O 使用量が増加するため、請求の増加が予想されます。ただし、TTL インデックスを作成してドキュメントを削除せず、時間に基づいてドキュメントをコレクションにセグメント化し、不要になったときにそのコレクションを削除する場合、I/O コストは発生しません。これは、TTL インデックスを使用するよりもコスト効率が大幅に向上します。

時系列ワークロードの場合、TTL インデックスの代わりにローリングコレクションを作成することを検討できます。ローリングコレクションは、データを削除し、I/O 負荷を小さくする優れた方法となるためです。大規模なコレクション (特に 1 TB 以上のコレクション) を所持する場合や TTL 削除の I/O コストが懸念される場合は、時間に基づいてドキュメントをコレクションに分割し、ドキュメントが不要になった時点でコレクションを削除することを推奨します。データ取り込みレートに応じて、1 日または週に 1 つのコレクションを作成できます。要件はアプリケーションによって異なりますが、経験則として、いくつかの大きいコレクションではなく小さいコレクションにすることをお勧めします。これらのコレクションを削除しても I/O コストは発生せず、TTL インデックスを使用するよりもコスト効率が向上します。

移行

ベストプラクティスとして、Amazon DocumentDB にデータを移行する場合、まず Amazon DocumentDB でインデックスを作成してからデータを移行することを推奨します。最初にインデックスを作成すると、全体の時間が短縮され、移行速度が向上します。これを行うには、Amazon

DocumentDB インデックスツール を使用します。移行の詳細については、Amazon DocumentDB 移行ガイド を参照してください。

また、本番データベースを移行する前に、機能、パフォーマンス、オペレーション、コストなどを考慮して、Amazon DocumentDB でアプリケーションを完全にテストすることをお勧めします。

クラスターパラメータグループの使用

クラスターパラメータグループを変更した場合は、その変更をテストクラスターでテストしてから本稼働クラスターに適用することを推奨します。クラスターのバックアップの詳細については、「Amazon DocumentDB でのバックアップと復元」を参照してください。

集約パイプラインクエリ

複数のステージを持つ集約パイプラインクエリを作成し、クエリ内のデータのサブセットのみを評価する場合は、`$match` ステージをパイプラインの最初のステージまたは先頭部分として使用します。`$match` を最初に使用すると、集約パイプラインクエリ内の後続ステージで処理するドキュメントの数が減り、クエリのパフォーマンスが向上します。

`batchInsert` および `batchUpdate`

`batchInsert` または `batchUpdate` の同時実行オペレーションを高率で実行しているときに、プライマリインスタンスで `FreeableMemory` (CloudWatch のメトリクス) の量がゼロになった場合は、バッチ挿入ワークロードやバッチ更新ワークロードの同時実行数を減らすことができます。ワークロードの同時実行数を減らせない場合は、インスタンスサイズを大きくして `FreeableMemory` の量を増やすことができます。

機能の違い: Amazon DocumentDB と MongoDB

Amazon DocumentDB (MongoDB 互換性) と MongoDB の機能上の違いを以下に示します。

トピック

- Amazon DocumentDB の機能上の利点
- 機能的な違いの更新
- MongoDB との機能的な違い

Amazon DocumentDB の機能上の利点

暗示的なトランザクション

Amazon DocumentDB では、すべての CRUD ステートメント (`findAndModify`、`update`、`insert`、`delete`) は、複数のドキュメントを変更するオペレーションにおいてもアトミック性および整合性を保証します。Amazon DocumentDB 4.0 の起動により、マルチステートメントおよびマルチコレクションオペレーションの ACID プロパティを提供する明示的なトランザクションがサポートされるようになりました。Amazon DocumentDB でのトランザクションの使用の詳細については、「Amazon DocumentDB でのトランザクション」を参照してください。

アトミック性および整合性の両方の動作を満たす複数のドキュメントを変更する Amazon DocumentDB のオペレーションの例を次に示します。

```
db.miles.update(
    { "credit_card": { $eq: true } },
    { $mul: { "flight_miles.$[]": NumberInt(2) } },
    { multi: true }
)
```

```
db.miles.updateMany(
    { "credit_card": { $eq: true } },
    { $mul: { "flight_miles.$[]": NumberInt(2) } }
)
```

```
db.runCommand({
  update: "miles",
  updates: [
    {
      q: { "credit_card": { $eq: true } },
      u: { $mul: { "flight_miles.$[]": NumberInt(2) } },
      multi: true
    }
  ]
})
```

```
db.products.deleteMany({
  "cost": { $gt: 30.00 }
})
```

```
db.runCommand({
  delete: "products",
  deletes: [{ q: { "cost": { $gt: 30.00 } }, limit: 0 }]
})
```

updateMany や deleteMany などの一括オペレーションを構成する個々のオペレーションはアトミックですが、一括オペレーション全体はアトミックではありません。たとえば、個々の挿入オペレーションがエラーなしで正常に実行された場合、insertMany オペレーション全体はアトミックです。insertMany オペレーションでエラーが発生した場合、insertMany オペレーション内の各挿入ステートメントはアトミックオペレーションとして実行されます。insertMany、、updateManyおよび deleteManyオペレーションに ACID プロパティが必要な場合は、トランザクションを使用することをお勧めします。

機能的な違いの更新

Amazon DocumentDB は、お客様より依頼のあった機能からさかのぼって、MongoDB との互換性を改善し続けています。このセクションでは、移行とアプリケーション構築を容易にするため、Amazon DocumentDB から削除された機能的な違いについて説明します。

トピック

- 配列インデックス作成

- マルチキーインデックス
- 文字列内の null 文字
- ロールベースアクセスコントロール
- $regex インデックス作成
- ネストされたドキュメントの投影

配列インデックス作成

2020 年 4 月 23 日以降、Amazon DocumentDB で 2,048 バイトを超える配列のインデックスを作成する機能がサポートされるようになりました。配列内の個々のキーの制限は依然として 2,048 バイトのままであり、これは MongoDB と一貫しています。

新しいインデックスを作成する場合、強化された機能を利用するためのアクションは必要ありません。既存のインデックスがある場合、インデックスを削除してから再作成することで、強化された機能を利用できます。強化された機能を使用した現在のインデックスバージョンは "v" : 3 です。

> **Note**
>
> 本番クラスターの場合、インデックスの削除はアプリケーションのパフォーマンスに影響を与える可能性があります。本番システムに変更を加える場合、最初にテストし、注意して進めることをお勧めします。さらに、インデックスの再作成にかかる時間は、コレクションの全体的なデータサイズの関数になります。

次のコマンドを使用して、インデックスのバージョンをクエリできます。

```
db.collection.getIndexes()
```

このオペレーションによる出力は、次のようになります。この出力では、インデックスのバージョンは "v" : 3 です。これは最新のインデックスバージョンです。

```
[
    {
        "v" : 3,
        "key" : {
        "_id" : 1
```

```
        },
        "name" : "_id_",
        "ns" : "test.test"
    }
]
```

マルチキーインデックス

2020 年 4 月 23 日より、Amazon DocumentDB では同じ配列内に複数のキーを持つ複合インデックスを作成する機能がサポートされるようになりました。

新しいインデックスを作成する場合、強化された機能を利用するためのアクションは必要ありません。既存のインデックスがある場合、インデックスを削除してから再作成することで、強化された機能を利用できます。強化された機能を使用した現在のインデックスバージョンは "v" : 3 です。

> **Note**
>
> 本番クラスターの場合、インデックスの削除はアプリケーションのパフォーマンスに影響を与える可能性があります。本番システムに変更を加える場合、最初にテストし、注意して進めることをお勧めします。さらに、インデックスの再作成にかかる時間は、コレクションの全体的なデータサイズの関数になります。

次のコマンドを使用して、インデックスのバージョンをクエリできます。

```
db.collection.getIndexes()
```

このオペレーションによる出力は、次のようになります。この出力では、インデックスのバージョンは "v" : 3 です。これは最新のインデックスバージョンです。

```
[
    {
        "v" : 3,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.test"
    }
```

```
]
```

文字列内の null 文字

2020 年 6 月 22 日現在、Amazon DocumentDB が文字列で null 文字 (`'\0'`) をサポートするようになりました。

ロールベースアクセスコントロール

2020 年 3 月 26 日現在、Amazon DocumentDB は組み込みロールによるロールベースのアクセスコントロール (RBAC) をサポートしています。詳細については[ロールベースのアクセスコントロール]を参照してください。

$regex インデックス作成

2020 年 6 月 22 日現在、Amazon DocumentDB は `$regex` 演算子がインデックスを活用する機能をサポートするようになりました。

`$regex` 演算子を使用してインデックスを活用するには、`hint()` コマンドを使用する必要があります。`hint()` を使用する際は、`$regex` を適用するフィールドの名前を指定する必要があります。たとえば、`product` というフィールドにインデックス名 `p_1` というインデックスがある場合、`db.foo.find({product: /^x.*/}).hint({product:1})` では `p_1` インデックスを使用しますが、`db.foo.find({product: /^x.*/}).hint("p_1")` ではインデックスを使用しません。インデックスが選択されているかどうかを確認するには、`explain()` コマンドを使用するか、低速クエリのログ記録にプロファイラーを使用します。例えば、`db.foo.find({product: /^x.*/}).hint("p_1").explain()`。

> **Note**
>
> `hint()` メソッドは、一度に 1 つのインデックスでのみ使用できます。

`$regex` クエリでのインデックスの使用は、プレフィックスを利用し、`i`、`m`、または `o` 正規表現オプションを指定しない正規表現クエリに最適化されます。

`$regex` でインデックスを使用する場合は、重複値の数がコレクション内のドキュメントの総数の 1% 未満であるごく一部のフィールドにインデックスを作成することをお勧めします。例えば、コレ

クションに 100,000 個のドキュメントが含まれている場合、同じ値が発生する回数が 1000 回以下のフィールドにのみインデックスを作成します。

ネストされたドキュメントの投影

Amazon DocumentDB と MongoDB 3.6 における `$project` 演算子と機能の差は、Amazon DocumentDB 4.0 で解決されていますが、Amazon DocumentDB 3.6 では未サポートのままです。

Amazon DocumentDB 3.6 は、投影を適用するときにネストされたドキュメントの最初のフィールドのみを考慮しますが、MongoDB 3.6 はサブドキュメントを解析し、各サブドキュメントにも投影を適用します。

例えば、投影が `“a.b.c”: 1` とすると、この動作は Amazon DocumentDB と MongoDB の両方で期待どおりに機能します。ただし、投影が `{a:{b:{c:1}}}` だとすると、Amazon DocumentDB 3.6 は投影を `a` にのみ適用し、`b` や `c` には適用しません。Amazon DocumentDB 4.0では、投影 `{a:{b:{c:1}}}` は `a`、`b`、および`c` に適用されます。

MongoDB との機能的な違い

トピック

- $vectorSearchoperator
- OpCountersCommand
- 管理データベースとコレクション
- cursormaxTimeMS
- explain()
- インデックスビルド
- パスに空のキーでルックアップ
- MongoDB API、オペレーション、およびデータ型
- mongodump および mongorestore ユーティリティ
- 結果の順序
- 再試行可能な書き込み
- スパースインデックス
- $all 式内での $elemMatch の使用
- $ne、$nin、$nor、$not、$exists および $elemMatch のインデックス作成

- フィールド名のドル ($) とドット (.)
- $lookup
- $natural および逆ソート

$vectorSearchoperator

Amazon DocumentDB は、独立した演算子として $vectorSearch をサポートしていません。代わりに、$search 演算子内の vectorSearch をサポートします。詳細については、「Amazon DocumentDB のベクトル検索」を参照してください。

OpCountersCommand

Amazon DocumentDB の OpCountersCommand 動作は MongoDB の opcounters.command 動作と次のように異なります。

- MongoDB は挿入、更新、opcounters.command 削除を除くすべてのコマンドをカウントしますが、Amazon DocumentDB はOpCountersCommand コマンドも除外します。find
- Amazon DocumentDB は、一部の内部コマンドを OpCountersCommand に加算します。

管理データベースとコレクション

Amazon DocumentDB は管理者データベースまたはローカルデータベースをサポートしていません。また、MongoDB system.* または startup_log コレクションもサポートしていません。

cursormaxTimeMS

Amazon DocumentDB では、cursor.maxTimeMS はカウンターを getMore リクエストごとにリセットします。したがって、3000 ミリ秒 maxTimeMS が指定されている場合、クエリは 2800 ミリ秒 になり、getMore リクエストは 300 ミリ秒 かかり、カーソルはタイムアウトしません。カーソルがタイムアウトするのは、単一のオペレーション (クエリまたは個々の getMore リクエスト) の場合のみです。指定された maxTimeMS よりも多くかかります。さらに、カーソルの実行時間をチェックするスイーパーは、5 分単位で実行されます。

explain()

Amazon DocumentDB は、分散型で耐障害性のある自己修復ストレージシステムを利用する専用データベースエンジンで MongoDB 3.6、4.0、5.0 APIs をエミュレートします。その結果、クエリプ

ランと explain() の出力は、Amazon DocumentDB と MongoDB の間で異なる場合があります。クエリプランを制御する場合は、$hint 演算子を使用して優先インデックスの選択を強制できます。

インデックスビルド

Amazon DocumentDB では、任意の時点でコレクションにおいてインデックスの作成が 1 つのみ許可されます。フォアグラウンドまたはバックグラウンドのいずれかで実行されます。インデックス構築が現在進行中のときに、createIndex() または dropIndex() などの操作が同じコレクションで発生した場合、新しく試行された操作は失敗します。

デフォルトでは、Amazon DocumentDB および MongoDB バージョン 4.0 のインデックスビルドはフォアグラウンドで行われます。MongoDB バージョン 4.2 以降では、createIndexes またはそのシェルヘルパー、createIndex() および createIndexes() に指定されている場合、バックグラウンドインデックス構築オプションは無視されます。

有効期限 (TTL) インデックスは、インデックスの構築が完了した後、ドキュメントの期限切れが開始されます。

パスに空のキーでルックアップ

パスの一部として (例 : x.、x..b) 空の文字列を含むキーを調べると、オブジェクトには空の文字列キーパス (例 : {"x" : [{ "" : 10 }, { "b" : 20 }]}) が配列内にあり、Amazon DocumentDB は MongoDB で同じルックアップを実行する場合とは異なる結果を返します。

MongoDB では、空の文字列キーがパスルックアップの末尾にない場合に、配列内の空のキーパスルックアップが期待どおりに機能します。ただし、空の文字列キーがパスルックアップの末尾にある場合、配列は検索されません。

ただし、Amazon DocumentDB では、配列内の最初の要素のみが読み取られます。getArrayIndexFromKeyString は空の文字列を0 に変換するため、文字列キールックアップは配列インデックスルックアップとして扱われるためです。

MongoDB API、オペレーション、およびデータ型

Amazon DocumentDB は MongoDB 3.6、4.0、5.0 API と互換性があります。サポートされている機能の最新リストについては、「Amazon DocumentDB でサポートされている MongoDB API、オペレーション、およびデータ型」を参照してください。

mongodump および mongorestore ユーティリティ

Amazon DocumentDB は管理者データベースをサポートしないため、mongodump または mongorestore ユーティリティを使用するときに、管理者データベースをダンプまたは復元しません。mongorestore を使用して、Amazon DocumentDB の新しいデータベースを作成するときに、復元オペレーションに加えてユーザーロールを再作成する必要があります。

> **Note**
>
> Amazon DocumentDB には、バージョン 100.6.1 までの MongoDB データベースツールをお勧めします。MongoDB データベースツールのダウンロードには、こちらからアクセスできます。

結果の順序

Amazon DocumentDB では、結果セットの暗黙的な並べ替え順序は保証されません。結果セットの順序を確実にするには、sort() を使用してソート順を明示的に指定します。

次の例では、在庫フィールドに基づいて、インベントリ収集の項目を降順にソートします。

```
db.inventory.find().sort({ stock: -1 })
```

$sort 集約ステージを使用する場合は、$sort ステージが、集約パイプラインの最後のステージでない限り、並べ替え順序は保持されません。$sort 集約ステージと組み合わせた $group 集約ステージを使用する場合、$sort 集約ステージはおよび $first および $last アキュムレータにのみ適用されます。Amazon DocumentDB 4.0 では、$push のサポートが追加され、前の $sort ステージ並べ替え順序を尊重します。

再試行可能な書き込み

MongoDB 4.2 互換ドライバー以降、再試行可能な書き込みはデフォルトで有効になっています。ただし、Amazon DocumentDB は現在、再試行可能な書き込みをサポートしていません。機能の違いは、次のようなエラーメッセージで示されます。

```
{"ok":0,"errmsg":"Unrecognized field: 'txnNumber'","code":9,"name":"MongoError"}
```

再試行可能な書き込みは、接続文字列 (などMongoClient("mongodb://my.mongodb.cluster/db?retryWrites=false")) または MongoClient コンストラクタのキーワード引数 (など) を使用して無効にできますMongoClient("mongodb://my.mongodb.cluster/db", retryWrites=False)。

次に、接続文字列の再試行可能な書き込みを無効にする Python の例を示します。

```
client =
 pymongo.MongoClient('mongodb://
<username>:<password>@docdb-2019-03-17-16-49-12.cluster-ccuszbx3pn5e.us-
east-1.docdb.amazonaws.com:27017/?
replicaSet=rs0',w='majority',j=True,retryWrites=False)
```

スパースインデックス

クエリで作成したスパースなインデックスを使用するには、インデックスをカバーするフィールドで $exists 句を使用する必要があります。を省略した場合$exists、Amazon DocumentDB はスパースインデックスを使用しません。

以下に例を示します。

```
db.inventory.count({ "stock": { $exists: true }})
```

スパースなマルチキーインデックスの場合、ドキュメントのルックアップによって値のセットが生成され、インデックス付きフィールドのサブセットのみが見つからない場合、Amazon DocumentDB は一意キー制約をサポートしません。たとえば、"a" : [{ "b" : 2 }, { "c" : 1 }] を入力すると、"a.c" がインデックスに格納されているため、createIndex({"a.b" : 1 }, { unique : true, sparse :true }) はサポートされません。

$all 式内での $elemMatch の使用

Amazon DocumentDB は $elemMatch 演算子が $all 内にある表現の使用をサポートしていません。回避策として、次のように $and 演算子を $elemMatch で使用できます。

元のオペレーション:

```
db.col.find({
  qty: {
```

```
    $all: [
      { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } },
      { "$elemMatch": { num: 40, size: "XL" } }
    ]
  }
})
```

更新されたオペレーション:

```
db.col.find({
  $and: [
    { qty: { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } } },
    { qty: { "$elemMatch": { qty: 40, size: "XL" } } }
  ]
})
```

$ne、$nin、$nor、$not、$exists および $elemMatch のインデックス作成

Amazon DocumentDB は、現在、$ne、$nin、$nor、$not、$exists、と $distinct の演算子でのインデックス使用をサポートしていません。そのため、これらの演算子を利用した結果は、収集スキャンになります。これらの演算子のいずれかを使用する前にフィルターまたは照合を実行すると、スキャン対象のデータ量が減り、パフォーマンスが向上します。

Amazon DocumentDB では、Amazon DocumentDB 5.0 とエラスティッククラスターの $elemMatch オペレーターによるインデックススキャンのサポートが追加されました。インデックススキャンは、クエリのみのフィルタに 1 つのレベルの $elemMatch フィルタがある場合はサポートされますが、ネストされた $elemMatch クエリが含まれる場合はサポートされません。

Amazon DocumentDB 5.0 でのインデックススキャンをサポートする $elemMatch クエリシェイプ:

```
db.foo.find( { "a": {$elemMatch: { "b": "xyz", "c": "abc"} } })
```

Amazon DocumentDB 5.0 でインデックススキャンをサポートしない $elemMatch クエリシェイプ:

```
db.foo.find( { "a": {$elemMatch: { "b": {$elemMatch: { "d": "xyz", "e": "abc"} }} } })
```

フィールド名のドル ($) とドット (.)

Amazon DocumentDB は、ネストされたオブジェクトでの $in、$nin、および $all の Dollar($) プレフィックス付きフィールドのクエリをサポートしていません。たとえば、次のクエリは Amazon DocumentDB では有効ではありません。

```
coll.find({"field": {"$all": [{ "$a": 1 }]}})
```

$lookup

Amazon DocumentDB は、等価マッチを実行する機能 (左外部結合など) および相関のないサブクエリをサポートしますが、相関のあるサブクエリはサポートしていません。

$lookup でインデックスを利用する

$lookup ステージ演算子で、インデックスを利用できるようになりました 。ユースケースに基づいて、パフォーマンスを最適化するために使用できるインデックス作成アルゴリズムが複数あります。このセクションでは、$lookup のさまざまなインデックス作成アルゴリズムについて説明し、ワークロードに最適なオプションを選択するのに役立ちます。

デフォルトでは、Amazon DocumentDB は allowDiskUse:false が使用された場合はハッシュアルゴリズムを使用し、allowDiskUse:true が使用された場合はソートマージを使用します。

> **Note**
>
> allowDiskUse オプションは現在、find コマンド向けにはサポートされていません。このオプションは集約の一部としてのみサポートされます。allowDiskUse:true で集約フレームワークを使用して、メモリ制限を超える可能性のある大規模なクエリを処理することをお勧めします。

ユースケースによっては、クエリオプティマイザで別のアルゴリズムを使用するように強制することが望ましい場合があります。以下は、$lookup の集約演算子を利用できる、さまざまなインデックス作成アルゴリズムの例です。

- ネストループ: 外部コレクションが 1 GB 未満で、外部コレクションのフィールドにインデックスがある場合は、通常、ネストループプランがワークロードに役立ちます。ネスト· ループ· アルゴリズムを使用している場合、説明プランではステージが NESTED_LOOP_LOOKUP と表示されます。

- ソートマージ: ソートマージプランは、外部コレクションのルックアップに使用されるフィールドにインデックスがなく、作業データセットがメモリに収まらない場合にワークロードにとって有益です。ソートマージアルゴリズムが使用されている場合、説明プランではステージを SORT_LOOKUP と表示されます。
- ハッシュ: 外部コレクションが < 1 GB で、作業データセットがメモリに収まる場合、ハッシュプランは通常ワークロードにとって有益です。ハッシュアルゴリズムが使用されている場合、説明プランはステージを HASH_LOOKUP と表示されます。

explain クエリで を使用して、$lookup演算子に使用されているインデックス作成アルゴリズムを特定できます。以下に例を示します。

```
db.localCollection.explain().aggregate(
   [
      {
         $lookup:
            {
               from: "foreignCollection",
               localField: "a",
               foreignField: "b",
               as: "joined"
            }
      }
   ]
)

output
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "test.localCollection",
        "winningPlan" : {
            "stage" : "SUBSCAN",
            "inputStage" : {
                "stage" : "SORT_AGGREGATE",
                "inputStage" : {
                    "stage" : "SORT",
                    "inputStage" : {
                        "stage" : "NESTED_LOOP_LOOKUP",
                        "inputStages" : [
                            {
```

```
                                "stage" : "COLLSCAN"
                            },
                            {
                                "stage" : "FETCH",
                                "inputStage" : {
                                    "stage" : "COLLSCAN"
                                }
                            }
                        ]
                    }
                }
            }
        },
    "serverInfo" : {
        "host" : "devbox-test",
        "port" : 27317,
        "version" : "3.6.0"
    },
    "ok" : 1
}
```

explain() メソッドを使用する代わりに、プロファイラーを使用して、 $lookup演算子の使用で使用されているアルゴリズムを確認できます。プロファイラーの詳細については、「Amazon DocumentDB オペレーションのプロファイリング」を参照してください。

planHint を使用する

クエリオプティマイザで別のインデックス作成アルゴリズムと $lookup を強制的に使用する場合、planHint を使用できます。これを行うには、集約ステージオプションのコメントを使用して、別のプランを強制します。コメントの構文の例を以下に示します。

```
comment : {
    comment :  "<string>",
    lookupStage : { planHint : "SORT" | "HASH" | "NESTED_LOOP" }
}
```

以下は、planHint を使用してクエリオプティマイザに HASH インデックスアルゴリズムの使用を強制する例です。

```
db.foo.aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
            },
      }
   ]
),
{
   comment : "{ \"lookupStage\" : { \"planHint\": \"HASH\" }}"
```

どのアルゴリズムがワークロードに最適かをテストするには、executionStats パラメータにある explain メソッドを使用して、$lookup ステージの実行時間を測定することができ、これはインデックス作成アルゴリズム (つまり、HASH / SORT/NESTED_LOOP) を変更しながらできます。

次の例は executionStats を使用して、$lookup ステージの実行時間を測定するには、SORT アルゴリズムを使用します。

```
db.foo.explain("executionStats").aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
            },
      }
   ]
),
{
   comment : "{ \"lookupStage\" : { \"planHint\": \"SORT\" }}"
```

$natural および逆ソート

Amazon DocumentDB は$natural、フォワードコレクションスキャンでのみ をサポートします。逆引き収集スキャン ({$natural: -1}) は になりますMongoServerError。

Amazon DocumentDB でサポートされている MongoDB API、オペレーション、およびデータ型

Amazon DocumentDB (MongoDB 互換) は、MongoDB ワークロードをサポートする、高速、スケーラブル、高可用性、完全管理のドキュメントデータベース・サービスです。Amazon DocumentDB は MongoDB 3.6、4.0、5.0 API と互換性があります。このセクションでは、サポートされている機能を一覧表示します。MongoDB API とドライバの使用のサポートについては、MongoDB コミュニティフォーラムを参照してください。Amazon DocumentDB サービスの使用に関するサポートについては、適切な AWS サポートチームにお問い合わせください。Amazon DocumentDB と MongoDB の機能的な違いについては、機能の違い: Amazon DocumentDB と MongoDB を参照してください。

内部専用であるか、フルマネージドサービスには適用されない、MongoDB コマンドと演算子は、サポートされておらず、サポートされている機能のリストにも含まれていません。

初めてのリリース以来 50 を超える新しい機能を追加しており、今後もお客様が必要とする機能を遡って提供していきます。最新のローンチに関する情報は、「Amazon DocumentDB に関する発表」を参照してください。

サポート対象外の機能をリクエストしたい場合は、アカウント ID、リクエストする機能、およびユースケースを記載した E メールを Amazon DocumentDB サービスチーム まで送信してお知らせください。

トピック

- データベースコマンド
- クエリおよびプロジェクション演算子
- 更新演算子
- 地理空間
- カーソルメソッド
- 集約パイプライン演算子
- データ型
- インデックスとインデックスプロパティ

データベースコマンド

トピック

- 管理コマンド
- 集計
- 認証
- 診断コマンド
- クエリ操作と書き込み操作
- ロール管理コマンド
- セッションコマンド
- ユーザー管理
- シャーディングコマンド

管理コマンド

コマンド	3.6	4.0	5.0	Elastic クラスター
上限付きコレクション	いいえ	いいえ	いいえ	いいえ
cloneCollectionAsCapped	いいえ	いいえ	いいえ	いいえ
collMod	部分的	部分的	部分的	部分的
collMod: expireAfterSeconds	はい	あり	あり	はい
convertToCapped	いいえ	いいえ	いいえ	いいえ
copydb	いいえ	いいえ	いいえ	いいえ
作成	はい	あり	あり	はい
createView	いいえ	いいえ	いいえ	いいえ

コマンド	3.6	4.0	5.0	Elastic クラスター
createIndexes	はい	あり	あり	はい
currentOp	はい	あり	あり	はい
drop	はい	あり	あり	はい
dropDatabase	はい	あり	あり	はい
dropIndexes	はい	あり	あり	はい
filemd5	いいえ	いいえ	いいえ	いいえ
getAuditConfig	いいえ	はい	はい	いいえ
killCursors	はい	あり	あり	はい
killOp	はい	あり	あり	はい
listCollections*	はい	あり	あり	はい
listDatabases	はい	あり	あり	はい
listIndexes	はい	あり	あり	はい
reIndex	いいえ	いいえ	あり	いいえ
renameCollection	はい	あり	はい	いいえ
setAuditConfig	いいえ	はい	はい	いいえ

* コンソールでは、`type` キーはサポートされていません。

集計

コマンド	3.6	4.0	5.0	Elastic クラスター
aggregate	はい	あり	あり	はい
count	はい	あり	あり	はい
distinct	はい	あり	あり	はい
mapReduce	いいえ	いいえ	いいえ	いいえ

認証

コマンド	3.6	4.0	5.0	Elastic クラスター
authenticate	はい	あり	あり	はい
ログアウト	はい	あり	あり	はい

診断コマンド

コマンド	3.6	4.0	5.0	Elastic クラスター
buildInfo	はい	あり	あり	はい
collStats	はい	あり	あり	はい
connPoolStats	いいえ	いいえ	いいえ	いいえ
connectionStatus	はい	あり	あり	はい
dataSize	はい	あり	あり	はい

コマンド	3.6	4.0	5.0	Elastic クラスター
dbHash	いいえ	いいえ	いいえ	いいえ
dbStats	はい	あり	あり	はい
explain	はい	あり	あり	はい
説明: execution Stats	はい	あり	あり	はい
機能	いいえ	いいえ	いいえ	いいえ
hostInfo	はい	あり	あり	はい
listCommands	はい	あり	あり	はい
プロファイラー	あり	あり	あり	いいえ
serverStatus	はい	あり	あり	はい
top	はい	あり	あり	はい

クエリ操作と書き込み操作

コマンド	3.6	4.0	5.0	Elastic クラスター
Change streams	あり	あり	あり	いいえ
削除	はい	あり	あり	はい
find	はい	あり	あり	はい
findAndModify	はい	あり	あり	はい
getLastError	いいえ	いいえ	いいえ	いいえ
getMore	はい	あり	あり	はい

コマンド	3.6	4.0	5.0	Elastic クラスター
getPrevError	いいえ	いいえ	いいえ	いいえ
GridFS	はい	あり	はい	いいえ
insert	はい	あり	あり	はい
parallelCollectionScan	いいえ	いいえ	いいえ	いいえ
resetError	いいえ	いいえ	いいえ	いいえ
更新	はい	あり	あり	はい
ReplaceOne	はい	あり	あり	はい

ロール管理コマンド

コマンド	3.6	4.0	5.0	Elastic クラスター
createRole	はい	あり	はい	いいえ
dropAllRolesFromDatabase	はい	あり	はい	いいえ
dropRole	はい	あり	はい	いいえ

コマンド	3.6	4.0	5.0	Elastic クラスター
gは	あり	はい	いいえ	
reは	あり	はい	いいえ	
rvc	はい	あり	はい	いいえ
rは	あり	はい	いいえ	
ueは	あり	はい	いいえ	

セッションコマンド

コマンド	3.6	4.0	5.0	Elastic クラスター
abortTransaction	いいえ	はい	はい	いいえ
commitTransaction	いいえ	はい	はい	いいえ
endSessions	いいえ	いいえ	いいえ	いいえ

コマンド	3.6	4.0	5.0	Elastic クラスター
killAllSessions	いいえ	はい	はい	いいえ
killAllSessionsByPattern	いいえ	いいえ	いいえ	いいえ
killSessions	いいえ	はい	はい	いいえ
refreshSessions	いいえ	いいえ	いいえ	いいえ
startSession	いいえ	はい	はい	いいえ

ユーザー管理

コマンド	3.6	4.0	5.0	Elastic クラスター
createUser	はい	あり	あり	はい
dropAllUsersFromDatabase	はい	あり	あり	はい
dropUser	はい	あり	あり	はい
grantRolesToUser	はい	あり	あり	はい
revokeRolesFromUser	はい	あり	あり	はい
updateUser	はい	あり	あり	はい
usersInfo	はい	あり	あり	はい

シャーディングコマンド

コマンド	Elastic クラスター
abortReshardCollection	いいえ
addShard	いいえ
addShardToZone	いいえ
balancerCollectionStatus	いいえ
balancerStart	いいえ
balancerStatus	いいえ
balancerStop	いいえ
checkShardingIndex	いいえ
clearJumboFlag	いいえ
cleanupOrphaned	いいえ
cleanupReshardCollection	いいえ
commitReshardCollection	いいえ
enableSharding	はい
flushRouterConfig	いいえ
getShardMap	いいえ
getShardVersion	いいえ
isdbgrid	いいえ
ListShards	いいえ
medianKey	いいえ

コマンド	Elastic クラスター
moveChunk	いいえ
movePrimary	いいえ
mergeChunks	いいえ
refineCollectionShardKey	いいえ
removeShard	いいえ
removeShardFromZone	いいえ
reshardCollection	いいえ
setAllowMigrations	いいえ
setShardVersion	いいえ
shardCollection	はい
shardingState	いいえ
split	いいえ
splitVector	いいえ
unsetSharding	いいえ
updateZoneKeyRange	いいえ

クエリおよびプロジェクション演算子

トピック

- 配列演算子
- ビット演算子
- コメント演算子
- 比較演算子

- 要素演算子
- 評価クエリ演算子
- 論理演算子
- プロジェクション演算子

配列演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$all	はい	あり	あり	はい
$elemMatch	はい	あり	あり	はい
$size	はい	あり	あり	はい

ビット演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$bitsAllSet	はい	あり	あり	はい
$bitsAnySet	はい	あり	あり	はい
$bitsAllClear	はい	あり	あり	はい
$bitsAnyClear	はい	あり	あり	はい

コメント演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$comment	はい	あり	あり	はい

比較演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$eq	はい	あり	あり	はい
$gt	はい	あり	あり	はい
$gte	はい	あり	あり	はい
$in	はい	あり	あり	はい
$lt	はい	あり	あり	はい
$lte	はい	あり	あり	はい
$ne	はい	あり	あり	はい
$nin	はい	あり	あり	はい

要素演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$exists	はい	あり	あり	はい
$type	はい	あり	あり	はい

評価クエリ演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$expr	いいえ	はい	はい	いいえ

コマンド	3.6	4.0	5.0	Elastic クラスター
$jsonSchema	いいえ	はい	はい	いいえ
$mod	はい	あり	あり	はい
$regex	はい	あり	あり	はい
$text	いいえ	いいえ	あり	いいえ
$where	いいえ	いいえ	いいえ	いいえ

論理演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$and	はい	あり	あり	はい
$nor	はい	あり	あり	はい
$not	はい	あり	あり	はい
$or	はい	あり	あり	はい

プロジェクション演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$	はい	あり	あり	はい
$elemMatch	はい	あり	あり	はい
$meta	いいえ	いいえ	あり	いいえ

コマンド	3.6	4.0	5.0	Elastic クラスター
$slice	はい	あり	あり	はい

更新演算子

トピック

- 配列演算子
- ビット演算子
- フィールド演算子
- 更新修飾子

配列演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$	はい	あり	あり	あり
$[]	あり	あり	あり	はい
$[<identifier>]	はい	あり	あり	はい
$addToSet	はい	あり	あり	はい
$pop	はい	あり	あり	はい
$pullAll	はい	あり	あり	はい
$pull	はい	あり	あり	はい
$push	はい	あり	あり	はい

ビット演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$bit	はい	あり	あり	はい

フィールド演算子

演算子	3.6	4.0	5.0	Elastic クラスター
$currentDate	はい	あり	あり	はい
$inc	はい	あり	あり	はい
$max	はい	あり	あり	はい
$min	はい	あり	あり	はい
$mul	はい	あり	あり	はい
$rename	はい	あり	あり	はい
$set	はい	あり	あり	はい
$setOnInsert	はい	あり	あり	はい
$unset	はい	あり	あり	はい

更新修飾子

演算子	3.6	4.0	5.0	Elastic クラスター
$each	はい	あり	あり	はい

演算子	3.6	4.0	5.0	Elastic クラスター
$position	はい	あり	あり	はい
$slice	はい	あり	あり	はい
$sort	はい	あり	あり	はい

地理空間

ジオメトリ指定子

クエリセレクター	3.6	4.0	5.0	Elastic クラスター
$box	いいえ	いいえ	いいえ	いいえ
$center	いいえ	いいえ	いいえ	いいえ
$centerSphere	いいえ	いいえ	いいえ	いいえ
$geometry	はい	あり	あり	はい
$maxDistance	はい	あり	あり	はい
$minDistance	はい	あり	あり	はい
$nearSphere	はい	あり	あり	はい
$polygon	いいえ	いいえ	いいえ	いいえ
$uniqueDocs	いいえ	いいえ	いいえ	いいえ

クエリセレクター

コマンド	3.6	4.0	5.0	Elastic クラスター
$geoIntersects	はい	あり	あり	はい
$geoWithin	はい	あり	あり	はい
$near	いいえ	いいえ	いいえ	いいえ
$nearSphere	はい	あり	あり	はい
$polygon	いいえ	いいえ	いいえ	いいえ
$uniqueDocs	いいえ	いいえ	いいえ	いいえ

カーソルメソッド

コマンド	3.6	4.0	5.0	Elastic クラスター
cursor.batchSize()	はい	あり	あり	はい
cursor.close()	はい	あり	あり	はい
cursor.collation()	いいえ	いいえ	いいえ	いいえ
cursor.comment()	はい	あり	あり	はい
cursor.count()	はい	あり	あり	はい
cursor.explain()	はい	あり	はい	いいえ
cursor.forEach()	はい	あり	あり	はい
cursor.hasNext()	はい	あり	あり	はい

コマンド	3.6	4.0	5.0	Elastic クラスター
cursor.hint()	はい	あり	はい	はい*
cursor.isClosed()	はい	あり	あり	はい
cursor.isExhausted()	はい	あり	はい	いいえ
cursor.itcount()	はい	あり	はい	いいえ
cursor.limit()	はい	あり	はい	いいえ
cursor.map()	はい	あり	はい	いいえ
cursor.max()	いいえ	いいえ	いいえ	いいえ
cursor.maxScan()	はい	あり	はい	いいえ
cursor.maxTimeMS()	はい	あり	はい	いいえ
cursor.min()	いいえ	いいえ	いいえ	いいえ
cursor.next()	はい	あり	あり	はい
cursor.noCursorTimeout()	いいえ	いいえ	いいえ	いいえ
cursor.objsLeftInBatch()	はい	あり	はい	いいえ
cursor.pretty()	はい	あり	はい	いいえ
cursor.readConcern()	はい	あり	はい	いいえ
cursor.readPref()	はい	あり	はい	いいえ

コマンド	3.6	4.0	5.0	Elastic クラスター
cursor.returnKey()	いいえ	いいえ	いいえ	いいえ
cursor.showRecordId()	いいえ	いいえ	いいえ	いいえ
cursor.size()	はい	あり	はい	いいえ
cursor.skip()	はい	あり	はい	いいえ
cursor.sort()	はい	あり	はい	いいえ
cursor.tailable()	いいえ	いいえ	いいえ	いいえ
cursor.toArray()	はい	あり	はい	いいえ

* インデックス `hint` がインデックス式でサポートされるようになりました。例えば、`db.foo.find().hint({x:1})`。

集約パイプライン演算子

トピック

- アキュムレータ式
- 算術演算子
- 配列演算子
- ブール演算子
- 比較演算子
- 条件式演算子
- データ型演算子
- データサイズ演算子
- 日付演算子
- リテラル演算子

- マージ演算子
- 自然演算子
- セット演算子
- ステージ演算子
- 文字列演算子
- システム変数
- テキスト検索演算子
- 型変換演算子
- 変数演算子
- その他の演算子

アキュムレータ式

式	3.6	4.0	5.0	Elastic クラスター
$accumulator	-	-	いいえ	いいえ
$addToSet	はい	あり	あり	はい
$avg	はい	あり	あり	はい
$count	-	-	いいえ	いいえ
$covariancePop	いいえ	いいえ	いいえ	いいえ
$covarianceSamp	いいえ	いいえ	いいえ	いいえ
$denseRank	いいえ	いいえ	いいえ	いいえ
$ 派生	いいえ	いいえ	いいえ	いいえ
$documentNumber	いいえ	いいえ	いいえ	いいえ
$expMovingAvg	いいえ	いいえ	いいえ	いいえ

式	3.6	4.0	5.0	Elastic クラスター
$first	はい	あり	あり	はい
$integral	いいえ	いいえ	いいえ	いいえ
$last	はい	あり	あり	はい
$max	はい	あり	あり	はい
$min	はい	あり	あり	はい
$push	はい	あり	あり	はい
$rank	いいえ	いいえ	いいえ	いいえ
$shift	いいえ	いいえ	いいえ	いいえ
$stdDevPop	いいえ	いいえ	いいえ	いいえ
$stdDevSamp	いいえ	いいえ	いいえ	いいえ
$sum	はい	あり	あり	はい

算術演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$abs	はい	あり	あり	はい
$add	はい	あり	あり	はい
$ceil	いいえ	はい	あり	はい
$divide	はい	あり	あり	はい
$exp	いいえ	はい	あり	はい

コマンド	3.6	4.0	5.0	Elastic クラスター
$floor	いいえ	はい	あり	はい
$ln	いいえ	はい	あり	はい
$log	いいえ	はい	あり	はい
$log10	いいえ	はい	あり	はい
$mod	はい	あり	あり	はい
$multiply	はい	あり	あり	はい
$pow	いいえ	いいえ	いいえ	いいえ
$round	-	-	いいえ	いいえ
$sqrt	いいえ	はい	あり	はい
$subtract	はい	あり	あり	はい
$trunc	いいえ	いいえ	いいえ	いいえ

配列演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$arrayElemAt	はい	あり	あり	はい
$arrayToObject	はい	あり	あり	はい
$concatArrays	はい	あり	あり	はい
$filter	はい	あり	あり	はい
$first	-	-	はい	いいえ

コマンド	3.6	4.0	5.0	Elastic クラスター
$in	はい	あり	あり	はい
$indexOfArray	はい	あり	あり	はい
$isArray	はい	あり	あり	はい
$last	-	-	はい	いいえ
$objectToArray	はい	あり	あり	はい
$range	はい	あり	あり	はい
$reverseArray	はい	あり	あり	はい
$reduce	はい	あり	あり	はい
$size	はい	あり	あり	はい
$slice	はい	あり	あり	はい
$zip	はい	あり	あり	はい

ブール演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$and	はい	あり	あり	はい
$not	はい	あり	あり	はい
$or	はい	あり	あり	はい

比較演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$cmp	はい	あり	あり	はい
$eq	はい	あり	あり	はい
$gt	はい	あり	あり	はい
$gte	はい	あり	あり	はい
$lt	はい	あり	あり	はい
$lte	はい	あり	あり	はい
$ne	はい	あり	あり	はい

条件式演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$cond	はい	あり	あり	はい
$ifNull	はい	あり	あり	はい
$switch	いいえ	はい	はい	いいえ

データ型演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$type	はい	あり	あり	はい

データサイズ演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$binarySize	-	-	いいえ	いいえ
$bsonSize	-	-	いいえ	いいえ

日付演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$dateAdd	いいえ	いいえ	はい	はい
$dateDiff	-	-	いいえ	いいえ
$dateFromParts	いいえ	いいえ	いいえ	いいえ
$dateFromString	はい	あり	あり	はい
$dateSubtract	いいえ	いいえ	はい	はい
$dateToParts	いいえ	いいえ	いいえ	いいえ
$dateToString	はい	あり	あり	はい
$dateTrunc	-	-	いいえ	いいえ
$dayOfMonth	はい	あり	あり	はい
$dayOfWeek	はい	あり	あり	はい
$dayOfYear	はい	あり	あり	はい
$hour	はい	あり	あり	はい
$isoDayOfWeek	はい	あり	あり	はい

コマンド	3.6	4.0	5.0	Elastic クラスター
$isoWeek	はい	あり	あり	はい
$isoWeekYear	はい	あり	あり	はい
$millisecond	はい	あり	あり	はい
$minute	はい	あり	あり	はい
$month	はい	あり	あり	はい
$second	はい	あり	あり	はい
$week	はい	あり	あり	はい
$year	はい	あり	あり	はい

リテラル演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$literal	はい	あり	あり	はい

マージ演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$mergeObjects	はい	あり	あり	はい

自然演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$natural	はい	あり	あり	はい

セット演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$allElementsTrue	いいえ	はい	あり	はい
$anyElementTrue	いいえ	はい	あり	はい
$setDifference	いいえ	はい	あり	はい
$setEquals	はい	あり	あり	はい
$setIntersection	はい	あり	あり	はい
$setIsSubset	はい	あり	あり	はい
$setUnion	はい	あり	あり	はい
$setWindowFields	いいえ	いいえ	いいえ	いいえ

ステージ演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$addFields	はい	あり	あり	はい
$bucket	いいえ	いいえ	いいえ	いいえ
$bucketAuto	いいえ	いいえ	いいえ	いいえ
$collStats	いいえ	はい	はい	いいえ
$count	はい	あり	あり	はい
$currentOp	はい	あり	あり	はい
$facet	いいえ	いいえ	いいえ	いいえ
$geoNear	はい	あり	あり	はい
$graphLookup	いいえ	いいえ	いいえ	いいえ
$group	はい	あり	あり	はい
$indexStats	はい	あり	あり	はい
$limit	はい	あり	あり	はい
$listLocalSessions	いいえ	いいえ	いいえ	いいえ
$listSessions	いいえ	いいえ	いいえ	いいえ
$lookup	はい	あり	あり	はい
$match	はい	あり	あり	はい
$merge	-	-	いいえ	いいえ
$out	はい	あり	はい	いいえ

コマンド	3.6	4.0	5.0	Elastic クラスター
$planCacheStats	-	-	いいえ	いいえ
$project	はい	あり	あり	はい
$redact	はい	あり	あり	はい
$replaceRoot	はい	あり	あり	はい
$sample	はい	あり	あり	はい
$set	-	-	いいえ	いいえ
$setWindowFields	-	-	いいえ	いいえ
$skip	はい	あり	あり	はい
$sort	はい	あり	あり	はい
$sortByCount	いいえ	いいえ	いいえ	いいえ
$unionWith	-	-	いいえ	いいえ
$unset	-	-	いいえ	いいえ
$unwind	はい	あり	あり	はい

文字列演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$concat	はい	あり	あり	はい
$indexOfBytes	はい	あり	あり	はい
$indexOfCP	はい	あり	あり	はい

コマンド	3.6	4.0	5.0	Elastic クラスター
$ltrim	いいえ	はい	はい	いいえ
$regexFind	-	-	はい	いいえ
$regexFindAll	-	-	はい	いいえ
$regexMatch	-	-	はい	いいえ
$replaceAll	-	-	はい	いいえ
$replaceOne	-	-	はい	いいえ
$rtrim	いいえ	はい	はい	いいえ
$split	はい	あり	あり	はい
$strcasecmp	はい	あり	あり	はい
$strLenBytes	はい	あり	あり	はい
$strLenCP	はい	あり	あり	はい
$substr	はい	あり	あり	はい
$substrBytes	はい	あり	あり	はい
$substrCP	はい	あり	あり	はい
$toLower	はい	あり	あり	はい
$toUpper	はい	あり	あり	はい
$trim	いいえ	はい	はい	いいえ

システム変数

コマンド	3.6	4.0	5.0	Elastic クラスター
$$CURRENT	いいえ	いいえ	いいえ	いいえ
$$DESCEND	はい	あり	あり	はい
$$KEEP	はい	あり	あり	はい
$$PRUNE	はい	あり	あり	はい
$$REMOVE	いいえ	いいえ	いいえ	いいえ
$$ROOT	はい	あり	あり	はい

テキスト検索演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$meta	いいえ	いいえ	あり	いいえ
$search	いいえ	いいえ	あり	いいえ

型変換演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$convert	いいえ	はい	あり	はい
$isNumber	-	-	いいえ	いいえ
$toBool	いいえ	はい	あり	はい

コマンド	3.6	4.0	5.0	Elastic クラスター
$toDate	いいえ	はい	あり	はい
$toDecimal	いいえ	はい	あり	はい
$toDouble	いいえ	はい	あり	はい
$toInt	いいえ	はい	あり	はい
$toLong	いいえ	はい	あり	はい
$toObjectId	いいえ	はい	あり	はい
$toString	いいえ	はい	あり	はい

変数演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$let	はい	あり	あり	はい
$map	はい	あり	あり	はい

その他の演算子

コマンド	3.6	4.0	5.0	Elastic クラスター
$getField	-	-	いいえ	いいえ
$rand	-	-	いいえ	いいえ
$sampleRate	-	-	いいえ	いいえ

データ型

コマンド	3.6	4.0	5.0	Elastic クラスター
32 ビット整数 (int)	はい	あり	あり	はい
64 ビット整数 (long)	はい	あり	あり	はい
配列	はい	あり	あり	はい
バイナリデータ	はい	あり	あり	はい
ブール値	はい	あり	あり	はい
日付	可能	あり	あり	はい
DBPointer	いいえ	いいえ	いいえ	いいえ
DBRefs	いいえ	いいえ	いいえ	いいえ
Decimal128	はい	あり	あり	はい
倍精度	はい	あり	あり	はい
JavaScript	いいえ	いいえ	いいえ	いいえ
JavaScript (スコープ付き)	いいえ	いいえ	いいえ	いいえ
MaxKey	はい	あり	あり	はい
MinKey	はい	あり	あり	はい
Null	はい	あり	あり	はい
オブジェクト	はい	あり	あり	はい
ObjectId	はい	あり	あり	はい

コマンド	3.6	4.0	5.0	Elastic クラスター
正規表現	はい	あり	あり	可能
String	可能	あり	あり	はい
記号	いいえ	いいえ	いいえ	いいえ
Timestamp	はい	あり	あり	はい
未定義	いいえ	いいえ	いいえ	いいえ

インデックスとインデックスプロパティ

トピック

- インデックス
- インデックスプロパティ

インデックス

コマンド	3.6	4.0	5.0	Elastic クラスター
2dsphere	はい	あり	あり	はい
2d インデックス	いいえ	いいえ	いいえ	いいえ
複合インデックス	はい	あり	あり	はい
ハッシュインデックス	いいえ	いいえ	いいえ	いいえ
マルチキーインデックス	はい	あり	あり	はい

コマンド	3.6	4.0	5.0	Elastic クラスター
単一フィールドインデックス	はい	あり	あり	はい
テキストインデックス	いいえ	いいえ	あり	いいえ
ワイルドカード	いいえ	いいえ	いいえ	いいえ

インデックスプロパティ

コマンド	3.6	4.0	5.0	Elastic クラスター
背景	はい	あり	あり	はい
大文字と小文字を区別しない	いいえ	いいえ	いいえ	いいえ
[非表示]	いいえ	いいえ	いいえ	いいえ
部分的	いいえ	いいえ	あり	いいえ
スパース	はい	あり	あり	はい
テキスト	いいえ	いいえ	あり	いいえ
TTL	はい	あり	あり	はい
Unique	はい	あり	あり	はい
ベクトル	いいえ	いいえ	あり	いいえ

Amazon DocumentDB 生成人工知能

Amazon DocumentDB は、機械学習 (ML) モデルと生成人工知能 (AI) モデルが Amazon DocumentDB に保存されているデータをリアルタイムで操作できるようにする機能を提供します。お客様は、別のインフラストラクチャの管理、別のサービスに接続するためのコードの記述、プライマリデータベースからのデータの複製に時間を費やす必要がなくなります。

人工知能と AWS が AI のニーズをサポートする方法の詳細については、この「What-is」記事を参照してください。

トピック

- Amazon SageMaker AI Canvas によるノーコード機械学習
- Amazon DocumentDB のベクトル検索

Amazon SageMaker AI Canvas によるノーコード機械学習

Amazon SageMaker AI Canvas を使用すると、1 行のコードを記述することなく、独自の AI/ML モデルを構築できます。リグレッションや予測などの一般的なユースケース用の ML モデルを構築し、Amazon Bedrock から基盤モデル (FM) にアクセスして評価できます。Amazon SageMaker AI JumpStart からパブリック FMs にアクセスして、コンテンツ生成、テキスト抽出、テキスト要約を行い、生成 AI ソリューションをサポートすることもできます。

SageMaker AI Canvas を使用してノーコード ML モデルを構築する方法

Amazon DocumentDB が Amazon SageMaker AI Canvas と統合され、Amazon DocumentDB に保存されているデータでノーコード機械学習 (ML) が可能になりました。Amazon DocumentDB に保存されているデータを使用して、いっさいコードを記述することなく、リグレッションや予測などのニーズに合った ML モデルを構築し、コンテンツの要約や生成の基礎モデルを使用できるようになりました。

SageMaker AI Canvas は、Amazon DocumentDB のお客様が AI/ML の専門知識を必要とせずに予測を生成したり、1 行のコードを記述したりできるようにするビジュアルインターフェイスを提供します。お客様は、 から SageMaker AI Canvas ワークスペースを起動し AWS Management Console、データ準備とモデルトレーニングのために Amazon DocumentDB データをインポートして参加できるようになりました。Amazon DocumentDB のデータを SageMaker AI Canvas で使用して、顧客離れの予測、不正検出、メンテナンス障害の予測、ビジネスメトリクスの予測、コンテンツの生成を行うモデルを構築および拡張できるようになりました。お客様は、SageMaker AI Canvas の

QuickSight とのネイティブ統合を使用して、チーム間で ML 主導のインサイトを公開および共有できるようになりました。SageMaker AI Canvas のデータ取り込みパイプラインは、デフォルトで Amazon DocumentDB セカンダリインスタンスで実行されるため、アプリケーションと SageMaker AI Canvas の取り込みワークロードのパフォーマンスが妨げられることはありません。

Amazon DocumentDB のお客様は、新しい Amazon DocumentDB ノーコード ML コンソールページに移動し、新規または利用可能な SageMaker AI Canvas ワークスペースに接続することで、SageMaker AI Canvas の使用を開始できます。

SageMaker AI ドメインとユーザープロファイルの設定

VPC 専用モードで実行されている SageMaker AI ドメインから Amazon DocumentDB クラスターに接続できます。VPC で SageMaker AI ドメインを起動することで、SageMaker AI Studio および Canvas 環境からのデータフローを制御できます。これにより、インターネットアクセスを制限し、標準の AWS ネットワークとセキュリティ機能を使用してトラフィックをモニタリングおよび検査し、VPC エンドポイントを介して他の AWS リソースに接続できます。Amazon SageMaker AI Canvas の開始方法とAmazon SageMaker AI デベロッパーガイドにあるインターネットアクセスのない VPC で Amazon SageMaker AI Canvas を設定して、Amazon DocumentDB クラスターに接続するための SageMaker AI ドメインを作成してください。 Amazon SageMaker

Amazon DocumentDB と SageMaker AI Canvas の IAM アクセス許可の設定

自分に関連付けられたロールと ID に `AmazonDocDBConsoleFullAccess` がアタッチされている Amazon DocumentDB ユーザーは、 AWS Management Consoleにアクセスできます。前述のロールまたは ID に以下のアクションを追加して、Amazon SageMaker AI Canvas でノーコード機械学習へのアクセスを提供します。

```
"sagemaker:CreatePresignedDomainUrl",
"sagemaker:DescribeDomain",
"sagemaker:ListDomains",
"sagemaker:ListUserProfiles"
```

SageMaker AI Canvas のデータベースユーザーとロールの作成

Amazon DocumentDB の ロールベースアクセス制御 (RBAC)を使用して、ユーザーがデータベースで実行できるアクションへのアクセスを制限できます。RBAC は、1 つ以上のロールをユーザーに付与することで機能します。これらのロールは、ユーザーがデータベースリソースに対して実行できるオペレーションを決定します。

Canvas ユーザーとして、ユーザー名とパスワードの認証情報を使用して Amazon DocumentDB データベースに接続します。Amazon DocumentDB RBAC 機能を使用して、特定のデータベースへの読み取りアクセス権を持つ Canvas ユーザーのデータベースユーザー/ロールを作成できます。

たとえば、`createUser` を使用します:

```
db.createUser({
user: "canvas_user",
pwd: "<insert-password>",
roles: [{role: "read", db: "sample-database-1"}]
})
```

これにより、`sample-database-1` データベースへの読み取りアクセス許可を持つ `canvas_user` が作成されます。Canvas アナリストは、この認証情報を使用して、Amazon DocumentDB クラスター内のデータにアクセスできます。詳細については、「ロールベースのアクセス制御 (RBAC) を使用したデータベースへのアクセス」を参照してください。

利用できるリージョン

ノーコード統合は、Amazon DocumentDB と Amazon SageMaker AI Canvas の両方がサポートされているリージョンで使用できます。対象リージョンには以下が含まれます。

- us-east-1 (バージニア北部)
- us-east-2 (オハイオ)
- us-west-2 (オレゴン)
- ap-northeast-1 (東京)
- ap-northeast-2 (ソウル)
- ap-south-1 (ムンバイ)
- ap-southeast-1 (シンガポール)
- ap-southeast-2 (シドニー)
- eu-central-1 (フランクフルト)
- eu-west-1 (アイルランド)

最新のリージョンの可用性については、Amazon SageMaker AI デベロッパーガイド」の「Amazon SageMaker AI Canvas」を参照してください。 Amazon SageMaker

Amazon DocumentDB のベクトル検索

ベクトル検索は、距離または類似度メトリクスを使用して複数のベクトル表現を比較することで、特定のデータポイントに類似したデータポイントを見つけるための機械学習手法です。2 つのベクトルがベクトル空間内で互いに近い距離にあるほど、基盤となる項目が類似していると見なされます。この手法は、データのセマンティックな意味を把握するのに役立ちます。こうしたアプローチは、お勧め提案システム、自然言語処理、画像認識など、さまざまなアプリケーションで役立ちます。

Amazon DocumentDB のベクトル検索機能は、JSON ベースのドキュメントデータベースの柔軟性と豊富なクエリ機能にベクトル検索ならではのメリットを組み合わせています。既存の Amazon DocumentDB データや柔軟なドキュメントデータ構造を使用して、セマンティック検索エクスペリエンス、お勧め製品の提案、パーソナライゼーション、チャットボット、不正検出、異常検出などの機械学習や生成 AI のユースケースを構築するには、Amazon DocumentDB のベクトル検索が最適です。ベクトル検索は、Amazon DocumentDB 5.0 のインスタンスベースのクラスターで使用できます。

トピック

- ベクトルの挿入
- ベクトルインデックスの作成
- インデックス定義の取得
- ベクトルのクエリ
- 特徴量と制限事項
- ベストプラクティス

ベクトルの挿入

Amazon DocumentDB データベースへのベクトル挿入には、既存の挿入方法を使用できます。

例

次の例では、テストデータベース内に 5 つのドキュメントのコレクションが作成されています。各ドキュメントには、製品名とそれに対応するベクトル埋め込みの 2 つのフィールドが含まれています。

```
db.collection.insertMany([
  {"product_name": "Product A", "vectorEmbedding": [0.2, 0.5, 0.8]},
  {"product_name": "Product B", "vectorEmbedding": [0.7, 0.3, 0.9]},
```

```
  {"product_name": "Product C", "vectorEmbedding": [0.1, 0.2, 0.5]},
  {"product_name": "Product D", "vectorEmbedding": [0.9, 0.6, 0.4]},
  {"product_name": "Product E", "vectorEmbedding": [0.4, 0.7, 0.2]}
]);
```

ベクトルインデックスの作成

Amazon DocumentDB は、階層ナビゲーションスモールワールド方式 (HNSW) によるインデックス作成と、フラット圧縮付き転置ファイル方式 (IVFFlat) によるインデックス作成の両方をサポートしています。IVFFlat インデックスは、ベクトルを複数のリストに分離したうえで、クエリベクトルに最も近いリストの選択されたサブセットを検索します。一方、HNSW インデックスはベクトルデータを多層グラフに整理します。HNSW は IVFFlat に比べてビルド時間が多くかかりますが、クエリのパフォーマンスと再現性が向上します。IVFFlat とは異なり、HNSW にはトレーニングステップがないため、初期データロードなしでインデックスを生成できます。ほとんどのユースケースでは、ベクトル検索に HNSW インデックスタイプを使用することをお勧めします。

ベクトルインデックスを作成しない場合、Amazon DocumentDB は高精度な最近傍探索を実行し、完全な再現性を保証します。ただし、本番稼働シナリオでは、速度も重要となります。そのためベクトルインデックスを使用することをお勧めします。ベクトルインデックスは、再現性を若干犠牲にしても速度を向上させる効果があります。なお、ベクトルインデックスの追加によりクエリ結果が異なる可能性があることに留意してください。

テンプレート

次の `createIndex` または `runCommand` テンプレートを使用して、ベクトルフィールドにベクトルインデックスを構築できます。

Using createIndex

> mongosh や Java などの特定のドライバーでは、`vectorOptions` のパラメータを `createIndex` に使用するとエラーが発生する可能性があります。このような場合は、`runCommand` を使用することをお勧めします。

```
db.collection.createIndex(
  { "<vectorField>": "vector" },
  { "name": "<indexName>",
    "vectorOptions": {
      "type": " <hnsw> | <ivfflat> ",
      "dimensions": <number_of_dimensions>,
      "similarity": " <euclidean> | <cosine> | <dotProduct> ",
      "lists": <number_of_lists> [applicable for IVFFlat],
```

```
      "m": <max number of connections> [applicable for HNSW],
      "efConstruction": <size of the dynamic list for index build> [applicable for
 HNSW]
    }
  }
);
```

Using runCommand

mongosh や Java などの特定のドライバーでは、vectorOptions のパラメータを createIndex に使用すると エラーが発生する可能性があります。このような場合は、runCommand を使用することをお勧めします。

```
db.runCommand(
  { "createIndexes": "<collection>",
  "indexes": [{
      key: { "<vectorField>": "vector" },
      vectorOptions: {
          type: " <hnsw> | <ivfflat> ",
          dimensions: <number of dimensions>,
          similarity: " <euclidean> | <cosine> | <dotProduct> ",
          lists: <number_of_lists> [applicable for IVFFlat],
          m: <max number of connections> [applicable for HNSW],
          efConstruction: <size of the dynamic list for index build> [applicable for
 HNSW]
          },
      name: "myIndex"
      }]
  }
);
```

パラメータ	要件	データ型	説明	値 (複数可)
name	オプション	文字列	インデックスの名前を指定します。	英数字
type	オプション		エラーのタイプを指定します。	サポートされている: hnsw または ivfflat

パラメータ	要件	データ型	説明	値 (複数可)
				デフォルト: HNSW (エンジンパッチ 3.0.4574 以降)
dimensions	必須	integer	ベクトルデータ内のディメンション数を指定します。	最大ディメンション数 2,000 個。
similarity	必須	文字列	類似性の計算に使用される距離メトリクスを指定します。	• **euclidean** • **cosine** • **dotProduct**
lists	IVFFlat に必要です	integer	IVFFlat インデックスがベクトルデータをグループ化するために使用するクラスターの数を指定します。推奨される設定方式は、ドキュメント数最大 100 万件までについてはドキュメント 1,000 件当りのドキュメント数、ドキュメント数が最大 100 万件を超える場合には `sqrt(# of documents)` となります。	最小: 1 最大: 以下の 特徴量と制限事項 のインスタンスタイプごとのリストを参照してください。

パラメータ	要件	データ型	説明	値 (複数可)
m	オプション	integer	特定の HNSW インデックス向けの最大接続数を指定します。	デフォルト: 16 範囲: [2, 100]
efConstruction	オプション	integer	HNSW インデックスのグラフを作成するためのダイナミック候補リストのサイズを指定します。 efConstruction の値は、(2 × m) 以上にする必要があります。	デフォルト: 64 範囲 [4, 1000]

IVFFlat および m 向け lists や、HNSW 向け efConstruction などのサブパラメータの値を適切に設定することが重要です。検索の精度/再現性、ビルド時間、パフォーマンスに影響するためです。リスト値を大きくすると、リスト 1 件当りのベクトル数が減るため、クエリの速度が上がり、リージョンが小さくなります。一方、リージョンサイズが小さくなるほど再現性エラーが多くなり、精度が低下する恐れがあります。HNSW の場合、m や efConstruction の値を上げると検索精度が高まりますが、インデックスのビルド時間とサイズも増加します。以下の例を参照してください。

例

HNSW

```
db.collection.createIndex(
  { "vectorEmbedding": "vector" },
  { "name": "myIndex",
    "vectorOptions": {
      "type": "hnsw",
      "dimensions": 3,
```

```
      "similarity": "euclidean",
      "m": 16,
      "efConstruction": 64
    }
  }
);
```

IVFFlat

```
db.collection.createIndex(
  { "vectorEmbedding": "vector" },
  { "name": "myIndex",
    "vectorOptions": {
      "type": "ivfflat",
      "dimensions": 3,
      "similarity": "euclidean",
      "lists":1
    }
  }
)
```

インデックス定義の取得

getIndexes コマンドを使用して、ベクトルインデックスを含むインデックスの詳細を表示できます。

例

```
db.collection.getIndexes()
```

出力例

```
[
 {
  "v" : 4,
  "key" : {
   "_id" : 1
  },
  "name" : "_id_",
  "ns" : "test.collection"
```

```
 },
 {
  "v" : 4,
  "key" : {
   "vectorEmbedding" : "vector"
  },
  "name" : "myIndex",
  "vectorOptions" : {
   "type" : "ivfflat",
   "dimensions" : 3,
   "similarity" : "euclidean",
   "lists" : 1
  },
  "ns" : "test.collection"
 }
]
```

ベクトルのクエリ

ベクトルクエリテンプレート

次のテンプレートを使用して、ベクトルをクエリします。

```
db.collection.aggregate([
  {
    $search: {
      "vectorSearch": {
        "vector": <query vector>,
        "path": "<vectorField>",
        "similarity": "<distance metric>",
        "k": <number of results>,
        "probes":<number of probes> [applicable for IVFFlat],
        "efSearch":<size of the dynamic list during search> [applicable for HNSW]
      }
    }
  }
]);
```

パラメータ	要件	タイプ	説明	値 (複数可)
vectorSearch	必須	オペレーター	$search コマンド内でベクトル	

パラメータ	要件	タイプ	説明	値 (複数可)
			をクエリする際に使用します。	
vector	必須	配列	類似ベクトルの検索に使用するクエリベクトルを示します。	
path	必須	文字列	ベクトルフィールドの名前を定義します。	
k	必須	integer	検索から返される結果の最大数を指定します。	
similarity	必須	文字列	類似性の計算に使用される距離メトリクスを指定します。	• **euclidean** • **cosine** • **dotProduct**

パラメータ	要件	タイプ	説明	値 (複数可)
probes	オプション	integer	ベクトル検索において検査対象とするクラスターの数。値を大きくすると、速度が低下する代わりに再現性が向上します。これは、最近傍検索のリスト数に設定できます (この時点では、プランナーはインデックスを使用しません)。微調整の開始点とするための推奨設定は `sqrt(# of lists)` です。	デフォルト: 1
efSearch	オプション	integer	HNSW インデックスが検索時に使用するダイナミック候補リストのサイズを指定します。`efSearch` の値が高いほど、速度は低下しますが再現性が向上します。	デフォルト: 40 範囲: [1, 1000]

必要なパフォーマンスと精度を実現するには、 efSearch (HNSW) または probes (IVFFlat) の値を微調整することが重要です。次のオペレーションの例を参照してください。

HNSW

```
db.collection.aggregate([
  {
    $search: {
      "vectorSearch": {
        "vector": [0.2, 0.5, 0.8],
        "path": "vectorEmbedding",
        "similarity": "euclidean",
        "k": 2,
        "efSearch": 40
      }
    }
  }
]);
```

IVFFlat

```
db.collection.aggregate([
  {
    $search: {
      "vectorSearch": {
        "vector": [0.2, 0.5, 0.8],
        "path": "vectorEmbedding",
        "similarity": "euclidean",
        "k": 2,
        "probes": 1
      }
    }
  }
]);
```

出力例

このオペレーションによる出力は、次のようになります。

```
{ "_id" : ObjectId("653d835ff96bee02cad7323c"), "product_name" : "Product A",
 "vectorEmbedding" : [ 0.2, 0.5, 0.8 ] }
```

```
{ "_id" : ObjectId("653d835ff96bee02cad7323e"), "product_name" : "Product C",
 "vectorEmbedding" : [ 0.1, 0.2, 0.5 ] }
```

特徴量と制限事項

バージョン互換性

- Amazon DocumentDB のベクトル検索は、Amazon DocumentDB 5.0 インスタンスベースのクラスターでのみ使用できます。

ベクトル

- Amazon DocumentDB は、最大 2,000 個のディメンションのベクトルをインデックス化できます。ただし、インデックスなしであれば最大 16,000 個のディメンションを保存できます。

インデックス

- IVFFlat インデックス作成の場合、lists パラメータの推奨設定方法は、ドキュメント数最大 100 万件までについては 1000 件あたりのドキュメント数、ドキュネント数が最大 100 万を超える場合には `sqrt(# of documents)` となります。作業メモリの制限により、Amazon DocumentDB はディメンション数に応じて、lists パラメータの特定の最大値をサポートします。参考までに、次の表は、ディメンション数が 500、1000、2,000 の場合のベクトルのリストパラメータの最大値を示します。

インスタンスタイプ	ディメンション数 500 の場合のリスト	ディメンション数 1000 の場合のリスト	ディメンション数 2000 の場合のリスト
t3.med	372	257	150
r5.l	915	741	511
r5.xl	1,393	1,196	901
r5.2xl	5,460	5,230	4,788
r5.4xl	7,842	7,599	7,138
r5.8xl	11,220	10,974	10,498

インスタンスタイプ	ディメンション数 500 の場合のリスト	ディメンション数 1000 の場合のリスト	ディメンション数 2000 の場合のリスト
r5.12xl	13,774	13,526	13,044
r5.16xl	15,943	15,694	15,208
r5.24xl	19,585	19,335	18,845

- compound、sparse、partial などの他のインデックスオプションは、ベクトルインデックスではサポートされていません。
- HNSW インデックスでは、並列インデックスビルドはサポートされていません。IVFFlat インデックスでのみサポートされています。

ベクトルクエリ

- ベクトル検索クエリでは、最適な結果を得るために、probes や efSearch などのパラメータを微調整することが重要です。probes または efSearch パラメータの値が高いほど、再現性は高くなりますが、速度が低下します。検索パラメータの微調整の開始点としての推奨設定は sqrt(# of lists) です。

ベストプラクティス

Amazon DocumentDB でベクトル検索を使用するためのベストプラクティスについて説明します。新しいベストプラクティスが確認されると、このセクションは更新されます。

- フラット圧縮付き転置ファイル方式 (IVFFlat) によるインデックス作成には、類似点に基づいてデータポイントをクラスター化および整理することが含まれます。したがって、インデックスをより効果的に運用するには、インデックスを作成する前に少なくとも一部のデータをロードしてみることをお勧めします。
- ベクトル検索クエリでは、最適な結果を得るために、probes や efSearch などのパラメータをファインチューニングすることが重要です。probes または efSearch パラメータの値が高いほど、再現性は高くなりますが、速度が低下します。probes パラメータの微調整の開始点としての推奨設定は sqrt(lists) です。

リソース

- ベクトル検索に関する新しいブログ記事
- セマンティック検索コードのサンプル
- Amazon DocumentDB ベクトル検索コードのサンプル

Amazon DocumentDB への移行

Amazon DocumentDB は、MongoDB API と互換性のあるフルマネージド型のデータベースサービスです。このセクションで詳しく説明するプロセスを使用して、オンプレミスまたは Amazon Elastic Compute Cloud (Amazon EC2)上で実行されている MongoDB データベースから Amazon DocumentDB にデータを移行できます。

トピック

- AWS Database Migration Service (DMS): クイックスタートガイドを使用して Amazon DocumentDB に移行する
- Amazon DocumentDB 移行ランブック

AWS Database Migration Service (DMS): クイックスタートガイドを使用して Amazon DocumentDB に移行する

トピック

- DMS ソースを準備する
- DMS のセットアップ
- DocumentDB 圧縮を有効にする
- レプリケーションタスクを作成する
- 進行状況をモニタリングする
- 追加情報

DMS ソースを準備する

DocumentDB 変更ストリーム変更ストリームの有効化を有効にするか、MongoDB Oplog が DMS 変更データキャプチャ (CDC) をサポートできるようにするには、「」を参照してください。

- DMS ソースは、含まれているすべてのコレクションについて DMS のフルロードが完了するまで、進行中のすべての変更を保持する必要があります。
- DocumentDB 変更ストリームは時間ベースです。`change_stream_log_retention_duration` 設定が全ロードを完了する時間をカバーするのに十分な大きさであることを確認します。

- MongoDB Oplog は固定サイズです。全ロード中にすべてのオペレーションを保持するサイズであることを確認します。

DMS のセットアップ

DMS インスタンス、ソース、ターゲットエンドポイントを作成し、各エンドポイントをテストします。

DocumentDB 圧縮を有効にする

カスタムパラメータグループを DocumentDB クラスターにアタッチし、default_collection_compression パラメータを有効に更新して、圧縮を有効にします。詳細については「コレクションレベルのドキュメント圧縮の管理」を参照してください。

レプリケーションタスクを作成する

1. DMS コンソールのナビゲーションペインで、移行またはレプリケートを選択し、タスクを選択します。
2. [Create task] (タスクの作成) を選択します。
3. 「タスクの作成」ページの「タスク設定」セクションで、次の操作を行います。
   - 一意で意味のあるタスク識別子 (mongodb-docdb-replication」など) を入力します。
   - ソースデータベースエンドポイントドロップダウンメニューで、以前に作成したソースエンドポイントを選択します。
   - ターゲットデータベースエンドポイントドロップダウンメニューで、以前に作成したターゲットエンドポイントを選択します。
   - タスクタイプで、移行とレプリケートを選択します。
4. 設定セクションで、次の操作を行います。
   - タスクログの場合は、CloudWatch ログをオンにするチェックボックスをオンにします。
   - 編集モード (セクションの上部) で、JSON エディタを選択し、次の属性を設定します。
     - `ParallelApplyThreads` を 5 (の下`TargetMetadata`) に設定します。これにより、CDC で 1 秒あたり最大 1000 回のinsert/update/deleteオペレーションが可能になります。
     - `MaxFullLoadSubTasks` を 16 (の下) に設定します`FullLoadSettings`。インスタンスのサイズに応じてこれを増やすことを検討してください。

- 大規模なコレクション (100 GB 以上) の場合、自動パーティションを有効にします (テーブルマッピングと `parallel-load` 属性の下) 。
  - 「type」 : 「partitions-auto」
  - number-of-partitions」: 16

進行状況をモニタリングする

AWS DMS コンソールを使用するか、カスタムダッシュボード (ダッシュボードツール) を作成して移行を追跡します。次のメトリクスに焦点を当てます。

- FullLoadThroughputBandwidthTarget — 移行の全ロードフェーズ中にターゲットデータベースにデータを転送するときに DMS が使用するネットワーク帯域幅 (KB/秒) を測定します。
- CDCLatencyTarget — ソースデータベースで変更が発生し、その変更がターゲットデータベースに適用されたときの遅延 (秒単位) を測定します。
- CDCThroughputRowsTarget — 移行の継続的なレプリケーションフェーズ中に DMS がターゲットデータベースに適用している 1 秒あたりの行数を測定します。

追加情報

Amazon DocumentDB および の詳細については AWS DMS、「」を参照してください。

- Amazon DocumentDB 移行ランブック
- MongoDB から Amazon DocumentDB への移行

Amazon DocumentDB 移行ランブック

このランブックは、 AWS Database Migration Service (DMS) を使用して MongoDB データベースを Amazon DocumentDB に移行するための包括的なガイドを提供します。これは、最初の検出から移行後の検証まで、end-to-endの移行ジャーニーを通じてデータベース管理者、クラウドエンジニア、開発者をサポートするように設計されています。

MongoDB と Amazon DocumentDB の実装とサポートされている機能の違いを考慮すると、このランブックは構造化された体系的なアプローチを強調しています。重要な移行前評価の概要、互換性に関する考慮事項、および最小限の中断で移行を成功させるために必要な主要なタスクについて詳しく説明します。

ランブックは、以下のトピックで構成されています。

- 互換性 — Amazon DocumentDB でサポートされている MongoDB 機能とデータ型を理解し、潜在的な非互換性を特定します。
- ワークロードの検出 — 読み取り/書き込みパターン、データボリューム、パフォーマンスベースラインなど、既存の MongoDB ワークロードを分析します。
- インデックスの移行 — Amazon DocumentDB MongoDB インデックスを抽出および変換するための戦略を分析します。
- ユーザー移行 — データベースユーザー、ロール、アクセスコントロールを Amazon DocumentDB に移行するためのアプローチについて詳しく説明します。
- データ移行 — 全ロードおよび変更データキャプチャ (CDC) など AWS DMS、 を使用したデータ移行のさまざまな方法について説明します。
- モニタリング — DMS またはネイティブツールを使用して移行する際のさまざまなモニタリングアプローチについて詳しく説明します。
- 検証 — 移行後のデータ整合性チェック、機能検証、パフォーマンス比較の手順を提供します。

このランブックのガイダンスに従うことで、チームはアプリケーションの機能を維持し、リスクを最小限に抑えながら、Amazon DocumentDB へのスムーズで安全で効率的な移行を確保できます。

互換性

トピック

- コア機能の互換性
- Amazon DocumentDB 互換性評価ツール

MongoDB から Amazon DocumentDB に移行する場合、移行を成功させるには、徹底的な初期評価と機能の互換性チェックが不可欠です。このプロセスは、集約パイプライン演算子、クエリパターン、インデックス、データモデルなど、MongoDB 機能の包括的なインベントリから始まります。

Amazon DocumentDB は MongoDB 3.6、4.0、5.0 API と互換性があるため、新しい MongoDB 固有の機能を使用するアプリケーションではリファクタリングが必要になる場合があります。評価する重要な領域には、シャーディングメカニズム (Amazon DocumentDB は異なるアプローチを使用) 、トランザクション実装、変更ストリーム機能、インデックスタイプ (特にスパースインデックスと部分インデックス) などがあります。

パフォーマンス特性も異なり、Amazon DocumentDB は予測可能なパフォーマンスを持つエンタープライズワークロード向けに最適化されています。テストでは、両方のシステムに対して代表的なワークロードを実行して、最適化が必要なクエリパターンを特定する必要があります。

評価フェーズでは、潜在的なパフォーマンスギャップを検出するために実行計画をモニタリングすることが重要です。これにより、明確な移行ロードマップを作成し、必要なアプリケーションの変更を特定し、スムーズな移行のための現実的なタイムラインを確立できます。

コア機能の互換性

包括的な機能サポート

- CRUD オペレーション — 一括演算子やクエリ演算子など、すべての基本的な作成、読み取り、更新、削除オペレーションを完全にサポートし、シームレスなアプリケーションの互換性を実現します。
- 豊富なインデックス作成機能 — 単一フィールド、複合、TTL、部分インデックス、スパースインデックス、2dsphere インデックスの包括的なサポートを活用して、テキストベースのルックアップのクエリパフォーマンスとテキストインデックス (バージョン 5) を最適化します。
- エンタープライズグレードのレプリケーション — リードレプリカを備えた堅牢な自動フェイルオーバーメカニズムにより、運用上のオーバーヘッドなしで優れた高可用性を実現できます。
- 高度なバックアップソリューション — データ保護のためのPoint-in-Timeリカバリ (PITR) とオンデマンドの手動スナップショットを備えた自動バックアップシステムにより、簡単にバックアップできます。

拡張 AWS統合機能

- 効率的な集約 — エンタープライズワークロード向けに最適化されたパフォーマンスで、最も一般的に使用される集約ステージ (`$match`、`$group`、`$sort`、、 `$project`など) を活用します。
- トランザクションのサポート — マルチドキュメントトランザクションとマルチコレクショントランザクションを実装し、ほとんどのビジネスアプリケーションのニーズに最適です。
- リアルタイムデータ追跡 — シンプルなコマンドで変更ストリームを有効にし、リアルタイムのデータ変更モニタリング用のシンプルなパラメータグループ設定で変更ストリームの保持期間を延長します。
- 位置情報ベースのサービス — `$geoNear`オペレーターインデックスと 2dsphere インデックスをサポートする地理空間アプリケーションを実装します。

- テキスト検索機能 — コンテンツ検出のニーズに応じて、組み込みのテキスト検索機能を使用します。

最新のアーキテクチャの利点

- クラウドネイティブ設計 — MapReduce AWSなどのレガシー機能を、より効率的な集約パイプラインオペレーションに置き換える、エンゲージ最適化アーキテクチャ。
- セキュリティの強化 — AWS Identity and Access Management (IAM)、SCRAM-SHA-1, SCRAM-SHA-256、X.509 証明書認証、パスワードベースの認証の利点があります。
- 予測可能なパフォーマンス — エンタープライズワークロード専用に最適化された一貫したパフォーマンスを体験できます。

Amazon DocumentDB の機能の包括的な概要については、Amazon DocumentDB でサポートされている MongoDB API、オペレーション、およびデータ型「」および機能の違い: Amazon DocumentDB と MongoDB「」を参照して、データベースの可能性を最大化してください。

Amazon DocumentDB は、MongoDB が提供するすべてのインデックスをサポートしているわけではありません。互換性を確認するための無料のインデックスツールが用意されています。インデックスツールを実行して非互換性を評価し、それに応じて回避策を計画することをお勧めします。

Amazon DocumentDB 互換性評価ツール

MongoDB から Amazon DocumentDB への互換性ツールは、GitHub で利用可能なオープンソースユーティリティです。MongoDB ログまたはアプリケーションのソースコードを分析することで Amazon DocumentDB との MongoDB ワークロードの互換性を評価するのに役立ちます。

主な特徴

- ワークロードの MongoDB API 使用パターンを識別します
- 移行前に潜在的な互換性の問題にフラグを付けます
- レコメンデーションを含む詳細な互換性レポートを生成します
- ローカルで実行できるスタンドアロンユーティリティとして利用可能

評価方法

ログベースの評価

- メリット:
  - 実際のランタイム動作とクエリパターンをキャプチャします
  - 実際の使用頻度とパフォーマンス特性を特定します
  - ソースコードに表示されない可能性のある動的クエリを検出します
  - アプリケーションのソースコードへのアクセスは不要
- デメリット:
  - プロファイリングを有効にして MongoDB ログにアクセスする必要があります
  - ログ記録期間中に発生したオペレーションのみをキャプチャします
  - 使用頻度の低い機能や季節的なワークロードを見逃す可能性がある

ソースコード分析

- メリット:
  - コードベースで考えられるすべての MongoDB オペレーションの包括的なカバレッジ
  - ほとんど実行されないコードパスの問題を特定できる
  - Amazon DocumentDB の違いの影響を受ける可能性のあるクライアント側のロジックを検出します
  - 評価を実行するためにアプリケーションを実行する必要がない
- デメリット:
  - 存在するが本番環境で実行されないコードにフラグを付ける可能性がある
  - 完全なアプリケーションソースコードにアクセスする必要があります
  - 動的に構築されたクエリを分析する機能が限られている

最良の結果を得るには、移行前に互換性の課題の全体像を把握するために、可能であれば両方の評価方法を使用することをお勧めします。

ワークロードの検出

MongoDB から Amazon DocumentDB に移行するには、既存のデータベースワークロードを完全に理解する必要があります。ワークロード検出は、データベースの使用パターン、データ構造、クエリパフォーマンス、運用上の依存関係を分析し、中断を最小限に抑えながらシームレスな移行を確保するプロセスです。このセクションでは、MongoDB から Amazon DocumentDB への効果的な移行を容易にするためのワークロード検出に関連する主要なステップの概要を説明します。

トピック

- 既存の MongoDB デプロイの評価
- データモデルの違いを特定する
- クエリとパフォーマンスの分析
- セキュリティとアクセスコントロールのレビュー
- 運用とモニタリングに関する考慮事項

既存の MongoDB デプロイの評価

移行前に、以下を含む現在の MongoDB 環境を評価することが重要です。

- クラスターアーキテクチャ — ノード、レプリカセット、シャーディング設定の数を特定します。MongoDB から Amazon DocumentDB に移行する場合、Amazon DocumentDB はユーザー制御のシャーディングをサポートしていないため、MongoDB シャーディング設定を理解することが重要です。 Amazon DocumentDB シャーディングされた MongoDB 環境用に設計されたアプリケーションでは、Amazon DocumentDB がストレージベースのアーキテクチャで異なるスケーリングアプローチを使用するため、アーキテクチャの変更が必要になります。Amazon DocumentDB に移行するときは、データ分散戦略を適応させ、場合によってはシャードコレクションを統合する必要があります。
- ストレージとデータボリューム — クラスターの合計データサイズとインデックスサイズを測定します。これを Oplog レビューツールで補完し、書き込みパターンとデータの増加速度を理解します。クラスターのサイズ設定の詳細については、「」を参照してくださいインスタンスのサイズ指定。
- ワークロードパターン — 読み取りおよび書き込みスループット、クエリ実行頻度、インデックス作成効率を分析します。
- 運用上の依存関係 — MongoDB に依存するすべてのアプリケーション、サービス、統合を文書化します。

データモデルの違いを特定する

Amazon DocumentDB は MongoDB と互換性がありますが、サポートされている機能には次のような違いがあります。

- トランザクション — Amazon DocumentDB は ACID トランザクションをサポートしますが、一部の を使用します制限。

- スキーマ設計 — ドキュメント構造、埋め込みドキュメント、リファレンスが Amazon DocumentDB のベストプラクティスと一致していることを確認します。

クエリとパフォーマンスの分析

クエリの動作を理解することで、移行と移行後のパフォーマンスを最適化できます。分析すべき主な領域は次のとおりです。

- スロークエリ — MongoDB のプロファイリングツールを使用して、実行時間が長いクエリを特定します。
- クエリパターン — CRUD オペレーションや集計など、一般的なクエリタイプを分類します。
- インデックスの使用 — インデックスが Amazon DocumentDB で効果的に使用されているか、最適化が必要かを評価します。Amazon DocumentDB でインデックスの使用状況を評価し、パフォーマンスを最適化するには、`$indexStats`集約パイプラインステージを重要なクエリの `explain()`メソッドと組み合わせて使用します。まず`db.collection.aggregate([{$indexStats{}}])`、 を実行して、使用されているインデックスを特定します。を使用して最も頻繁にクエリを実行することで、より詳細な分析を行うことができます`explainPlan`。
- 同時実行とワークロードの分散 — 読み取りと書き込みの比率、接続プーリング、パフォーマンスのボトルネックを評価します。

セキュリティとアクセスコントロールのレビュー

認証と認可

- MongoDB RBAC を Amazon DocumentDB IAM および RBAC にマッピング — MongoDB のロールベースのアクセスコントロールユーザーとロールを AWS Identity and Access Management (IAM) ポリシーと Amazon DocumentDB SCRAM 認証ユーザーにマッピングします。
- ユーザー移行戦略 — データベースユーザー、カスタムロール、権限を Amazon DocumentDB でサポートされている認証メカニズムに移行する計画を立てます。
- 権限の違い — Amazon DocumentDB と同等のもの (クラスター管理ロールなど) を直接使用せずに MongoDB 権限を特定します。 Amazon DocumentDB
- アプリケーション認証 — Amazon DocumentDB のパスワードポリシーの接続文字列と認証情報管理を更新します。シークレットマネージャーを使用して、認証情報を保存し、パスワードを更新できます。

- サービスアカウント管理 — でサービスアカウントの認証情報を管理するプロセスを確立します AWS Secrets Manager。
- 最小特権の実装 — アクセスコントロールを確認して絞り込み、新しい環境に最小特権の原則を実装します。

Encryption

保管中および転送中の暗号化がコンプライアンス要件と一致していることを確認します。

ネットワーク構成

Virtual Private Cloud (VPC) のセットアップとセキュリティグループルールを計画します。

運用とモニタリングに関する考慮事項

システムの信頼性を維持するために、ワークロード検出には以下も含める必要があります。

- バックアップと復元戦略 — 既存のバックアップ方法と Amazon DocumentDB のバックアップ機能を評価します。
- AWS Backup 統合 — を活用して、Amazon DocumentDB を含む AWS のサービス全体で一元化されたバックアップ管理 AWS Backup を行います。
- CloudWatch メトリクス — MongoDB モニタリングメトリクスを、CPU、メモリ、接続、ストレージの Amazon DocumentDB CloudWatch メトリクスにマッピングします。
- Performance Insights — Amazon DocumentDB Performance Insights を実装して、データベースの負荷を視覚化し、詳細なクエリ分析でパフォーマンスの問題を分析します。
- プロファイラー — Amazon DocumentDB MongoDB のプロファイラーに似ていますが、Amazon DocumentDB 固有の設定) をキャプチャするように Amazon DocumentDB プロファイラーを設定します。
  - 適切なしきい値を持つパラメータグループを使用して を有効にします。
  - プロファイラーデータを分析して最適化の機会を特定する
- CloudWatch Events — Amazon DocumentDB クラスターイベントのイベント駆動型モニタリングを設定します。
  - バックアップイベント、メンテナンスウィンドウ、フェイルオーバーの通知を設定します。
  - アラートおよび自動応答のために Amazon SNS と統合 AWS Lambda します。
- 監査ログ記録 — ユーザーアクティビティとセキュリティ関連のイベントを追跡するための監査ログ記録設定を計画します。

- 拡張モニタリング — 1 秒間隔できめ細かな OS レベルのメトリクスの拡張モニタリングを有効にします。

インデックスの移行

MongoDB から Amazon DocumentDB への移行には、データだけでなくインデックスも転送してクエリパフォーマンスを維持し、データベースオペレーションを最適化する必要があります。このセクションでは、互換性と効率を確保しながら、MongoDB から Amazon DocumentDB にインデックスを移行するための詳細なstep-by-stepプロセスの概要を説明します。

Amazon DocumentDB インデックスツールの使用

インデックスツールのクローンを作成する

```
git clone https://github.com/aws-samples/amazon-documentdb-tools.git
cd amazon-documentdb-tools/index-tool
```

```
pip install -r requirements.txt
```

MongoDB からインデックスをエクスポートする (MongoDB から移行する場合)

```
python3 migrationtools/documentdb_index_tool.py --dump-indexes --dir
 mongodb_index_export --uri
'mongodb://localhost:27017'
```

Amazon DocumentDB からインデックスをエクスポートする (Amazon DocumentDB から移行する場合)

```
python3 migrationtools/documentdb_index_tool.py --dump-indexes --dir docdb_index_export
 --uri
'mongodb://user:password@mydocdb.cluster-cdtjj00yfi95.eu-west-
2.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=rds-combined-ca-
bundle.pem&replicaSet=rs0&retryWrites=false'
```

インデックスをインポートする

```
python3 migrationtools/documentdb_index_tool.py --restore-indexes --skip-incompatible
 --dir
mongodb_index_export --uri 'mongodb://user:password@mydocdb.cluster-cdtjj00yfi95.eu-
west-
```

```
2.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=rds-combined-ca-
bundle.pem&replicaSet=rs0&retryWrites=false'
```

インデックスの検証

```
python3 migrationtools/documentdb_index_tool.py --show-issues --dir
 mongodb_index_export
```

ユーザー移行

MongoDB から Amazon DocumentDB にユーザーを移行することは、アクセスコントロール、認証、データベースセキュリティを維持する上で不可欠です。このセクションでは、Amazon MongoDB ユーザーのロールとアクセス許可を保持しながら、MongoDB ユーザーを正常に移行するための詳細な手順について説明します。 Amazon DocumentDB

Amazon DocumentDB ユーザーエクスポートツールの使用

は、ユーザーとロールを MongoDB または Amazon DocumentDB から JavaScript ファイルに[Export Users tool]エクスポートし、それを別のクラスターで再作成するために使用できます。

前提条件

```
# Clone the repository
git clone https://github.com/awslabs/amazon-documentdb-tools.git
cd amazon-documentdb-tools/migration/export-users
```

```
# Install required dependencies
pip install pymongo
```

ステップ 1: ユーザーとロールをエクスポートする

```
# Export users and roles to JavaScript files
python3 docdbExportUsers.py \
--users-file mongodb-users.js \
--roles-file mongodb-roles.js \
--uri "mongodb://admin:password@source-host:27017/"
```

ステップ 2: ユーザーファイルを編集する

```
// Example of how to update the users.js file
// Find each user creation statement and add the password
db.getSiblingDB("admin").createUser({
user: "appuser",
// Add password here
pwd: "newpassword",
roles: [
{ role: "readWrite", db: "mydb" }
]
})
```

ステップ 3: カスタムロールを Amazon DocumentDB に復元する

```
# Import roles first
mongo --ssl \
--host target-host:27017 \
--sslCAFile rds-combined-ca-bundle.pem \
--username admin \
--password password \
mongodb-roles.js
```

ステップ 4: Amazon DocumentDB にユーザーを復元する

```
# Import users after roles are created
mongo --ssl \
--host target-host:27017 \
--sslCAFile rds-combined-ca-bundle.pem \
--username admin \
--password password \
mongodb-users.js
```

重要な注意事項

- セキュリティ上の理由からパスワードはエクスポートされないため、users.js ファイルに手動で追加する必要があります。
- ロールを適切に割り当てるには、ユーザーの前にロールをインポートする必要があります。
- このツールは、mongo シェルで直接実行できる JavaScript ファイルを生成します。
- カスタムロールとその権限は移行中も保持されます。
- このアプローチにより、インポート前にユーザーアクセス許可を確認および変更できます。

この方法は、MongoDB から Amazon DocumentDB にユーザーとロールを移行するための安全で柔軟なアプローチを提供し、移行プロセス中にパスワードをリセットできるようにします。

データ移行

トピック

- オンライン移行
- オフライン移行
- 前提条件
- Amazon DocumentDB クラスターを準備する
- データダンプを実行する (mongodump)
- ダンプファイルを復元環境に転送する
- Amazon DocumentDB (mongorestore) へのデータの復元

オンライン移行

このセクションでは、最小限のダウンタイムと継続的なレプリケーションを可能にする AWS DMS ために、 を使用して MongoDB から Amazon DocumentDB へのオンライン移行を実行する詳細な手順について説明します。まず、Amazon DocumentDB クラスターをターゲットとして設定し、MongoDB インスタンスがソースとして適切に設定されていることを確認します。通常、変更データキャプチャにはレプリカセットモードが必要です。次に、DMS レプリケーションインスタンスを作成し、必要な接続の詳細を使用してソースエンドポイントとターゲットエンドポイントを定義します。エンドポイントを検証したら、フルデータロード、継続的なレプリケーション、またはその両方を含む移行タスクを設定して開始します。

ターゲットを設定する (Amazon DocumentDB)

> **Note**
>
> 移行する Amazon DocumentDB クラスターを既にプロビジョニングしている場合は、このステップをスキップできます。

カスタムパラメータグループを作成する

「」の AWS Management Console 「」または AWS CLI 「」の手順を参照してください Amazon DocumentDB クラスターパラメータグループを作成する。

Amazon DocumentDB クラスターの作成

> **Note**
>
> このガイドには Amazon DocumentDB クラスターを作成する他の手順がありますが、このセクションの手順は、特に大量のデータを新しいクラスターに移行するタスクに適用されます。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン
   > (≡)
   > を選択します。

3. Amazon DocumentDB マネジメントコンソールで、[クラスター] の下にある [作成] を選択します。
4. Amazon DocumentDB クラスターの作成ページのクラスタータイプセクションで、インスタンスベースのクラスターを選択します (これはデフォルトのオプションです)。
5. クラスター設定セクションで、次の操作を行います。
   - [クラスター識別子] には、**`mydocdbcluster`** などの一意の名称を入力します。コンソールは、入力方法に関係なく、すべてのクラスター名を小文字に変更することに注意してください。
   - エンジンバージョンで、5.0.0 を選択します。
6. クラスターストレージ設定セクションで、Amazon DocumentDB 標準設定をそのままにします (これはデフォルトのオプションです)。
7. インスタンス設定 セクション:
   - DB インスタンスクラスの場合は、メモリ最適化クラス (r クラスを含む) を選択します (デフォルト)。
   - インスタンスクラス では、ワークロードに基づいてインスタンスクラスを選択します。例:

- db.r6g.large: 小規模なワークロード向け
- db.r6g.4xlarge: 大規模なワークロード向け

ベストプラクティスとして、最適なフルロードスループットを得るには、インスタンスを可能な限り大きく選択し、移行が完了したらスケールダウンすることをお勧めします。

- インスタンス数 で、インスタンスを 1 つ選択します。1 つのインスタンスを選択すると、コストを最小限に抑えることができます。フルロード移行が完了したら、高可用性のために 3 つのインスタンスにスケールすることをお勧めします。

8. 認証セクションで、プライマリユーザーのユーザー名を入力し、セルフマネージドを選択します。パスワードを入力して確認します。
9. ネットワーク設定セクションで、VPC とサブネットグループを選択し、VPC セキュリティグループを設定します。インバウンドルールを更新して、Amazon DocumentDB セキュリティグループが DMS インスタンスのセキュリティグループからのインバウンド接続を許可していることを確認します。
10. Encryption-at-restセクションで、暗号化を有効にし (推奨) 、KMS キーを選択または入力します。
11. Backup セクションで、バックアップ保持期間 (1 ~ 35 日) を設定します。
12. 設定を確認し、クラスターの作成を選択します。

    デプロイには通常 10 ~ 15 分かかります。

ソースを設定する

MongoDB と Amazon DocumentDB は、シナリオに応じて、どちらも移行ソースとして機能します。

- ソースとしての MongoDB — オンプレミスまたはセルフマネージド MongoDB から Amazon DocumentDB または他の AWS データベースサービスに移行する場合に一般的です。移行中の変更データキャプチャをサポートするために、適切なサイズのオポログを使用してレプリカセットモードで実行する必要があります (フルロード中にすべてのオペレーションを保持するサイズになっていることを確認してください) 。
- ソースとしての Amazon DocumentDB — 通常、クロスリージョンレプリケーション、バージョンアップグレード、MongoDB Atlas などの他のデータベースサービスへの移行に使用されます。クラスターchange_stream_log_retention_durationパラメータグループの パラメータを設定して、移行中に進行中の変更をキャプチャ変更ストリームの有効化する必要がありま

す。change_stream_log_retention_duration 設定がフルロードの完了に必要な時間をカバーするのに十分な大きさであることを確認します。

移行を開始する前に、 AWS DMS アクセスを許可するようにソースを設定します。

適切なアクセス許可を持つ MongoDB ユーザーを作成します。

```
db.createUser({
user: "dmsUser",
pwd: "yourSecurePassword",
roles: [{ role: "readAnyDatabase", db: "admin" }]
})
```

ネットワークと認証を設定します。

MongoDB から DMS への移行用のネットワーク接続を設定する場合：

EC2-hosted MongoDB ソース

- EC2 セキュリティグループを変更して、DMS レプリケーションインスタンスのセキュリティグループからのインバウンドトラフィックを許可します。
- TCP ポート 27017 (またはカスタム MongoDB ポート) のルールを追加します。
- DMS レプリケーションインスタンスのセキュリティグループ ID を正確なアクセスコントロールのソースとして使用します。
- EC2 インスタンスのサブネットに DMS レプリケーションインスタンスのサブネットへのルートがあることを確認します。

オンプレミス MongoDB ソース

- DMS レプリケーションインスタンスのパブリック IP アドレスからのインバウンド接続を許可するようにファイアウォールを設定します。
- AWS Direct Connect または VPN を使用している場合は、ネットワークと DMS インスタンスを含む VPC 間の適切なルーティングを確認してください。
- DMS サブネットから MongoDB サーバーへの telnet または nc コマンドを使用して接続をテストします。

MongoDB Atlas ソース

- DMS レプリケーションインスタンスの IP アドレスを MongoDB Atlas IP 許可リストに追加します。
- Atlas が実行されている場合は AWS 、VPC と MongoDB Atlas VPC 間の VPC ピアリングを設定します AWS。
- 別のクラウドプロバイダーで実行されている場合は、プライベート接続 (エンタープライズ階層) AWS PrivateLink を設定します。
- 適切な読み取り/書き込みアクセス許可を持つ専用ユーザーを作成します。
- SSL モードを「verify-full」に設定して MongoDB Atlas 接続文字列を使用します。
- 移行期間に十分なオプログサイズを確保します。

Amazon DocumentDB ソース

DMS レプリケーションインスタンスのセキュリティグループからのインバウンドトラフィックを許可するように、ソース Amazon DocumentDB セキュリティグループを設定します。

DMS レプリケーションインスタンスを作成する

最適な DMS 設定とインスタンスサイズで最適な移行インフラストラクチャを作成するため、DMS Buddy を使用して DMS インフラストラクチャをプロビジョニングすることをお勧めします。

を手動で設定する場合は、次の手順に従います。

1. DMS AWS コンソールを開き、レプリケーションインスタンスの作成を選択します。
2. レプリケーションインスタンスの詳細を入力します。
    - インスタンス名: 一意の名前を選択します。
    - インスタンスクラス: ワークロードに基づいて選択します。例: dms.r5.large (小規模なワークロード) 、dms.r5.4xlarge (大規模なワークロード) 。
    - エンジンバージョン: 3.5.4
    - 割り当てられたストレージ: デフォルトは 100GB (必要に応じて増加) です。これは、ドキュメントのサイズ、更新/秒、全ロード時間によって決まります。
    - マルチ AZ 配置: 必要に応じて高可用性を有効にします。
    - Amazon DocumentDB と同じ VPC を選択します。
    - セキュリティグループがソースと Amazon DocumentDB からのインバウンドトラフィックを許可していることを確認します。

3. レプリケーションインスタンスの作成をクリックし、ステータスが使用可能になるまで待ちます。

DMS エンドポイントを作成する

ソースエンドポイントを作成する

MongoDB ソースの場合

1. DMS コンソールのナビゲーションペインで、移行またはレプリケートを選択し、エンドポイントを選択します。
2. [エンドポイントの作成] を選択します。
3. エンドポイントの作成ページで、ソースエンドポイントを選択します。
4. エンドポイント設定セクションで、次の操作を行います。

   - 一意で意味のあるエンドポイント識別子 (「mongodb-source」など) を入力します。
   - ソースエンジンとして MongoDB を選択します。
   - [Access to endpoint database] (エンドポイントデータベースへのアクセス) では、[Provide access information manually] (アクセス情報を手動で提供) を選択します。
   - サーバー名に、*MongoDB ##### DNS #/IP ####*を入力します。
   - ポートには、27017 (デフォルトの MongoDB ポート) と入力します。
   - 認証モードでは、アプリケーションに適したモード (パスワード/SSL) を選択します (デフォルトは Secrets Manager)。
   - 認証モードがパスワードの場合は、以下を指定します。
     - ユーザー名とパスワード: MongoDB 認証情報を入力します。
     - データベース名: ソースデータベース名。
     - 認証メカニズム: SCRAM-SHA-1 (デフォルト) または適切なメカニズム
5. メタデータモードでは、ドキュメントのデフォルト設定のままにします。
6. 追加の接続属性：

   - authSource=admin (認証データベースが異なる場合)
   - replicaSet=<your-replica-set-name> (CDC に必須)

Amazon DocumentDB ソースの場合

1. DMS コンソールのナビゲーションペインで、移行またはレプリケートを選択し、エンドポイントを選択します。
2. [エンドポイントの作成] を選択します。
3. エンドポイントの作成ページで、ソースエンドポイントを選択します。
4. エンドポイント設定セクションで、次の操作を行います。
   - 一意で意味のあるエンドポイント識別子 (「docdb-source」など) を入力します。
   - Amazon DocumentDB をソースエンジンとして選択します。
   - [Access to endpoint database] (エンドポイントデータベースへのアクセス) では、[Provide access information manually] (アクセス情報を手動で提供) を選択します。
   - サーバー名に、*`### Amazon DocumentDB ############`*を入力します。
   - ポートには、27017 (デフォルトの Amazon DocumentDB ポート) と入力します。
   - SSL モードの場合は、verify-full (Amazon DocumentDB に推奨) を選択します。
   - CA 証明書で、Amazon RDS ルート CA 証明書を選択します。
   - 認証モードでは、アプリケーションに適したモード (パスワード/SSL) を選択します (デフォルトはシークレットマネージャー) 。
   - 認証モードがパスワードの場合は、以下を指定します。
     - ユーザー名とパスワード: Amazon DocumentDB 認証情報を入力します。
     - データベース名: ソースデータベース名。
     - 認証メカニズム: SCRAM-SHA-1 (デフォルト) または適切なメカニズム
5. メタデータモードでは、ドキュメントのデフォルト設定のままにします。

ターゲットエンドポイントを作成する (Amazon DocumentDB)

1. DMS コンソールのナビゲーションペインで、移行またはレプリケートを選択し、エンドポイントを選択します。
2. [エンドポイントの作成] を選択します。
3. エンドポイントの作成ページで、ターゲットエンドポイントを選択します。
4. エンドポイント設定セクションで、次の操作を行います。
   - 一意で意味のあるエンドポイント識別子 (「docdb-target」など) を入力します。
   - ターゲットエンジンとして Amazon DocumentDB を選択します。

- エンドポイントデータベースへのアクセス で、データベースへのアクセスを認証するために使用する方法を選択します。
  - AWS Secrets Manager を選択した場合は、Secret フィールドに Amazon DocumentDB 認証情報を保存するシークレットを選択します。
  - アクセス情報を手動で提供するを選択した場合：
    - サーバー名に、*##### Amazon DocumentDB ############*を入力します。
    - ポートには、27017 (デフォルトの Amazon DocumentDB ポート) と入力します。
    - SSL モードの場合は、verify-full (Amazon DocumentDB に推奨) を選択します。
    - CA 証明書の場合は、SSL 検証用の CA 証明書バンドルをダウンロードして指定します。
    - 認証モードでは、アプリケーションに適したモード (パスワード/SSL) を選択します (デフォルトはシークレットマネージャー) 。
    - 認証モードがパスワードの場合は、以下を指定します。
      - ユーザー名とパスワード: Amazon DocumentDB 認証情報を入力します。
      - データベース名: ソースデータベース名。
      - 認証メカニズム: SCRAM-SHA-1 (デフォルト) または適切なメカニズム
5. メタデータモードでは、ドキュメントのデフォルト設定のままにします。

レプリケーションタスクを作成する

1. DMS コンソールのナビゲーションペインで、移行またはレプリケートを選択し、タスクを選択します。
2. [Create task] (タスクの作成) を選択します。
3. 「タスクの作成」ページの「タスク設定」セクションで、次の操作を行います。
   - 一意で意味のあるタスク識別子 (mongodb-docdb-replication」など) を入力します。
   - ソースデータベースエンドポイントドロップダウンメニューで、以前に作成したソースエンドポイントを選択します。
   - ターゲットデータベースエンドポイントドロップダウンメニューで、以前に作成したターゲットエンドポイントを選択します。
   - タスクタイプで、移行とレプリケートを選択します。
4. 設定セクションで、次の操作を行います。
   - ターゲットテーブル準備モードでは、デフォルト設定のままにします。

- 全ロードの完了後にタスクを停止するには、デフォルト設定のままにします。
- LOB 列の設定では、制限付き LOB モードの設定をそのままにします。
- データ検証の場合は、デフォルト設定の「オフ」のままにします。
- タスクログの場合は、CloudWatch ログをオンにするチェックボックスをオンにします。
- バッチ最適化適用の場合、デフォルト設定の はオフ (オフ) のままにします。

5. タスク設定セクションの上部に戻り、編集モードで JSON エディタを選択し、次の属性を設定します。

```
{
  "TargetMetadata": {
    "ParallelApplyThreads": 5
  },
  "FullLoadSettings": {
    "MaxFullLoadSubTasks": 16
  }
}
```

6. テーブルマッピングセクションで、新しい選択ルールを追加します。

- スキーマ名には、移行するソースデータベースを追加します。% を使用して複数のデータベースを指定します。
- スキーマテーブル名には、移行するソースコレクションを追加します。% を使用して複数のコレクションを指定します。
- アクションの場合、デフォルト設定の「含める」のままにします。

7. 大規模なコレクション (100GB 以上) の場合は、テーブル設定ルールを追加します。

- スキーマ名には、移行するソースデータベースを追加します。% を使用して複数のデータベースを指定します。
- スキーマテーブル名には、移行するソースコレクションを追加します。% を使用して複数のコレクションを指定します。
- パーティションの数には、16 と入力します (より小さくする必要がありますMaxFullLoadSubTask) 。

8. 移行前評価セクションで、オフになっていることを確認します。

オフライン移行

このセクションでは、 mongodumpおよび のネイティブ MongoDB ツールを使用して、セルフマネージド MongoDB インスタンスから Amazon DocumentDB へのオフライン移行を実行するプロセスの概要を説明しますmongorestore。

前提条件

ソース MongoDB の要件

- 適切なアクセス許可を持つソース MongoDB インスタンスへのアクセス。
- 必要に応じて をインストールmongodumpします (MongoDB のインストール中にインストールされます)。
- ダンプファイルに十分なディスク容量があることを確認します。

Amazon DocumentDB のターゲット要件

- Amazon DocumentDB クラスターがプロビジョニングされていることを確認します。
- 移行を容易にするために、Amazon DocumentDB と同じ VPC に EC2 インスタンスがあることを確認します。
- ネットワーク接続は、ソース環境と Amazon DocumentDB の間で利用できる必要があります。
- mongorestore は移行 EC2 インスタンスにインストールする必要があります。
- Amazon DocumentDB にアクセスするには、適切な IAM アクセス許可を設定する必要があります。

一般的な要件

- AWS CLI を設定する必要があります (中間ストレージに AWS サービスを使用している場合)
- データ転送には十分な帯域幅が必要です。
- ダウンタイムウィンドウは承認する必要があります (ライブ移行を行う場合は、他のアプローチを検討してください)

Amazon DocumentDB クラスターを準備する

で Amazon DocumentDB クラスターを作成します AWS。

- ワークロードに基づいて適切なインスタンスサイズ。
- VPC、サブネット、セキュリティグループを設定します。
- パラメータグループを介して必要なパラメータを有効にします。

データダンプを実行する (mongodump)

ダンプファイルを作成するには、次のいずれかのオプションを選択します。

- オプション 1: 基本

```
mongodump --
uri="mongodb://<source_user>:<source_password>@<source_host>:<source_port>/
<database>" --
out=/path/to/dump
```

- オプション 2: 制御とパフォーマンスの向上

```
mongodump \
--uri="mongodb://<source_user>:<source_password>@<sourcehost>:<source_port>" \
--out=/path/to/dump \
--gzip \# Compress output
--numParallelCollections=4 \# Parallel collections dump
--ssl \# If using SSL
--authenticationDatabase=admin \ # If auth is required
--readPreference=secondaryPreferred # If replica set
```

- オプション 3: 大規模なデータベース

```
mongodump \
--host=<source_host> \
--port=<source_port> \
--username=<source_user> \
--password=<source_password> \
--db=<specific_db> \# Only dump specific DB
--collection=<specific_collection> \ # Only dump specific collection
--query='{ "date": { "$gt": "2020-01-01" } }' \ # Filter documents
--archive=/path/to/archive.gz \# Single archive output
--gzip \
--ssl
```

ダンプファイルを復元環境に転送する

ダンプサイズに基づいて適切な方法を選択します。

- 小規模 — 移行マシン (前に作成した EC2 インスタンス) に直接コピーします。

```
scp -r /path/to/dump user@migration-machine:/path/to/restore
```

- Medium — Amazon S3 を中間ストレージとして使用します。

```
aws s3 cp --recursive /path/to/dump s3://your-bucket/mongodb-dump/
```

- Large — 非常に大規模なデータベースの場合は、 AWS DataSync または物理的な転送を検討してください。

Amazon DocumentDB (mongorestore) へのデータの復元

復元プロセスを開始する前に、Amazon DocumentDB でインデックスを作成します。Amazon DocumentDB インデックスツールを使用して、インデックスをエクスポートおよびインポートできます。

データを復元するには、次のいずれかのオプションを選択します。

- オプション 1: 基本的な復元

```
mongorestore --uri="mongodb://<docdb_user>:<docdb_password>@<docdb_endpoint>:27017"
/path/to/dump
```

- オプション 2: 制御とパフォーマンスの向上

```
mongorestore \
--uri="mongodb://<docdb_user>:<docdb_password>@<docdb_endpoint>:27017" \
--ssl \
--sslCAFile=/path/to/rds-combined-ca-bundle.pem \ # DocumentDB CA cert
--gzip \# If dumped with gzip
--numParallelCollections=4 \# Parallel restoration
--numInsertionWorkersPerCollection=4 \# Parallel documents insertion
--noIndexRestore \# skip indexes as they are pre-created
/path/to/dump
```

- オプション 3: 大規模なデータベースまたは特定のコントロール

```
mongorestore \
--host=<docdb_endpoint> \
--port=27017 \
--username=<docdb_user> \
--password=<docdb_password> \
--ssl \
--sslCAFile=/path/to/rds-combined-ca-bundle.pem \
--archive=/path/to/archive.gz \# If using archive format
--gzip \
--nsInclude="db1.*" \# Only restore specific namespaces
--nsExclude="db1.sensitive_data" \ # Exclude specific collections if needed
--noIndexRestore \# skip indexes as they are pre-created
--writeConcern="{w: 'majority'}" # Ensure write durability
```

モニタリング

このセクションでは、以下からの継続的な移行の進行状況、パフォーマンス、状態を追跡するための詳細なモニタリングプロセスについて説明します。

MongoDB から Amazon DocumentDB へ

or

Amazon DocumentDB から Amazon DocumentDB へ

モニタリングステップは、移行方法 (AWS DMS、mongodump/mongorestore、またはその他のツール) に関係なく適用されます。

AWS DMS 移行モニタリング (該当する場合)

次の主要な CloudWatch メトリクスをモニタリングします。

全ロードフェーズのメトリクス

- FullLoadThroughputBandwidthTarget — 全ロード中のネットワーク帯域幅 (KB/秒)
- FullLoadThroughputRowsTarget — 1 秒あたりにロードされた行/ドキュメントの数
- FullLoadThroughputTablesTarget — 1 分あたりに完了したテーブル/コレクションの数
- FullLoadProgressPercent — 全ロード完了の割合
- TablesLoaded — 正常にロードされたテーブル/コレクションの数

- TablesLoading — 現在ロードされているテーブル/コレクションの数
- TablesQueued — ロードを待機しているテーブル/コレクションの数
- TablesErrored — ロードに失敗したテーブル/コレクションの数

CDC フェーズメトリクス

- CDCLatencyTarget — ソース変更とターゲットアプリケーション間の遅延時間 (秒)
- CDCLatencySource — ソースの変更とそれを読み取る DMS の間の遅延時間 (秒)
- CDCThroughputRowsTarget — 継続的なレプリケーション中に適用される 1 秒あたりの行数
- CDCThroughputBandwidthTarget — CDC 中のネットワーク帯域幅 (KB/秒)
- CDCIncomingChanges — ソースから受信した変更イベントの数
- CDCChangesMemoryTarget — 変更をターゲット側に保存するために使用するメモリ (MB)

リソースメトリクス

- CPUUtilization — レプリケーションインスタンスの CPU 使用率
- FreeableMemory — レプリケーションインスタンスで使用可能なメモリ
- FreeStorageSpace — レプリケーションインスタンスで使用可能なストレージ
- NetworkTransmitThroughput — レプリケーションインスタンスのネットワークスループット
- NetworkReceiveThroughput — レプリケーションインスタンスのネットワークスループット

エラーメトリクス

- ErrorsCount — 移行中のエラーの合計数
- TableErrorsCount — テーブル固有のエラーの数
- RecordsErrorsCount — レコード固有のエラーの数

移行パフォーマンスが低下した場合に通知を受け取るCPUUtilizationには、
CDCLatencyTargetや などの重要なメトリクスの CloudWatch アラームを作成します。

DMS ログ (CloudWatch ログ)

1. Amazon CloudWatch Logs コンソールに移動します。

2. ロググループで を検索して選択します。「dms-tasks –」のようになります。
3. エラー情報を含む可能性のあるログストリームを探します。
   - 名前に「エラー」があるストリーム
   - タスク IDs またはエンドポイント名を持つストリーム
   - 移行中の最新のログストリーム
4. これらのストリーム内で、次のようなキーワードを検索します。
   - 「エラー」
   - 「例外」
   - 「失敗」
   - 「警告」

DMS タスクのステータス (使用 AWS CLI)

```
aws dms describe-replication-tasks --filters Name=replication-task id,Values=<task_id>
 --query
"ReplicationTasks[0].Status"
```

予想されるステータスフロー：

作成中 → 準備完了 → 実行中 → 停止中 → 停止 (または失敗)

を使用したモニタリング **docdb-dashboarder**

このdocdb-dashboarderツールは、重要なパフォーマンスメトリクスを含む CloudWatch ダッシュボードを自動的に生成することで、Amazon DocumentDB クラスターの包括的なモニタリングを提供します。これらのダッシュボードには、重要なクラスターレベルのメトリクス (レプリカラグ、オペレーションカウンター) 、インスタンスレベルのメトリクス (CPU、メモリ、接続) 、ストレージメトリクス (ボリューム使用量、バックアップストレージ) が表示されます。移行シナリオの場合、このツールは、CDC レプリケーションの遅延やオペレーションレートなどのメトリクスを使用して移行の進行状況を追跡する特殊なダッシュボードを提供します。ダッシュボードでは、複数のクラスターを同時にモニタリングし、NVMe-backed インスタンスのサポートを含めることができます。これらのメトリクスを視覚化することで、チームはパフォーマンスのボトルネックをプロアクティブに特定し、リソース割り当てを最適化し、Amazon DocumentDB デプロイのスムーズな運用を確保できます。このツールを使用すると、ダッシュボードを手動で作成する必要がなくなり、すべ

ての環境で一貫したモニタリングが可能になります。セットアップ手順と高度な設定オプションについては、Amazon DocumentDB Dashboarder Tool GitHub リポジトリを参照してください。

検証

トピック

- 検証チェックリスト
- スキーマとインデックスの検証
- データサンプリングとフィールドレベルの検証
- DataDiffer ツールを使用した検証

このセクションでは、以下から移行した後のデータ整合性、整合性、アプリケーションの互換性を確保するための詳細な検証プロセスについて説明します。

MongoDB から Amazon DocumentDB へ

or

Amazon DocumentDB から Amazon DocumentDB へ

検証ステップは、移行方法 (AWS DMS、mongodump/mongorestore、またはその他のツール) に関係なく適用されます。

検証チェックリスト

各コレクションのドキュメント数がソースとターゲットの間で一致することを確認します。

MongoDB ソース

```
mongo --host <source_host> --port <port> --username <user> -- password <password> --
eval
"db.<collection>.count()"
```

Amazon DocumentDB ターゲット

```
mongo --host <target_host> --port <port> --username <user> -- password <password> --
eval
"db.<collection>.count()"
```

スキーマとインデックスの検証

以下を確認してください。

- すべてのコレクションがターゲットに存在します。
- インデックスは正しくレプリケートされます。
- スキーマ定義 (適用されている場合) は同じです。

コレクションを確認する (ソースとターゲット)

```
mongo --host <source_host> --eval "show collections"
mongo --host <target_host> --ssl --eval "show collections"
```

チェックインデックス (ソースとターゲット)

```
mongo --host <source_host> --eval" db.<collection>.getIndexes()"
mongo --host <target_host> --ssl -eval" db.<collection>.getIndexes()"
```

コレクションのリストを比較して、欠落しているコレクションや追加のコレクションがないことを確認します。

インデックス名、キー定義、一意の制約、TTL インデックス (存在する場合) を確認して、インデックスを検証します。

スキーマ検証ルールを確認する (MongoDB でスキーマ検証を使用する場合)

```
mongo --host <source_host> --eval" db.getCollectionInfos({name: '<collection>'})
[0].options.validator"
   mongo --host <target_host> --ssl -eval" db.getCollectionInfos({name:
 '<collection>'})[0].options.validator"
```

データサンプリングとフィールドレベルの検証

ドキュメントをランダムにサンプリングし、ソースとターゲットのフィールドを比較できます。

手動サンプリング

5 つのランダムドキュメント (ソース) を取得します。

```
mongo --host <source_host> --eval "db.<collection>.aggregate([{ \$sample: { size:
 5 } }])"
```

同じドキュメント IDs (ターゲット) を取得します。

```
mongo --host <target_host> --ssl -eval "db.<collection>.find({ _id: { \$in:
 [<list_of_ids>] } })"
```

自動サンプリング

```
import pymongo
# Connect to source and target
source_client = pymongo.MongoClient("<source_uri>")
target_client = pymongo.MongoClient("<target_uri>", ssl=True)
source_db = source_client["<db_name>"]
target_db = target_client["<db_name>"]
# Compare 100 random documents
for doc in source_db.<collection>.aggregate([{ "$sample":
{ "size": 100 } }]):
target_doc = target_db.<collection>.find_one({ "_id":
doc["_id"] })
if target_doc != doc:
print(f"# Mismatch in _id: {doc['_id']}")
else:
print(f"# Match: {doc['_id']}")
```

DataDiffer ツールを使用した検証

DataDiffer ツールは、ソースデータベースとターゲットデータベース間でデータを比較する信頼性の高い方法を提供します。

前提条件

DataDiffer ツールをインストールする前に、次の前提条件を満たす必要があります。

- Python 3.7 以上
- PyMongo ライブラリ
- ソース MongoDB クラスターとターゲット Amazon DocumentDB クラスターの両方へのネットワーク接続

セットアップとインストール

リポジトリのクローンを作成し、DataDiffer ディレクトリに移動します。

```
git clone https://github.com/awslabs/amazon-documentdb-tools.git
cd amazon-documentdb-tools/migration/data-differ
```

必要な依存関係をインストールする

```
pip install -r requirements.txt
```

データ検証の実行

接続の詳細を含む設定ファイル (config.json など) を作成する

```
{
"source": {
"uri": "mongodb://username:password@source-mongodb-
host:27017/?replicaSet=rs0",
"db": "your_database",
"collection": "your_collection"
},
"target": {
"uri": "mongodb://username:password@target-docdb-
cluster.region.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0",
"db": "your_database",
"collection": "your_collection"
},
"options": {
"batch_size": 1000,
"threads": 4,
"sample_size": 0,
"verbose": true
}
}
```

DataDiffer ツールを実行する

```
python differ.py --config config.json
```

大規模なコレクションの場合、サンプリングを使用してデータのサブセットを検証する

```
python differ.py --config config.json --sample-size 10000
```

複数のコレクションを検証するには、個別の設定ファイルを作成するか、バッチモードを使用します。

```
python differ.py --batch-config batch_config.json
```

結果の解釈

ツールは以下を出力します。

- ソースとターゲットの合計ドキュメント数
- 一致するドキュメントの数
- 欠落しているドキュメントの数
- 違いがあるドキュメントの数
- 差異の詳細レポート (存在する場合)

ベストプラクティス

DataDiffer ツールを使用する際のベストプラクティスは次のとおりです。

- 段階的に実行する — まずドキュメント数を検証し、次にキードキュメントのサンプルを作成し、最後に必要に応じて完全な比較を実行します。
- スキーマの違いを確認する — Amazon DocumentDB には、MongoDB と比較していくつかの制限があります。ツールでは、互換性のないデータ型または構造が強調表示されます。
- クワイエットピリオド中の検証 — 書き込みオペレーションが最小限であるときに検証を実行して整合性を確保します。
- リソース使用状況のモニタリング — 比較プロセスはリソースを大量に消費する可能性があります。それに応じてバッチサイズとスレッド数を調整します。
- インデックスの検証 — データの検証後、ターゲット Amazon DocumentDB クラスターで必要なインデックスがすべて作成されていることを確認します。
- ドキュメントの検証結果 — 移行ドキュメントの一部として、各コレクションの検証結果を記録します。

Amazon DocumentDB のアップグレード

入門段落をここに挿入します。

トピック

- Amazon DocumentDB のインプレースメジャーバージョンアップグレード
- を使用した Amazon DocumentDB クラスターのアップグレード AWS Database Migration Service

Amazon DocumentDB のインプレースメジャーバージョンアップグレード

Amazon DocumentDB では、広範なテストの後にのみデータベースエンジンの新しいバージョンを一般利用できます。Amazon DocumentDB クラスターを新しいバージョンにアップグレードする方法とタイミングは、選択することができます。

現在、Amazon DocumentDB は Amazon DocumentDB 3.6、4.0、5.0 の 3 つのメジャーバージョンをサポートしています。クラスターのエンドポイント、ストレージ、タグはそのままに、データベースのインプレースメジャーバージョンアップグレード (MVU) を実行することで、変更を加えることなく引き続きアプリケーションを使用できます。この機能は、Amazon DocumentDB 5.0 が利用可能なすべてのリージョンで無料で利用できます。

> **Important**
>
> Amazon DocumentDB クラスターは、インプレースメジャーバージョンアップグレード中は使用できず、その間、クラスターは複数回再起動されます。アップグレードの開始後は、クラスターへの接続、読み取り、または書き込みを控えてください。アップグレードのダウンタイムは、コレクションやインデックス、データベース、インスタンスの数によってクラスターごとに異なる場合があります。アップグレードは、メンテナンスウィンドウまたは使用率が低い時間帯に行うことをお勧めします。クラスターをアップグレードすると、クラスターを以前のバージョンにダウングレードすることはできませんが、アップグレード前のスナップショットを新しいクラスターに復元することはできます。

トピック

- MVU 前提条件と制限事項

- インプレースメジャーバージョンアップグレードのベストプラクティス
- インプレースメジャーバージョンアップグレードの実行
- Amazon DocumentDB 3.6/4.0 から 5.0 にアップグレードされたクラスターと新規作成される Amazon DocumentDB 5.0 クラスターの違い
- インプレースメジャーバージョンアップグレードのトラブルシューティング

MVU 前提条件と制限事項

以下は、インプレースメジャーバージョンアップグレードに対する前提条件と制限事項です。該当する場合は、アップグレードを実行する前に理解して対処しておく必要があります。

- インスタンスタイプ: Amazon DocumentDB 4.0/5.0 は r4.* インスタンスをサポートしていません。インプレースメジャーバージョンアップグレードを行うには、r4.* インスタンスを r5.* インスタンスに変更します。詳細については「Amazon DocumentDB インスタンスの変更」を参照してください。Amazon DocumentDB エンジンのバージョンごとのサポートされているインスタンスについては、「リージョン別のサポートされるインスタンスクラス」を参照してください。
- インスタンスの OS パッチ: インプレースメジャーバージョンアップグレードを行うには、最新のオペレーティングシステム (OS) パッチが必要です。インプレースアップグレードを行う前に、保留中の OS メンテナンスアクションをインスタンスに適用しておいてください。詳細については、「Amazon DocumentDB オペレーティングシステムの更新」を参照してください。

  > **Note**
  >
  > 保留中のクラスターレベルのエンジンパッチがある場合、状況によっては、インスタンスの OS パッチが表示されないことがあります。インスタンスの OS パッチの適用して、インプレースメジャーバージョンアップグレードを行う前に、クラスターレベルのエンジンパッチの適用が必要になる場合があります。「クラスターのエンジンバージョンへのパッチ更新の実行」を参照してください。

- インプレースメジャーバージョンアップグレードは、Amazon DocumentDB 5.0 が利用可能なすべてのリージョンで行うことができます。
- Amazon DocumentDB 4.0 がターゲットバージョンの場合、インプレースメジャーバージョンアップグレードはできません。
- Amazon DocumentDB 4.0 以降、ユーザー名に含まれる「.」はサポートされません。Amazon DocumentDB 3.6 から 5.0 へのアップグレードを計画しており、かつ「.」を含むユーザー名をお使いの場合は、インプレース MVU に進む前に「.」なしでユーザー名を再作成してください。

- 現在、Amazon DocumentDB グローバルクラスターとエラスティッククラスターでは、インプレースバージョンアップグレードはできません。

> **Note**
>
> グローバルクラスターをアップグレードするには、セカンダリクラスターをグローバルクラスターから削除し、プライマリクラスターをリージョンクラスターに変換し、リージョン (プライマリ) クラスターでインプレースメジャーバージョンアップグレードを実行します。次に、以前と同じエンドポイントを保持するには、同じ名前を使用してセカンダリクラスターを追加してグローバルクラスターを再作成します。アップグレードしたプライマリクラスターが新しく追加したセカンダリクラスタにデータを複製している間は、IO 料金が発生することに注意してください。削除する前にグローバルクラスタからセカンダリクラスタを削除する方法の詳細な手順については、「Amazon DocumentDB グローバルクラスターからのクラスターの削除」を参照してください。

- 大量のインデックス (3,000 個以上) があり、バーストパフォーマンスインスタンス (t3.medium や t4g.mediumt など) で運用している場合、インプレースメジャーバージョンアップグレードを実行するには、プライマリインスタンスをよりサイズの大きいインスタンス (r5.xlarge など) にスケールアップする必要があります。インスタンスサイズは、インプレースメジャーバージョンアップグレードの完了後にスケールダウンできます。インプレースメジャーバージョンアップグレードにおいて db.t3 および db.t4g のインスタンスタイプでサポートするインデックスの最大数については、以下の表を参照してください。

インスタンス	インプレース MVU でサポートされる最大インデックス数
db.t4g.medium	3K
db.t3.medium	10K

インプレースメジャーバージョンアップグレードのベストプラクティス

トピック

- クローンクラスターを使用したインプレースメジャーバージョンアップグレードのテスト
- インプレースメジャーバージョンアップグレードの前
- インプレースメジャーバージョンアップグレード中

- インプレースメジャーバージョンアップグレード後

クローンクラスターを使用したインプレースメジャーバージョンアップグレードのテスト

1. インプレースメジャーバージョンアップグレードをテストでは、高速クローン作成機能を使用してターゲットクラスターのクローンを作成することをお勧めします。クラスター上のデータを変更しない限り、クローンボリュームでインプレースメジャーバージョンアップグレードをテストしても、ストレージコストは発生しません。クローン作成の詳細については、「Amazon DocumentDB クラスターのボリュームのクローン作成」を参照してください。
2. インプレースメジャーバージョンアップグレードの実行にかかる時間のより具体的な数字を得るには、クローンクラスターのインスタンス数をターゲットクラスターに一致させてください。
3. 新しくアップグレードした Amazon DocumentDB 5.0 クラスターの機能の違いを十分にテストして、すべてが期待どおりに動作することを確認することをお勧めします。

インプレースメジャーバージョンアップグレードの前

1. バージョン互換のパラメータグループを用意します。

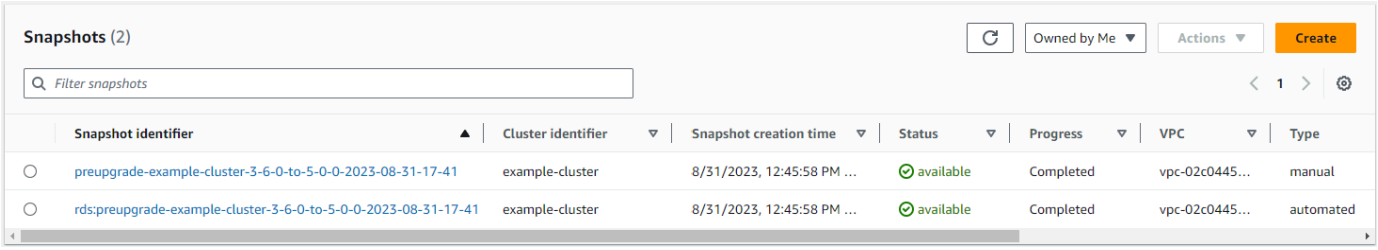
   新しいエンジンバージョンには、Amazon DocumentDB のデフォルトクラスターパラメータグループを使用することも、独自のカスタムクラスターパラメータグループを作成することもできます。

   アップグレードリクエストの一部として Amazon DocumentDB クラスターの新しいパラメータグループを関連付けると、インプレースメジャーバージョンアップグレードによってクラスターが自動的に再起動され、新しいパラメータグループが適用されます。
2. 「前提条件と制限事項」セクションで説明しているように、インプレースメジャーバージョンアップグレードの前提条件を満たしていることを確認します。
3. スナップショットを手動で作成します。

   アップグレードプロセスでは、アップグレード中に DB クラスターのスナップショットが作成されます。アップグレードプロセスの前に、独自のスナップショットを手動で作成しておくことを強くお勧めします。「手動クラスタースナップショットの作成」を参照してください。

> **Note**
>
> アップグレードプロセスで自動作成されたスナップショットは、インプレースメジャーバージョンアップグレードが完了しても自動的には削除されません。このスナップショットは、保持期間内である限り料金は発生しません。クラスターのアップグレードが成功したことを確認したら、このスナップショットを削除できます。

スナップショットの名前は `preupgrade-<name>-<version>-<timestamp>` です。



4. クラスターのインプレースメジャーバージョンアップグレードをすでにスケジュールしているかどうかを確認してください。

   クラスターを変更して次のメンテナンスウィンドウに適用することを選択した場合、インプレースメジャーバージョンアップグレードスケジュールはコンソールには表示されませんが、CLI で確認できます。`describe-db-clusters` コマンドを実行して、インプレースメジャーバージョンアップグレードが既にスケジュールされているかどうかを確認できます。

   ```
   aws docdb describe-db-cluster \
     --region us-east-1 \
     --db-cluster-identifier mydocdbcluster
   ```

   上記の例では、各*###############*をクラスターの情報に置き換えます。

   このコマンドは、以下の出力を返します。

   ```
   "PendingModifiedValues": {
       "EngineVersion": "5.0.0"
   },
   ```

5. インプレースメジャーバージョンアップグレード後は、下位環境で実行計画や機能に違いを付け、ボリュームクローンを使用して複数回クラスターのテストを実行します。インプレースメジャーバージョンアップグレードの実行時間をより正確に予測できるよう、同じ数とサ

イズのインスタンスでクローンを作成することをお勧めします。詳細については、「Amazon DocumentDB クラスターのボリュームのクローン作成」を参照してください。

6. 前のステップが成功したら、本番クラスターでのインプレースメジャーバージョンアップグレードに進みます。

インプレースメジャーバージョンアップグレード中

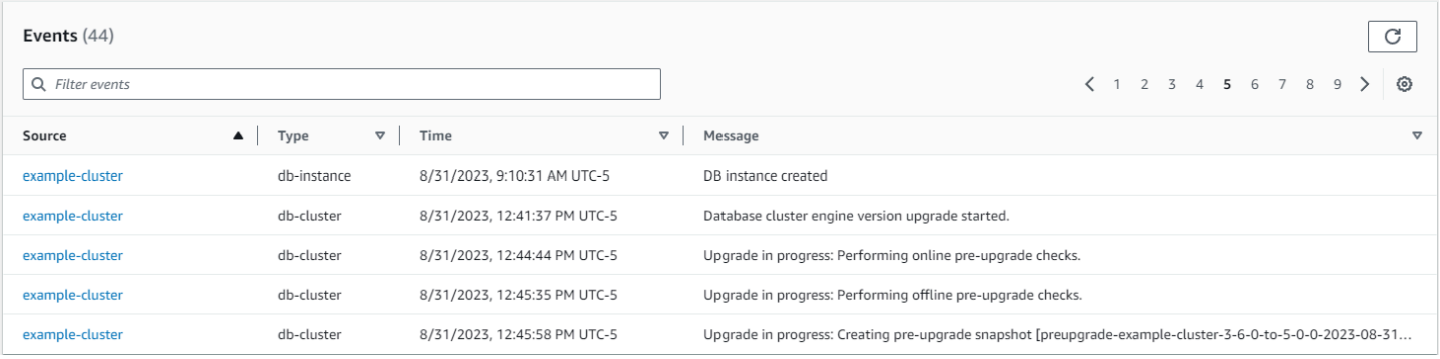
クラスターメンテナンスイベントにサブスクリプションすることで、インプレースメジャーバージョンアップグレードの進捗状況をモニタリングできます。アップグレードが完了すると、「データベースクラスターのメジャーバージョンがアップグレードされました」というイベントが送信されます。アップグレード中に発生するこのイベントやその他のイベントは、Amazon DocumentDB コンソールのクラスター詳細ページの「イベントとタグ」セクションに表示されます。その後、クラスターのステータスが「アップグレード中」から「利用可能」に変わります。

CLI から、`aws docdb create-event-subscription` を実行してイベントを作成したり、`aws docdb describe-events` を実行して進行状況をモニタリングしたりできます。また、E メールやプッシュメッセージ、その他の方法での上記イベントのイベント通知先として Amazon SNS を設定することもできます。詳細については、「Amazon DocumentDB イベントにサブスクライブする」を参照してください。

インプレースメジャーバージョンアップグレードでは、アップグレード中に以下のイベントが生成されます。

- アップグレード中: アップグレード前のスナップショット [preupgrade-<クラスター名>-<タイムスタンプ>] を作成しています
- アップグレード中: ボリュームのクローンを作成しています。
- アップグレード中: ライターをアップグレードしています。
- アップグレード中: リーダーをアップグレードしています。
- データベースクラスターエンジンのメジャーバージョンがアップグレードされました。

イベントはコンソールの [イベント] ページにも表示されます。



では AWS CLI、 describe-events コマンドを実行して進行状況を追跡できます。

```
aws docdb describe-events
  --source-identifier mydocdbcluster
  --source-type db-cluster
```

上記の例では、各##############をクラスターの情報に置き換えます。

このコマンドは、以下の出力を返します。

```
{
    "Events": [
        {
            "SourceIdentifier": "mydocdbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster engine version upgrade started.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2023-07-11T23:20:32.444000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:xxxx:cluster:mycluster"
        }
    ]
}
```

インプレースメジャーバージョンアップグレード後

Amazon DocumentDB 3.6 の場合は、クラスターにタグを追加することで、新しく作成された Amazon DocumentDB 5.0 クラスターではなく Amazon DocumentDB 3.6 から Amazon DocumentDB 5.0 にアップグレードされたことを区別できます。アップグレードした Amazon DocumentDB 5.0 クラスターと新しい Amazon DocumentDB 5.0 クラスターの違いに関するセクションを参照してください。

アップグレード後の状態に復元する必要がある場合に備えて、インプレース MVU が終了した後に手動スナップショットを作成します。自動スナップショット処理は、インプレースメジャーバージョンアップグレードが完了するとすぐに再開されます。この手動スナップショットは、保持期間内である限り料金は発生しません。

Amazon DocumentDB 5.0 に関連する新機能 (クライアント側のフィールドレベル暗号化など) を使用するには、ドライバーのバージョンを MongoDB 5.0 API バージョンにアップグレードすることをお勧めします。詳細については、「Amazon DocumentDB 5.0 の新機能」の Amazon DocumentDB 5.0 の機能リストを参照してください。

> **Important**
>
> インプレースメジャーバージョンアップグレード (MVU) の実行直後、Amazon DocumentDB 5.0 クラスターがインデックスメタデータを自動再入力し、データベースエンジンがクエリ実行プランを最適化します。お使いの Amazon DocumentDB クラスターにおいて通常想定されるクエリパフォーマンスは、インデックスメタデータの再計算プロセス完了後に再開されます。多くの場合、このプロセスは数分で完了しますが、クラスターのインデックス数によっては最大 2 時間程度かかる場合があります。インプレース MVU の直後にライターインスタンスを再起動、フェイルオーバー、またはスケールアップ/スケールダウンすると、クラスターのインデックスメタデータ計算プロセスが中断される可能性があります。したがってこうした変更は、インプレース MVU の完了後、Amazon DocumentDB 5.0 クラスターで想定通りのクエリパフォーマンスが得られるかを確認した後で実施することをお勧めします。さらに、インプレース MVU が完了すると、使用可能な変更ストリームデータは過去 3 時間に制限されます。
>
> この一時的なパフォーマンスの低下がインプレース MVU から 2 時間以上続く場合は、 AWS サポートにお問い合わせください。

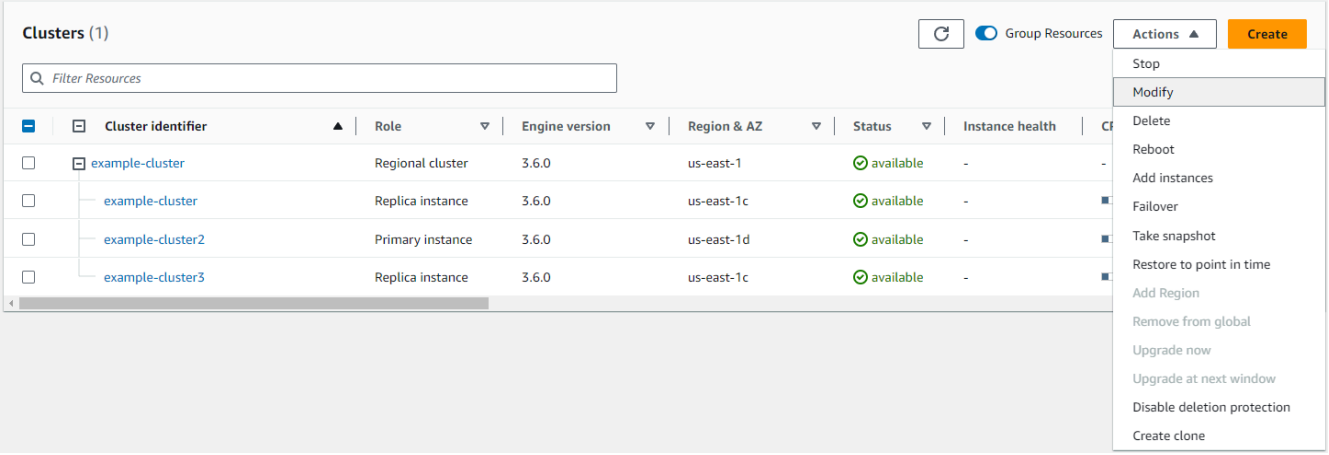
アップグレードfした Amazon DocumentDB 5.0 クラスターを十分にテストして、すべてが期待どおりに動作することを確認します。

インプレースメジャーバージョンアップグレードの実行

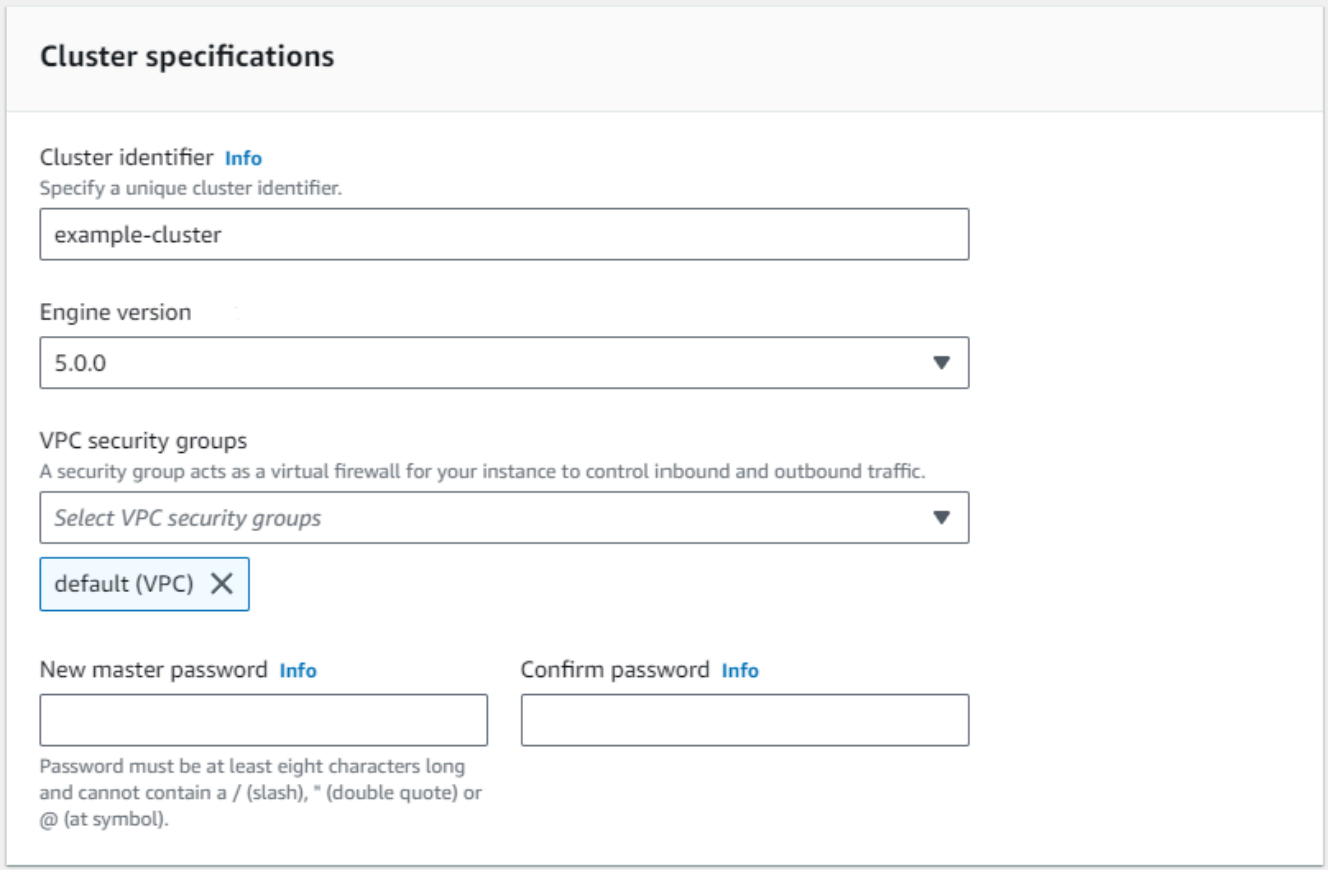
Using the AWS Management Console

AWS Management Consoleを使用してインプレースメジャーバージョンアップグレードを実行する

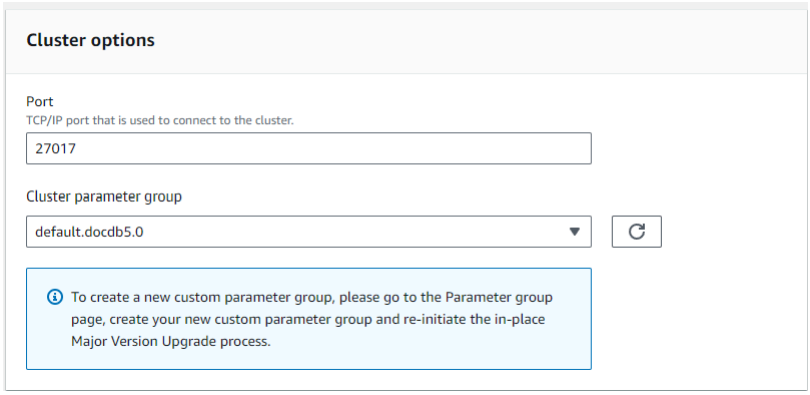
1. AWS Management Console にサインインし、Amazon DocumentDB コンソールを開きます。
2. [クラスタ] テーブルで、ソースクラスタを選択し、[アクション] をクリックして、 [変更]をクリックします。



3. クラスター仕様の変更セクションのクラスターの変更ダイアログで、エンジンバージョンドロップダウンメニューからターゲットデータベースバージョン (5.0.0) を選択します。



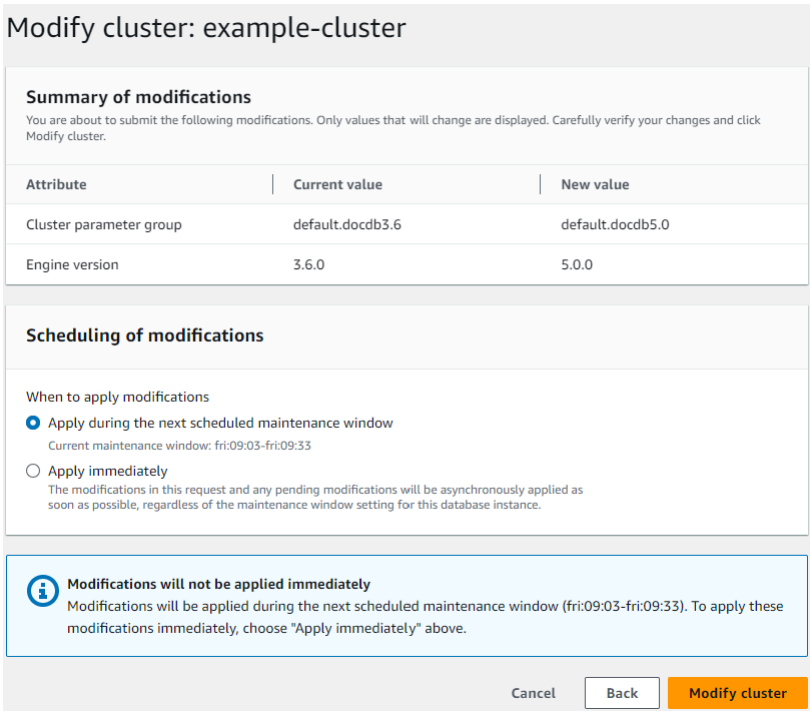
4. [クラスターオプション] セクションで、適切なクラスターパラメータグループ (default.docdb5.0) または独自に作成したパラメータグループを選択します。



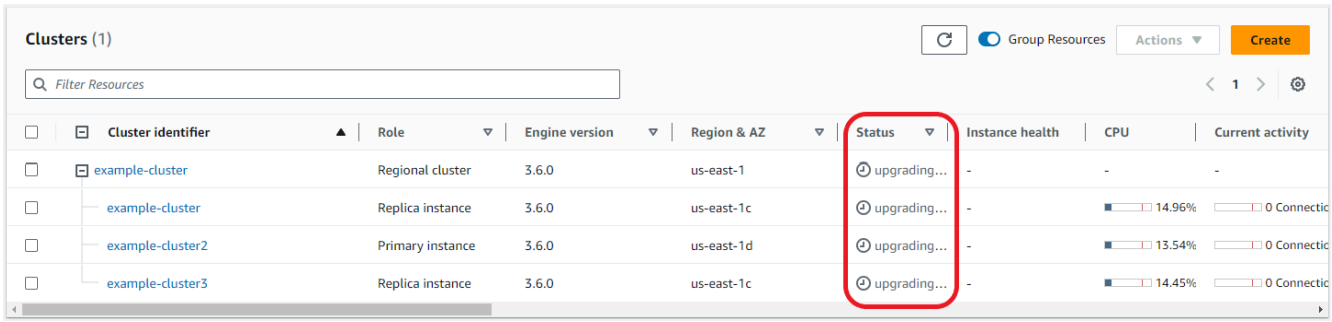
5. 完了したら、下にスクロールして [続行] を選択します。

6. [変更のスケジュール] セクションで、希望のスケジュールプラン (すぐに適用するか、次のメンテナンスウィンドウに適用) を選択します。

   その後、[Modify Cluster] を選択します。



7. クラスターの表に、アップグレード中のクラスターのステータスを書き留めてください。



Using the AWS CLI

目的のエンジンバージョンオプションとallow-major-version-upgradeフラグが設定された modify-db-cluster コマンドを使用します。

```
aws docdb modify-db-cluster \
  --db-cluster-identifier mydocdbcluster \
  --allow-major-version-upgrade \
  --engine-version 5.0.0 \
  --apply-immediately \
  --cluster-parameter-group mydocdbparametergroup \
  --region us-east-1
```

上記の例では、各##############をクラスターの情報に置き換えます。

Amazon DocumentDB 3.6/4.0 から 5.0 にアップグレードされたクラスターと新規作成される Amazon DocumentDB 5.0 クラスターの違い

- インプレースメジャーバージョンアップグレードしても、元のインデックスはアップグレード後のクラスターに引き継がれます。Amazon DocumentDB 5.0 では、インデックスメンテナンスとガベージコレクションプロセスの全体的な効率、特に低カーディナリティインデックスの効率が向上しました。一般的なベストプラクティスとして、MVU が正常に完了した後に reindex コマンドを使用してインデックスを再作成することをお勧めします。インデックスの再作成は必須ではなく、追加の I/O が含まれます。 詳細については、「」を参照してください[を使用したインデックスのメンテナンス reIndex](#)。
- 複数の数値データ型のサブドキュメント比較：
  - Amazon DocumentDB 3.6 から移行したクラスターの場合、クラスターは Amazon DocumentDB 3.6 のサブドキュメント比較動作を継承します。機能の違いはサブドキュメント内の数値型 (長整数型、倍精度型、Decimal128 型など) にあるだけです。例えば {a: {b:

`{NumberLong(1)}` は Amazon DocumentDB 3.6 で `{a: {b: 1}}` に等しくありませんが、Amazon DocumentDB 4.0 以降では等しいになります。

- このサブドキュメント比較動作は、Amazon DocumentDB 3.6 と、インプレースメジャーバージョンアップグレードを使用してバージョン 3.6 からアップグレードされた Amazon DocumentDB 5.0 クラスターでのみ存在します。これは、新しく作成された Amazon DocumentDB 5.0 クラスターには該当しません。

> **Note**
>
> Amazon DocumentDB 3.6/4.0と Amazon DocumentDB 5.0 の機能上の違いの一覧については、「Amazon DocumentDB の MongoDB との互換性」を参照してください。

インプレースメジャーバージョンアップグレードのトラブルシューティング

- 障害が発生した場合、インプレースメジャーバージョンアップグレードは、アップグレードのロールバックを試みて、アップグレード開始前のクラスターの最後の動作状態を引き継ごうとします。ロールバックが成功すると、「データベースクラスターはアップグレードできない状態です: DocumentDB クラスターはメジャーバージョンのアップグレードを正常に完了できない状態です。」というイベントが生成されます。この時点で、 AWS サポートチームに連絡してトラブルシューティングを行い、バージョンアップグレードを再試行する必要があります。ワークロードは以前と同様に使い続けることができます。アップグレードに予想以上に時間がかかるその他のまれなシナリオでは、 AWS サポートチームにお問い合わせください。
- しかし、インプレース MVU の正常な完了後、インデックスメタデータの更新プロセスの実行中に、短時間ではありますがアップグレードされたクラスターのパフォーマンスが一時的に低下し、CPU 使用率が高くなる可能性があります。2 時間以上パフォーマンスが低下し続ける場合は、 AWS サポートにお問い合わせください。

を使用した Amazon DocumentDB クラスターのアップグレード AWS Database Migration Service

> **Important**
>
> Amazon DocumentDB は MongoDB と同じサポートライフサイクルに従わず、MongoDB のライフサイクルの終了スケジュールは Amazon DocumentDB には適用されません。Amazon DocumentDB 3.6 のライフサイクル終了に関する現在の計画はありません。既存の MongoDB 3.6 ドライバ、アプリケーション、ツールは引き続き Amazon DocumentDB と連携します。

を使用すると、Amazon DocumentDB クラスターを最小限のダウンタイムで上位バージョンにアップグレードできます AWS DMS。 AWS DMS は、古い Amazon DocumentDB バージョン、リレーショナルデータベース、非リレーショナルデータベースからターゲット Amazon DocumentDB クラスターに簡単に移行できるフルマネージドサービスです。

トピック

- ステップ 1: 変更ストリームを有効にする
- ステップ 2: 変更ストリームログの保持期間を変更する
- ステップ 3: インデックスを移行する
- ステップ 4: AWS DMS レプリケーションインスタンスを作成する
- ステップ 5: AWS DMS ソースエンドポイントを作成する
- ステップ 6: AWS DMS ターゲットエンドポイントを作成する
- ステップ 7: 移行タスクの作成と実行
- ステップ 8: アプリケーションエンドポイントから Amazon DocumentDB クラスター への変更

ステップ 1: 変更ストリームを有効にする

最小限のダウンタイム移行を実行するには、クラスターの変更ストリームへのアクセス AWS DMS が必要です。Amazon DocumentDB 変更ストリーム は、クラスターのコレクションとデータベース内で発生する変更イベントの時系列シーケンスを提供します。変更ストリームから読み取ると AWS DMS 、 は変更データキャプチャ (CDC) を実行し、ターゲット Amazon DocumentDB クラスターに増分更新を適用できます。

特定のデータベースのすべてのコレクションの変更ストリームを有効にするには、[mongo] シェルを使用して Amazon DocumentDB クラスターを認証し、次のコマンドを実行します。

```
db.adminCommand({modifyChangeStreams: 1,
    database: "db_name",
    collection: "",
    enable: true});
```

ステップ 2: 変更ストリームログの保持期間を変更する

次に、変更ストリームに変更イベントを保持したい期間に基づいて、変更ストリームの保持期間を変更します。たとえば、 を使用した Amazon DocumentDB クラスターの移行 AWS DMS に 12 時間かかることが予想される場合は、変更ストリームの保持期間を 12 時間を超える値に設定する必要があります。Amazon DocumentDB クラスターのデフォルトの保持期間は 3 時間です。 AWS Management Console または を使用して、Amazon DocumentDB クラスターの変更ストリームログの保持期間を 1 時間から 7 日の間で変更できます AWS CLI。さらなる詳細については、「変更ストリームログの保持期間を変更するには」を参照してください。

ステップ 3: インデックスを移行する

Amazon DocumentDB クラスターと同じインデックスを、Amazon DocumentDB クラスター上に作成します。はデータの移行 AWS DMS を処理しますが、インデックスは移行されません。インデックスを移行するには、Amazon DocumentDB インデックスツールを使用して、Amazon DocumentDB クラスターからインデックスをエクスポートします。このツールを入手するには、Amazon DocumentDB ツール GitHub リポジトリのクローンを作成し、README.md 中の手順に従います。このツールは、Amazon DocumentDB クラスターと同じ Amazon VPC で実行されている Amazon EC2 インスタンスまたは AWS Cloud9 環境から実行できます。 Amazon DocumentDB

次の例では、各 ############## を独自の情報に置き換えます。

次のコードは、Amazon DocumentDB クラスターからインデックスをダンプします。

```
python migrationtools/documentdb_index_tool.py --dump-indexes
--uri mongodb://sample-user:user-password@sample-source-cluster.node.us-
east 1.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false'
--dir ~/index.js/
```

```
2020-02-11 21:51:23,245: Successfully authenticated to database: admin2020-02-11
 21:46:50,432: Successfully connected to instance docdb-40-xx.cluster-xxxxxxxx.us-
east-1.docdb.amazonaws.com:27017
2020-02-11 21:46:50,432: Retrieving indexes from server...2020-02-11 21:46:50,440:
 Completed writing index metadata to local folder: /home/ec2-user/index.js/
```

インデックスが正常にエクスポートされたら、ターゲットの Amazon DocumentDB クラスター中にそれらのインデックスを復元します。前のステップでエクスポートしたインデックスを復元するには、Amazon DocumentDB インデックスツールを使用します。次のコマンドは、指定したディレクトリから Amazon DocumentDB クラスター中のインデックスを復元します。

```
python migrationtools/documentdb_index_tool.py --restore-indexes
--uri mongodb://sample-user:user-password@sample-destination-
cluster.node.us-east 1.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false'
--dir ~/index.js/

2020-02-11 21:51:23,245: Successfully authenticated to database: admin2020-02-11
 21:51:23,245: Successfully connected to instance docdb-50-xx.cluster-xxxxxxxx.us-
east-1.docdb.amazonaws.com:27017
2020-02-11 21:51:23,264: testdb.coll: added index: _id
```

インデックスを正しく復元したことを確認するには、Mongo シェルを使用して Amazon DocumentDB クラスターに接続し、特定のコレクションのインデックスをリストします。以下のコードを参照してください。

```
mongo --ssl
--host docdb-xx-xx.cluster-xxxxxxxx.us-east-1.docdb.amazonaws.com:27017
--sslCAFile rds-ca-2019-root.pem --username documentdb --password documentdb

db.coll.getIndexes()
```

ステップ 4: AWS DMS レプリケーションインスタンスを作成する

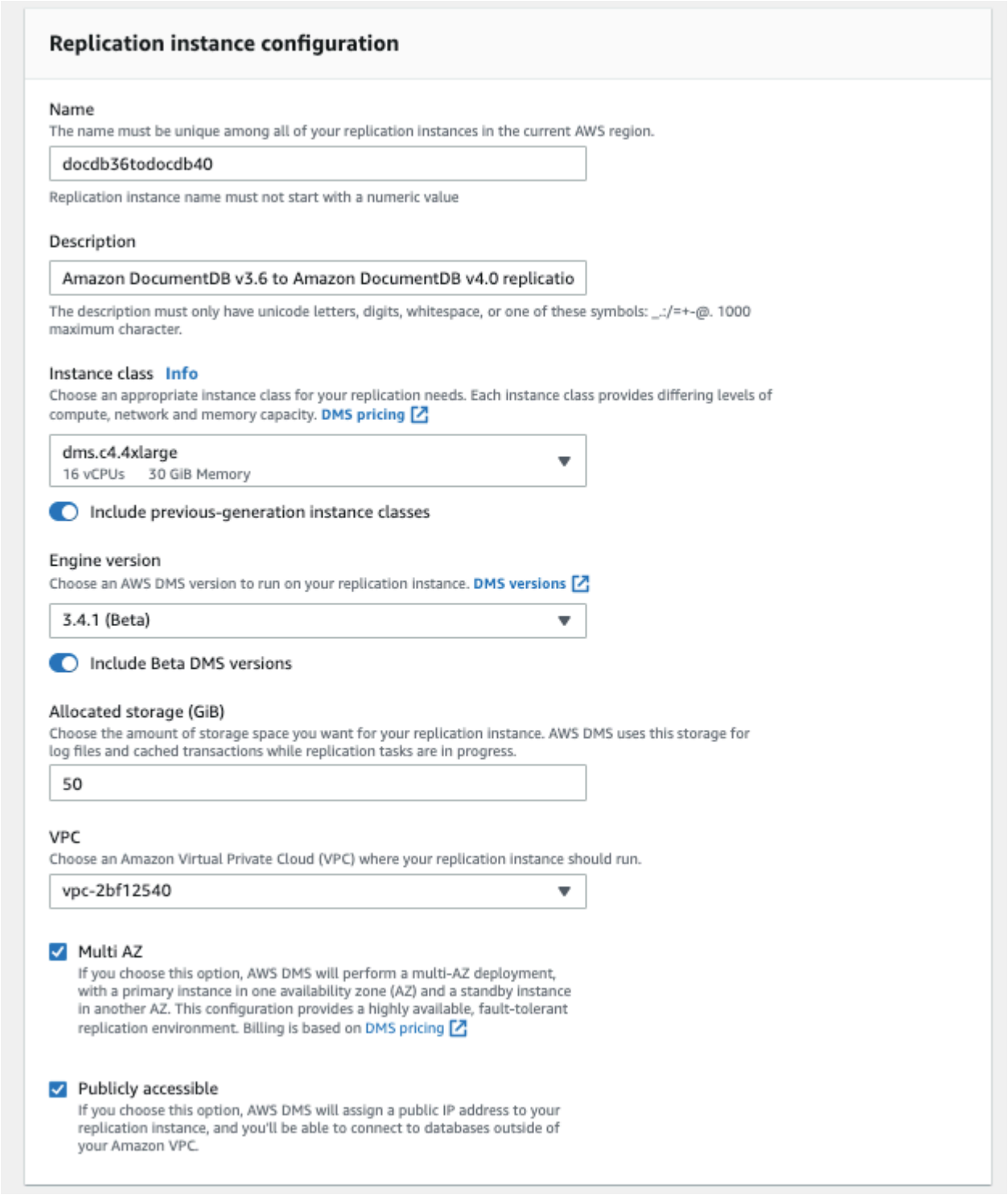
AWS DMS レプリケーションインスタンスは、ソース Amazon DocumentDB クラスターに接続してデータを読み取り、ターゲット Amazon DocumentDB クラスターに書き込みます。 AWS DMS レプリケーションインスタンスは、一括ロードオペレーションと CDC オペレーションの両方を実行できます。この処理のほとんどはメモリ内で行われます。ただし、大きいトランザクションではディスク上でのバッファリングが必要になることがあります。キャッシュされたトランザクションとログファ

イルもディスクに書き込まれます。データが移行されると、レプリケーションインスタンスは変更イベントをストリーミングして、ソースとターゲットが同期していることを確認します。

AWS DMS レプリケーションインスタンスを作成するには：

1. AWS DMS コンソールを開きます。
2. ナビゲーションペインで [Replication instances] (レプリケーション インスタンス) を選択します。
3. [Create replication instance (レプリケーションインスタンスの作成)] を選択し、次の情報を入力します。

   - 名前については、希望する名前を入力します。例えば、`docdb36todocdb40`。
   - 説明 については、選んだ説明を入力します。リストアイテムについては、Amazon DocumentDB 3.6 を Amazon DocumentDB 4.0 レプリケーションインスタンスに。
   - インスタンスクラス については、必要に応じてサイズを選択します。
   - エンジンバージョン については、`3.4.1.` を選択します。
   - Amazon VPC については、ソースとターゲットの Amazon DocumentDB クラスターを収容する Amazon VPC を選択します。
   - ストレージ割り当て (GiB) については、デフォルトの 50 GiB を使用します。書き込みスループットのワークロードが高い場合は、ワークロードに合わせてこの値を増やします。
   - マルチ AZ については、高可用性とフェイルオーバーサポートが必要な場合は はい を選択します。
   - [Publicly accessible (パブリックアクセス可能)] を有効にします。



4. [Create replication instance] (レプリケーションインスタンスの作成) を選択します。

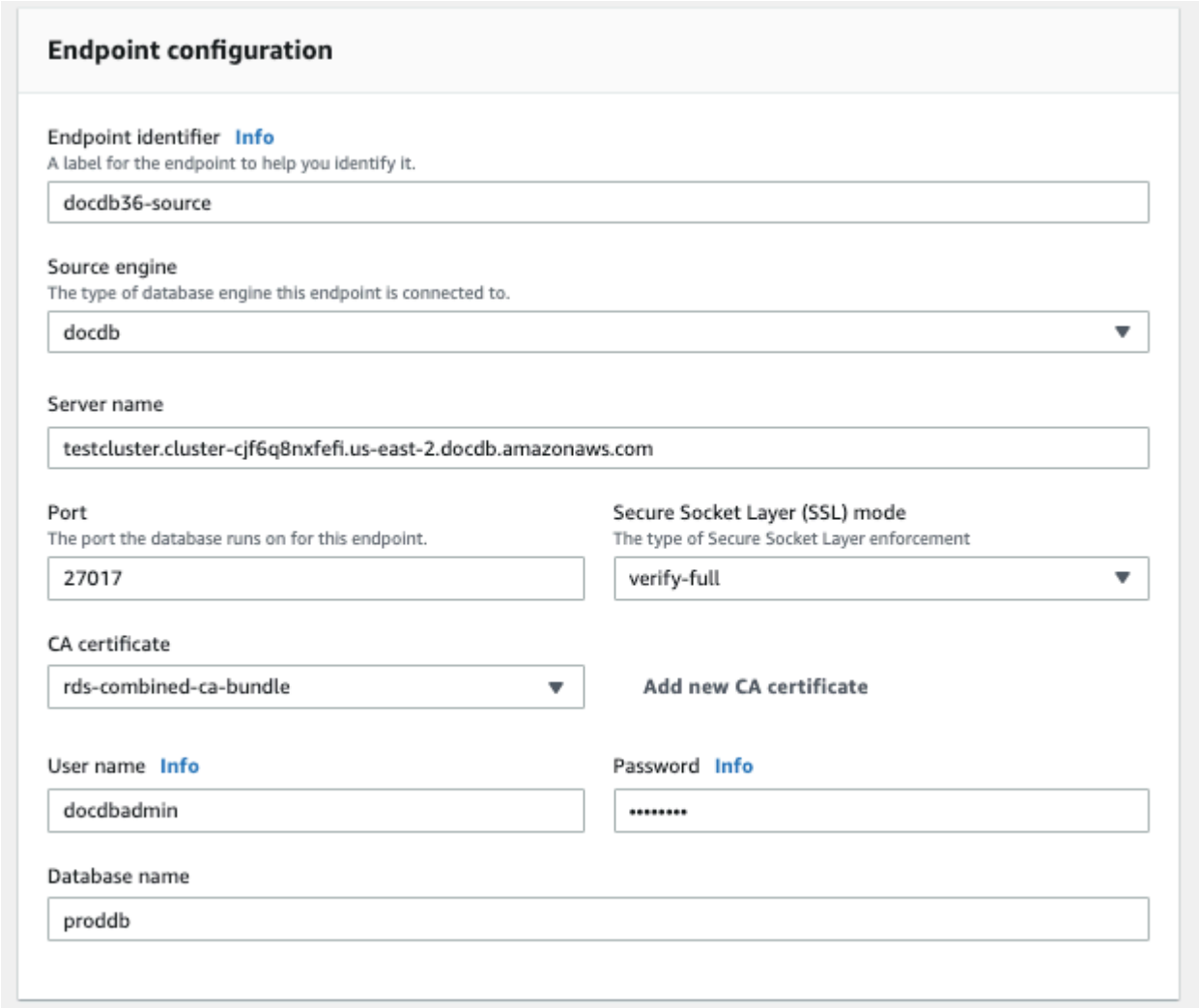
ステップ 5: AWS DMS ソースエンドポイントを作成する

ソースエンドポイントは、ソース Amazon DocumentDB クラスターに使用されます。

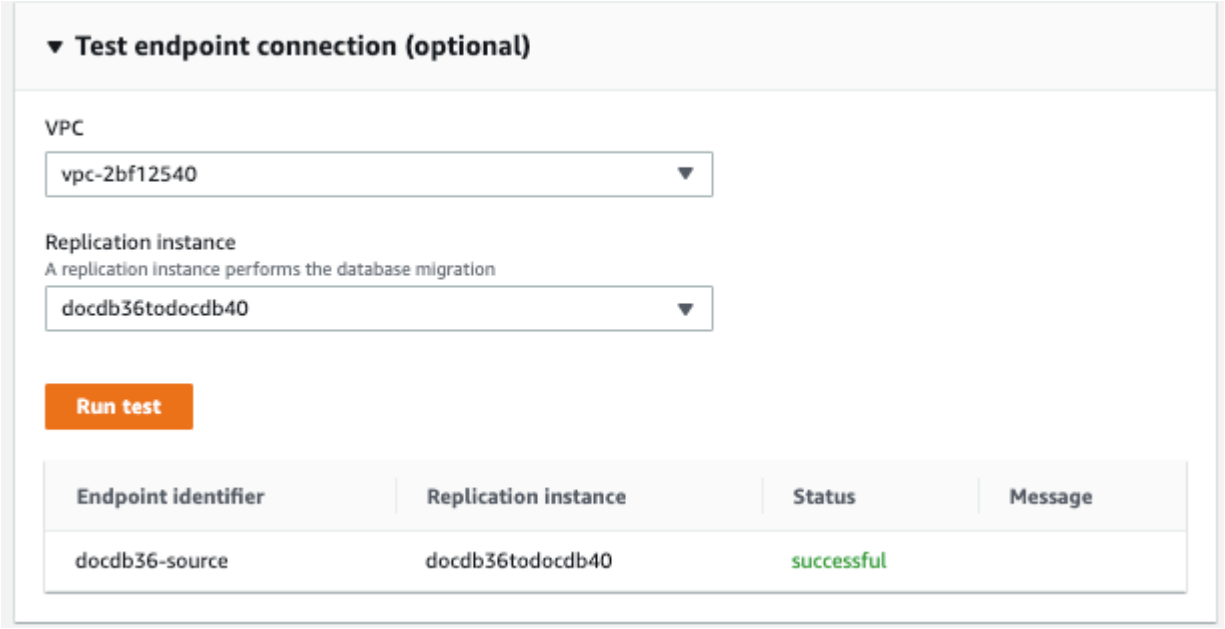
ソースエンドポイントを作成するには

1. AWS DMS コンソールを開きます。
2. ナビゲーションペインで、[エンドポイント] を選択します。
3. `Create endpoint` を選択し、以下の情報を入力します。

   - [Endpoint type (エンドポイントタイプ)] で [Source (ソース)] を選択します。
   - >エンドポイント識別子 については、覚えやすい名前を入力します (例: `docdb-source`)。
   - ソースエンジン については、`docdb` を選択します。
   - サーバー名 については、Amazon DocumentDB クラスターの DNS 名を入力します。
   - ポート については、ソース Amazon DocumentDB クラスターのポート番号を入力します。
   - SSL モード については、`verify-full` を選択します。
   - CA 証明書 については、新しい CA 証明書を追加する を選択します。TLS 接続バンドルを作成するために、新しい CA 証明書 をダウンロードします。証明書識別子 については、`rds-combined-ca-bundle` を入力します。[Import certificate file (証明書ファイルのインポート)] で、[Choose file (ファイルの選択)] を選択し、以前にダウンロードした `.pem` ファイルに移動します。ファイルを選択して開きます。証明書をインポートする を選択し、次に 証明書を選択する のドロップダウンから `rds-combined-ca-bundle` を選択します。
   - [ユーザーネーム] には、アップグレード元の Amazon DocumentDB クラスターのプライマリユーザー名を入力します。
   - [パスワード] には、アップグレード元の Amazon DocumentDB クラスターのプライマリパスワードを入力します。
   - データベース については、アップグレードするデータベース名を入力します。



4. 接続をテストして、正常にセットアップされたことを確認します。



5. [エンドポイントの作成] を選択します。

> **Note**
>
> AWS DMS は一度に 1 つのデータベースのみを移行できます。

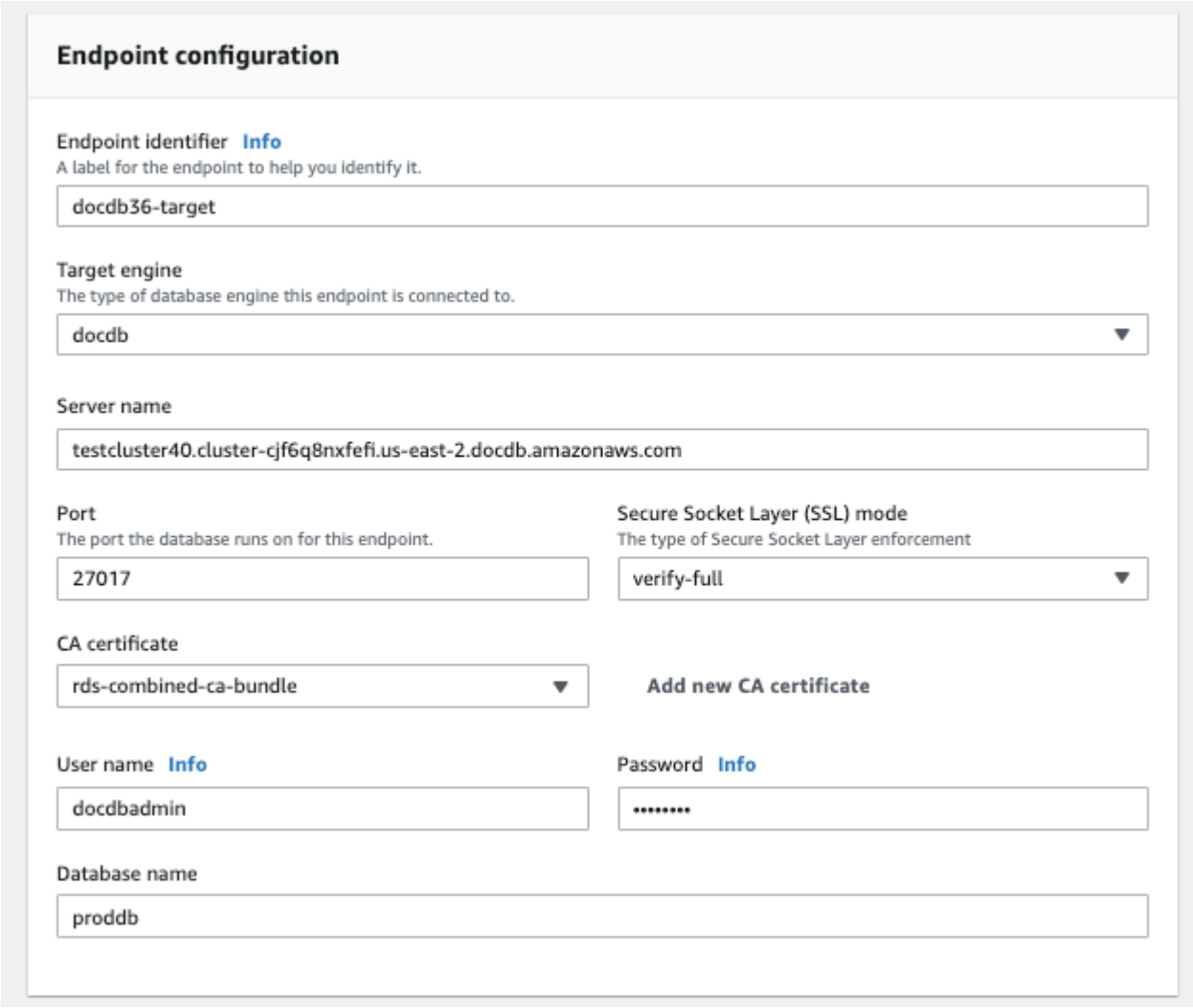
ステップ 6: AWS DMS ターゲットエンドポイントを作成する

ターゲットエンドポイントは Amazon DocumentDB クラスター用です。

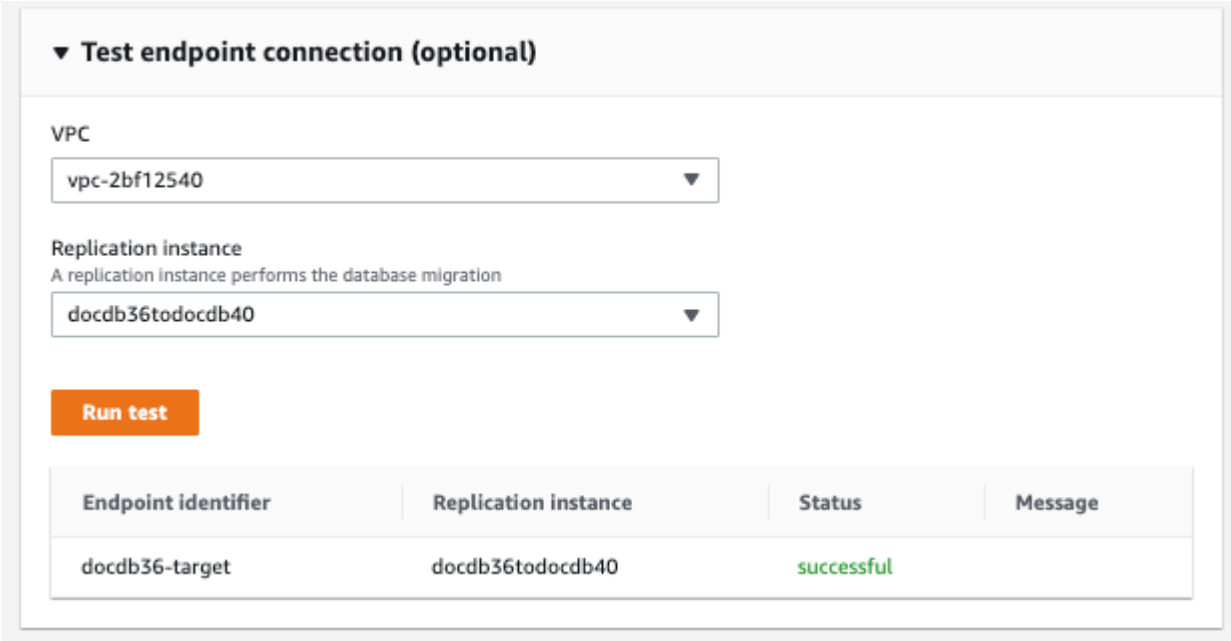
ターゲットエンドポイントを作成するには

1. AWS DMS コンソールを開きます。
2. ナビゲーションペインで、[エンドポイント] を選択します。
3. [Create endpoint (エンドポイントの作成)] を選択し、次の情報を入力します。
   - [Endpoint type (エンドポイントタイプ)] で [Target (ターゲット)] を選択します。
   - [Endpoint identifier (エンドポイント識別子)] に、覚えやすい名前を入力します (例: `docdb-target`)。
   - ソースエンジン については、`docdb` を選択します。
   - サーバー名 については、ターゲット Amazon DocumentDB クラスターの DNS 名を入力します。

- ポート については、ターゲット Amazon DocumentDB クラスターのポート番号を入力します。
- SSL モード については、`verify-full` を選択します。
- CA 証明書については、証明書を選択する というドロップダウンから既存の `rds-combined-ca-bundle` 証明書を選択します。
- [ユーザーネーム] には、アップグレード先の Amazon DocumentDB クラスターのプライマリユーザー名を入力します。
- [パスワード] については、アップグレード先の Amazon DocumentDB クラスターのプライマリパスワードを入力します。
- データベース名 については、ソースエンドポイントの設定に使用したのと同じデータベース名を入力します。



4. 接続をテストして、正常にセットアップされたことを確認します。



5. [エンドポイントの作成] を選択します。

ステップ 7: 移行タスクの作成と実行

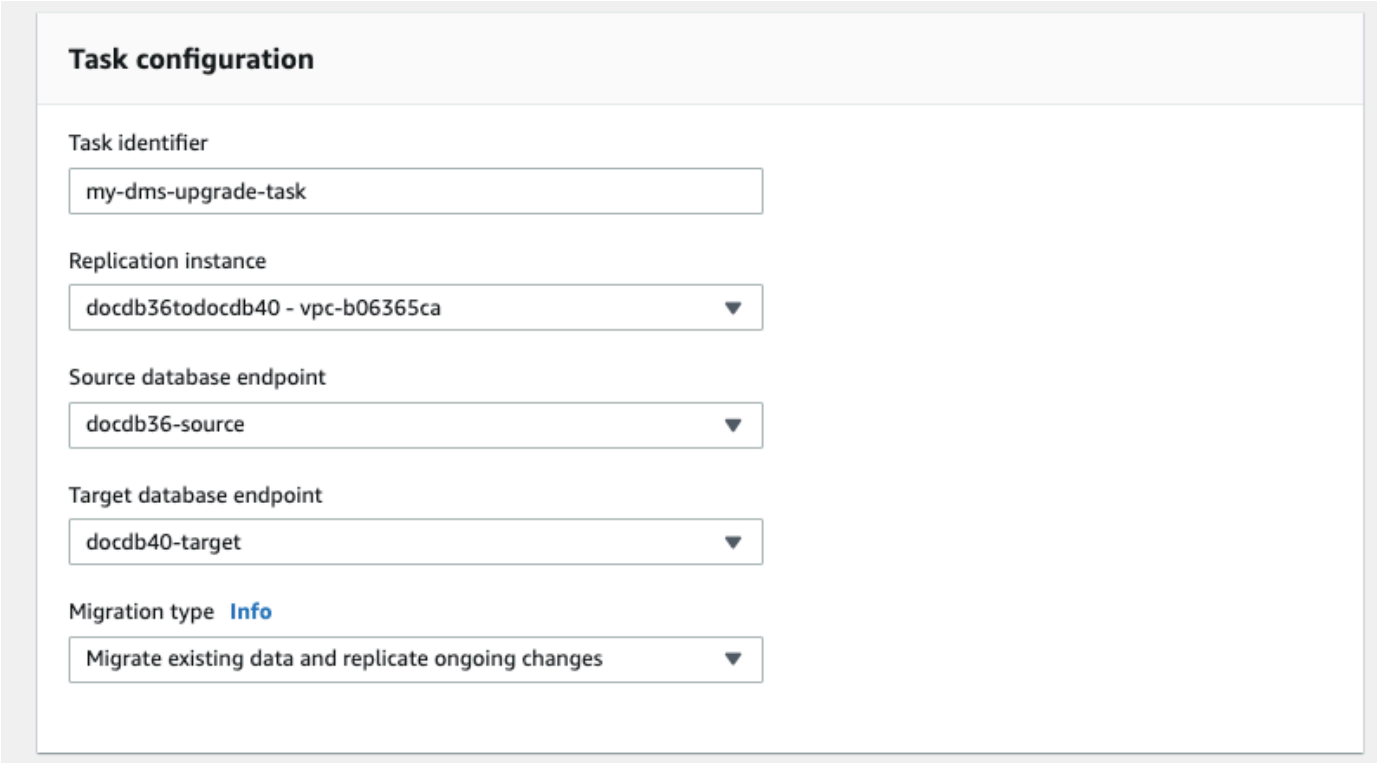
AWS DMS タスクは、レプリケーションインスタンスをソースインスタンスとターゲットインスタンスにバインドします。タスクを作成するときに、ソースエンドポイント、ターゲットエンドポイント、レプリケーションインスタンスおよび移行設定を特定します。 AWS DMS タスクは、既存のデータの移行、既存のデータの移行、進行中の変更のレプリケート、データ変更のレプリケートのみの 3 つの異なる移行タイプで作成できます。このウォークスルーの目的は、Amazon DocumentDB クラスターを最小限のダウンタイムでアップグレードすることなので、手順では既存のデータを移行し、進行中の変更をレプリケートするオプションを利用します。このオプションでは、 は既存のデータの移行中に変更を AWS DMS キャプチャします。 は、バルクデータがロードされた後も変更をキャプチャして適用 AWS DMS し続けます。最終的にソースデータベースとターゲットデータベースは同期され、移行に伴うダウンタイムは最小限に抑えられます。

以下は、ダウンタイムを最小限に抑えた移行のための移行タスクの作成手順です。

1. AWS DMS コンソールを開きます。
2. ナビゲーションペインで、[データベース移行タスク] を選択します。
3. データベース移行タスクの作成を選択し、タスク設定セクションに次の情報を入力します。

   - [タスク識別子] に、覚えやすい名前を入力します (例: `my-dms-upgrade-task`)。

- 記述型 Amazon リソースネーム (ARN) には、デフォルトの DMS ARN を上書きするわかりやすい名前を入力します。
- [Replication instance (レプリケーションインスタンス)] で、「ステップ 4: AWS DMS レプリケーションインスタンスを作成する」で作成したレプリケーションインスタンスを選択します。
- ソースデータベースエンドポイントで、 で作成したソースエンドポイントを選択しますステップ 5: AWS DMS ソースエンドポイントを作成する。
- ターゲットデータベースエンドポイントで、 で作成したターゲットエンドポイントを選択しますステップ 6: AWS DMS ターゲットエンドポイントを作成する。
- 移行タイプで、移行とレプリケートを選択します。

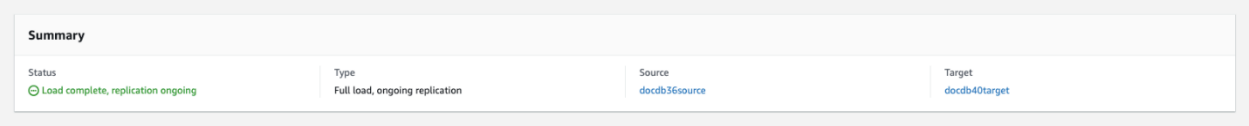


4. タスク設定セクションに次の情報を入力します。

- ターゲットテーブル準備モードセクションで、何もしないを選択します。これにより、ステップ 3 で作成したインデックスが削除されないようになります。
- タスクログのサブセクションで、CloudWatch ログを有効にするを選択します。
- 移行タスクのスタートアップ の設定については、[作成時に自動で設定する] を選択します。これにより、移行タスクを作成すると、自動的に移行タスクが開始されます。

- データベース移行タスクの作成を選択します。

AWS DMS は、ソース Amazon DocumentDB クラスターからターゲット Amazon DocumentDB クラスターへのデータの移行を開始するようになりました。タスクのステータスが、[Starting (開始中)] から [Running (実行中)] に変わります。 AWS DMS コンソールでタスクを選択すると、進行状況をモニタリングできます。数分/時間後 (移行のサイズに応じて)、ステータスが [ロード完了、レプリケーション継続中] に変わります。つまり、 AWS DMS はソース Amazon DocumentDB クラスターからターゲット Amazon DocumentDB クラスターへのフルロード移行を完了し、変更イベントをレプリケートしています。



最終的には、ソースとターゲットが同期します。同期しているかどうかを確認するためには、コレクションの `count()` 操作で、すべての変更イベントが移行されたことを確認できます。

ステップ 8: アプリケーションエンドポイントから Amazon DocumentDB クラスター への変更

フルロードが完了し、CDC プロセスが継続的にレプリケートされると、アプリケーションのデータベース接続エンドポイントをソース Amazon DocumentDB クラスターからターゲット Amazon DocumentDB クラスター クラスターに変更できます。

Amazon DocumentDB のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS とお客様の間の責任共有です。このドキュメントは、Amazon DocumentDB 使用時における責任共有モデルの適用法を理解するのに役立ちます。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ — AWS クラウドで AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。 は、安全に使用できるサービス AWS も提供します。「AWS 」 コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。Amazon DocumentDB (MongoDB 互換) に適用されるコンプライアンスプログラムについては、「AWS コンプライアンスプログラム別の適用範囲内のサービス」をご覧ください。
- クラウド内のセキュリティ — お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、お客様のデータの機密性、組織の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

Amazon DocumentDB は、「連邦リスク承認管理プログラム (FedRAMP)」の認可事業者です。これには AWS GovCloud (米国) リージョンの FedRAMP High 認可と AWS 、米国東部/西部リージョンの FedRAMP Moderate 認可があります。 AWS とコンプライアンスの取り組みの詳細については、AWS「コンプライアンスプログラムによる対象範囲内のサービス」を参照してください。

> **Note**
>
> この章は、インスタンスベースのクラスターと Elastic クラスターの両方に適用されます。詳細については、以下のトピックを参照してください。

また、Amazon DocumentDB リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。以下のトピックでは、セキュリティとコンプライアンスの目的を満たすように Amazon DocumentDB を設定する方法について説明します。

トピック

- Amazon DocumentDB と によるパスワード管理 AWS Secrets Manager
- Amazon DocumentDB におけるデータ保護

- Amazon DocumentDB の Identity and Access Management
- IAM ID を使用した認証
- Amazon DocumentDB ユーザーの管理
- ロールベースのアクセス制御 (RBAC) を使用したデータベースへのアクセス
- Amazon DocumentDB でのログ記録とモニタリング
- Amazon DocumentDB TLS 証明書の更新
- Amazon DocumentDB TLS 証明書の更新 - GovCloud
- Amazon DocumentDB のコンプライアンス検証
- Amazon DocumentDB の耐障害性
- Amazon DocumentDB のインフラストラクチャセキュリティ
- Amazon DocumentDB API とインターフェイス VPC エンドポイント (AWS PrivateLink) について
- Amazon DocumentDB のセキュリティベストプラクティス
- Amazon DocumentDB イベントの監査

Amazon DocumentDB と によるパスワード管理 AWS Secrets Manager

Amazon DocumentDB は Secrets Manager と統合して、クラスターのプライマリユーザーパスワードを管理します。

トピック

- Secrets Manager と Amazon DocumentDB の統合に関する制限事項
- を使用したプライマリユーザーパスワードの管理の概要 AWS Secrets Manager
- でのプライマリユーザーパスワードの Amazon DocumentDB 管理の強制 AWS Secrets Manager
- Secrets Manager を使用したクラスターのプライマリユーザーパスワードの管理

Secrets Manager と Amazon DocumentDB の統合に関する制限事項

Secrets Manager を使用したプライマリユーザーパスワードの管理は、以下の機能ではサポートされていません。

- Amazon DocumentDB グローバルデータベースの一部であるクラスター
- Amazon DocumentDB クロスリージョンリードレプリカ

を使用したプライマリユーザーパスワードの管理の概要 AWS Secrets Manager

を使用すると AWS Secrets Manager、データベースパスワードを含むコード内のハードコードされた認証情報を Secrets Manager への API コールに置き換えて、プログラムでシークレットを取得できます。Secrets Manager の詳細については、AWS Secrets Manager ユーザーガイドを参照してください。

Secrets Manager にデータベースシークレットを保存すると、 AWS アカウントで料金が発生します。料金については、「AWS Secrets Manager の料金」を参照してください。

次のいずれかの操作を実行するときに、Amazon DocumentDB が Amazon DocumentDB クラスターの Secrets Manager でプライマリユーザーパスワードを管理するように指定できます。

- クラスターを作成する
- クラスターを変更する

Amazon DocumentDB が Secrets Manager でプライマリユーザーパスワードを管理するように指定すると、Amazon DocumentDB はパスワードを生成して Secrets Manager に保存します。シークレットを直接操作して、プライマリユーザーの認証情報を取得できます。また、カスタマーマネージドキーを指定してシークレットを暗号化したり、Secrets Manager が提供する KMS キーを使用したりすることもできます。

Amazon DocumentDB はシークレットの設定を管理し、デフォルトで 7 日ごとにシークレットをローテーションします。ローテーションスケジュールなど、一部の設定を変更できます。Secrets Manager でシークレットを管理するクラスターを削除すると、シークレットと関連するメタデータも削除されます。

シークレット内の認証情報を使用してクラスターに接続するには、Secrets Manager からシークレットを取得できます。詳細については、「 AWS Secrets Manager ユーザーガイド」の「 シークレット AWS Secrets Managerの認証情報を使用して JDBC を使用して からシーク AWS Secrets Manager レットを取得し、SQL データベースに接続する」を参照してください。

でのプライマリユーザーパスワードの Amazon DocumentDB 管理の強制 AWS Secrets Manager

IAM 条件キーを使用して、プライマリユーザーパスワードの Amazon DocumentDB 管理を適用できます AWS Secrets Manager。次のポリシーでは、プライマリユーザーのパスワードが Secrets

Manager の Amazon DocumentDB によって管理されていない限り、ユーザーがインスタンスまたはクラスターを作成または復元することはできません。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "rds:CreateDBCluster"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
            }
        }
    ]
}
```

Secrets Manager を使用したクラスターのプライマリユーザーパスワードの管理

次のアクションを実行するときに、Secrets Manager でプライマリユーザーパスワードの Amazon DocumentDB 管理を設定できます。

- Amazon DocumentDB クラスターの作成
- Amazon DocumentDB クラスターの変更

Amazon DocumentDB コンソールまたは を使用して AWS CLI 、これらのアクションを実行できます。

Amazon DocumentDB におけるデータ保護

責任 AWS 共有モデル、Amazon DocumentDB のデータ保護 (MongoDB 互換) に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された AWS 責任共有モデルおよび GDPR のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、その中のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して Amazon DocumentDB AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

トピック

- クライアント側のフィールドレベルの暗号化
- Amazon DocumentDB データの保管時の暗号化
- Encrypting data in transit
- キー管理

クライアント側のフィールドレベルの暗号化

Amazon DocumentDB のクライアント側のフィールドレベル暗号化 (FLE) を使用すると、Amazon DocumentDB クラスターに転送される前に、クライアントアプリケーションの機密データを暗号化できます。機密データは、クラスターで保存および処理されるときは暗号化されたままで、取得時にはクライアントアプリケーションで復号化されます。

トピック

- 入門
- クライアント側 FLE でのクエリ
- 制限

入門

Amazon DocumentDB でのクライアント側 FLE の初期設定は、暗号化キーの作成、アプリケーションへのロールの関連付け、アプリケーションの設定、暗号化オプションによる CRUD オペレーションの定義を含む 4 ステップのプロセスです。

トピック

- ステップ 1: 暗号化キーの作成
- ステップ 2: ロールとアプリケーションの関連付け
- ステップ 3: アプリケーションの設定
- ステップ 4: CRUD オペレーションの定義
- 例:クライアント側のフィールドレベル暗号化設定ファイル

ステップ 1: 暗号化キーの作成

を使用して AWS Key Management Service、機密データフィールドの暗号化と復号に使用される対称キーを作成し、必要な IAM 使用権限を提供します。 は、データキー (DKs) の暗号化に使用される

カスタマーキー (CK) AWS KMS を保存します。セキュリティ体制を強化するため、カスタマーキーは KMS に保存することをお勧めします。データキーは Amazon DocumentDB コレクションに保存されるセカンダリキーで、Amazon DocumentDB にドキュメントを保存する前に機密フィールドを暗号化するために必要です。カスタマーキーはデータキーを暗号化し、データキーはデータを暗号化および復号化します。グローバルクラスターを使用している場合は、さまざまなリージョンのさまざまなサービスロールで使用できるマルチリージョンキーを作成できます。

キーの作成方法など AWS Key Management Service、 の詳細については、AWS 「 Key Management Service デベロッパーガイド」を参照してください。

ステップ 2: ロールとアプリケーションの関連付け

適切な AWS KMS アクセス許可を持つ IAM ポリシーを作成します。このポリシーは、アタッチされた IAM アイデンティティで、リソースフィールドで指定された KMS キーの暗号化と復号化ができるようにします。アプリケーションはこの IAM ロールを引き受けて認証します AWS KMS。

ポリシーは次のようになります。

```
{ "Effect": "Allow",
"Action": ["kms:Decrypt", "kms:Encrypt"],
"Resource": "Customer Key ARN"
}
```

ステップ 3: アプリケーションの設定

これで、 でカスタマーキーを定義 AWS KMS し、IAM ロールを作成して、カスタマーキーにアクセスするための適切な IAM アクセス許可を付与しました。必要なパッケージをインポートします。

```
import boto3
import json
import base64
from pymongo import MongoClient
from pymongo.encryption import (Algorithm,
                                ClientEncryption)
```

```
# create a session object:
my_session = boto3.session.Session()

# get access_key and secret_key programmatically using get_frozen_credentials() method:
 current_credentials = my_session.get_credentials().get_frozen_credentials()
```

1. KMS プロバイダーのタイプに「aws」を指定し、前のステップで取得したアカウント認証情報を入力します。

```
provider = "aws"
kms_providers = {
    provider: {
        "accessKeyId": current_credentials.access_key,
        "secretAccessKey": current_credentials.secret_key
    }
}
```

2. データキーの暗号化に使用するカスタマーキーを指定します。

```
customer_key = {
“region”: “AWS region of the customer_key”,
    “key”: “customer_key ARN”
}

key_vault_namespace = "encryption.dataKeys"

key_alt_name = 'TEST_DATA_KEY'
```

3. MongoClient オブジェクトを設定します。

```
client = MongoClient(connection_string)

coll = client.test.coll
coll.drop()

client_encryption = ClientEncryption(
    kms_providers, # pass in the kms_providers variable from the previous step
    key_vault_namespace = key_vault_namespace,
    client,
    coll.codec_options
)
```

4. データキーを生成します。

```
data_key_id = client_encryption.create_data_key(provider,
    customer_key,
    key_alt_name = [key_alt_name])
```

5. 既存のデータキーを取得します。

```
data_key = DataKey("aws",
    master_key = customer_key)
key_id = data_key["_id"]
data_key_id = client[key_vault_namespace].find_one({"_id": key_id})
```

ステップ 4: CRUD オペレーションの定義

CRUD オペレーションを暗号化オプションで定義します。

1. 1 つのドキュメントの書き込み/読み取り/削除を行うコレクションを定義します。

```
coll = client.gameinfo.users
```

2. 明示的な暗号化 - フィールドを暗号化して以下を挿入します。

> **Note**
>
> 「key_id」または「key_alt_name」の 1 つを正確に指定する必要があります。

```
encrypted_first_name = client_encryption.encrypt(
    "Jane",
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
    key_alt_name=data_key_id
)
encrypted_last_name = client_encryption.encrypt(
    "Doe",
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
    key_alt_name=data_key_id
)
encrypted_dob = client_encryption.encrypt(
    "1990-01-01",
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Random,
    key_alt_name=data_key_id
)

coll.insert_one(
    {"gamerTag": "jane_doe90",
    "firstName": encrypted_first_name,
    "lastName": encrypted_last_name,
```

```
    "dateOfBirth":encrypted_dob,
    "Favorite_games":["Halo","Age of Empires 2","Medal of Honor"]
})
```

例:クライアント側のフィールドレベル暗号化設定ファイル

次の例では、各 ############## を独自の情報に置き換えます。

```
# import python packages:
import boto3
import json
import base64
from pymongo import MongoClient
from pymongo.encryption import (Algorithm,
                                ClientEncryption)

def main():

    # create a session object:
    my_session = boto3.session.Session()

    # get aws_region from session object:
    aws_region = my_session.region_name

    # get access_key and secret_key programmatically using get_frozen_credentials()
 method:
    current_credentials = my_session.get_credentials().get_frozen_credentials()
    provider = "aws"

    # define the kms_providers which is later used to create the Data Key:
    kms_providers = {
        provider: {
            "accessKeyId": current_credentials.access_key,
            "secretAccessKey": current_credentials.secret_key
        }
    }

    # enter the kms key ARN. Replace the example ARN value.
    kms_arn = "arn:aws:kms:us-east-1:123456789:key/abcd-efgh-ijkl-mnop"
    customer_key = {
        "region": aws_region,
        "key":kms_arn
    }
```

```
    # secrets manager is used to strore and retrieve user credentials for connecting to
 an Amazon DocumentDB cluster.
    # retrieve the secret using the secret name. Replace the example secret key.
    secret_name = "/dev/secretKey"
    docdb_credentials = json.loads(my_session.client(service_name = 'secretsmanager',
 region_name = "us-east-1").get_secret_value(SecretId = secret_name)['SecretString'])

    connection_params = '/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false'
    conn_str = 'mongodb://' + docdb_credentials["username"] + ':' +
 docdb_credentials["password"] + '@' + docdb_credentials["host"] + ':' +
 str(docdb_credentials["port"]) + connection_params
    client = MongoClient(conn_str)

    coll = client.test.coll
    coll.drop()

    # store the encryption data keys in a key vault collection (having naming
 convention as db.collection):
    key_vault_namespace = "encryption.dataKeys"
    key_vault_db_name, key_vault_coll_name = key_vault_namespace.split(".", 1)

    # set up the key vault (key_vault_namespace) for this example:
    key_vault = client[key_vault_db_name][key_vault_coll_name]
    key_vault.drop()
    key_vault.create_index("keyAltNames", unique=True)

    client_encryption = ClientEncryption(
        kms_providers,
        key_vault_namespace,
        client,
        coll.codec_options)

    # create a new data key for the encrypted field:
    data_key_id = client_encryption.create_data_key(provider, master_key=customer_key,
 key_alt_names=["some_key_alt_name"], key_material = None)

    # explicitly encrypt a field:
    encrypted_first_name = client_encryption.encrypt(
    "Jane",
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
    key_id=data_key_id
    )
```

```
    coll.insert_one(
    {"gamerTag": "jane_doe90",
    "firstName": encrypted_first_name
    })
    doc = coll.find_one()
    print('Encrypted document: %s' % (doc,))

    # explicitly decrypt the field:
    doc["encryptedField"] = client_encryption.decrypt(doc["encryptedField"])
    print('Decrypted document: %s' % (doc,))

    # cleanup resources:
    client_encryption.close()
    client.close()

    if __name__ == "__main__":
        main()
```

クライアント側 FLE でのクエリ

Amazon DocumentDB は、クライアント側 FLE によるポイント等式クエリをサポートしています。不等式クエリや比較クエリは、不正確な結果を返す可能性があります。読み取りオペレーションと書き込みオペレーションは、復号化された値に対して同じオペレーションを実行する場合と比較して、予期しない動作や誤った動作をする可能性があります。

たとえば、ゲーマースコアが 500 を超えるドキュメントのフィルターをクエリするには:

```
db.users.find( {
    "gamerscore" : { $gt : 500 }
})
```

クライアントは明示的な暗号化の方法を使用してクエリ値を暗号化します。

```
encrypted_gamerscore_filter = client_encryption.encrypt(
    500,
        Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
        key_alt_name=data_key_id
        )

db.users.find( {
    "gamerscore" : { $gt : encrypted_gamerscore_filter }
} )
```

検索オペレーションでは、Amazon DocumentDB は、暗号化された値 500 と各ドキュメントに保存されている暗号化されたフィールド値を不等号チェックを使用して比較します。検索オペレーションの不等式チェックは、復号化されたデータと値を使用して実行すると、結果の生成に成功しても、異なる結果が返されることがあります。

制限

Amazon DocumentDB のクライアント側のフィールドレベル暗号化には、以下の制限が適用されます。

- Amazon DocumentDB がサポートするのは、ポイント等式クエリのみです。不等式クエリや比較クエリは、不正確な結果を返す可能性があります。読み取りオペレーションと書き込みオペレーションは、復号化された値に対して同じオペレーションを実行する場合と比較して、予期しない動作や誤った動作をする可能性があります。ゲーマースコアが 500 を超えるドキュメントのフィルターをクエリするには、

  ```
  db.users.find( {
      "gamerscore" : { $gt : 500 }
      })
  ```

  クライアントは明示的な暗号化の方法を使用してクエリ値を暗号化します。

  ```
  encrypted_gamerscore_filter = client_encryption.encrypt(
      500,
      Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
      key_alt_name=data_key_id
  )

  db.users.find({
      "gamerscore" : { $gt : encrypted_gamerscore_filter }
  })
  ```

  検索オペレーションでは、Amazon DocumentDB は、暗号化された値 500 と各ドキュメントに保存されている暗号化されたフィールド値を不等号チェックを使用して比較します。検索オペレーションの不等式チェックは、復号化されたデータと値を使用して実行すると、結果の生成に成功しても、異なる結果が返されることがあります。

- Amazon DocumentDB は、Mongo Shell からの明示的なクライアント側 FLE をサポートしていません。ただし、この機能はサポートされているどのドライバーでも動作します。

Amazon DocumentDB データの保管時の暗号化

> **Note**
>
> AWS KMS は、カスタマーマスターキー (CMK) という用語を AWS KMS keyおよび KMS キーに置き換えています。この概念に変更はありません。重大な変更を防ぐために、 AWS KMS は、この用語のいくつかのバリエーションを保持しています。

クラスターを作成するときに、ストレージの暗号化オプションを指定して、Amazon DocumentDB クラスター内の保存データを暗号化できます。ストレージの暗号化はクラスター全体で有効になり、プライマリインスタンスとレプリカを含むインスタンス全体に適用されます。また、クラスターのストレージボリューム、データ、インデックス、ログ、自動バックアップ、スナップショットにも適用されます。

Amazon DocumentDB は、256 ビット Advanced Encryption Standard (AES-256) を使用して、 AWS Key Management Service () に保存されている暗号化キーを使用してデータを暗号化しますAWS KMS。保存暗号化が有効な Amazon DocumentDB クラスターを使用する場合、アプリケーションロジックやクライアント接続を変更する必要はありません。Amazon DocumentDB はパフォーマンスの影響を最小限に抑えながら、データの暗号化と復号を透過的に処理します。

Amazon DocumentDB は と統合 AWS KMS され、エンベロープ暗号化と呼ばれる方法を使用してデータを保護します。Amazon DocumentDB クラスターが で暗号化されると AWS KMS、Amazon DocumentDB は KMS キーを使用して暗号化テキストデータキーを生成し、ストレージボリュームを暗号化するように AWS KMS に要求します。暗号化テキストデータキーは、定義する KMS を使用して暗号化され、暗号化されたデータおよびストレージメタデータとともに保存されます。Amazon DocumentDB が暗号化されたデータにアクセスする必要がある場合、KMS キーを使用して暗号化テキストデータキーを復号 AWS KMS 化するようにリクエストし、プレーンテキストデータキーをメモリにキャッシュして、ストレージボリューム内のデータを効率的に暗号化および復号します。

Amazon DocumentDB のストレージ暗号化機能は、サポートされているすべてのインスタンスサイズと、 AWS リージョン Amazon DocumentDB が利用可能なすべての で使用できます。

Amazon DocumentDB クラスターで保管時の暗号化を有効にする

クラスターが または AWS Command Line Interface () を使用してプロビジョニングされている場合、Amazon DocumentDB クラスターで保管時の暗号化を有効 AWS Management Console または無効にできますAWS CLI。コンソールを使用して作成したクラスターでは、デフォルトで保管時の

暗号化が有効になっています。を使用して作成したクラスター AWS CLI では、保管時の暗号化がデフォルトで無効になっています。したがって、保存時の暗号化は `--storage-encrypted` パラメータを使用して明示的に有効にする必要があります。いずれの場合も、クラスターの作成後は、保存時の暗号化オプションを変更することはできません。

Amazon DocumentDB は AWS KMS 、 を使用して暗号化キーを取得および管理し、これらのキーの使用方法を制御するポリシーを定義します。 AWS KMS キー識別子を指定しない場合、Amazon DocumentDB はデフォルトの AWS マネージドサービス KMS キーを使用します。Amazon DocumentDB は、 AWS リージョン 内の各 に個別の KMS キーを作成します AWS アカウント。詳細については、「AWS Key Management Service の概念」を参照してください。

独自の KMS キーの作成を開始するには、開始方法 の AWS Key Management Service デベロッパーガイド を参照してください。

> **Important**
>
> Amazon DocumentDB がサポートしているのは対称 KMS キーのみであるため、クラスターの暗号化には対称暗号化 KMS キーを使用して必要があります。Amazon DocumentDB クラスター内のデータの暗号化に非対称 KMS キー を使用しないでください。詳細については、AWS Key Management Service デベロッパーガイドの「AWS KMSの非対称キー」を参照してください。

Amazon DocumentDB がクラスターの暗号化キーにアクセスできなくなった場合、例えばキーへのアクセス権が失効した場合など、暗号化されたクラスターは終了状態になります。この場合、クラスターはバックアップからのみ復元できます。Amazon DocumentDB では、バックアップは常に 1 日間有効になります。

また、暗号化された Amazon DocumentDB クラスターのキーを無効にすると、最終的にそのクラスタに対する読み取りおよび書き込みのアクセス権は失われます。Amazon DocumentDB で、アクセスできないキーで暗号化されたクラスターが発生すると、クラスターが終了状態になります。この場合、クラスターは使用できなくなり、データベースの現在の状態を復元することはできません。クラスターを復元するには、Amazon DocumentDB の暗号化キーへのアクセスを再び有効にした後、バックアップからクラスターを復元します。

> **Important**
>
> 暗号化されたクラスターの KMS キーを作成した後は、そのクラスターを変更することはできません。暗号化されたクラスターを作成する前に、暗号化キーの要件を確認してください。

Using the AWS Management Console

保管時の暗号化オプションは、クラスターの作成時に指定します。 AWS Management Console を使用してクラスターを作成すると、保管時の暗号化がデフォルトで有効になります。クラスターの作成後に変更することはできません。

クラスターの作成時に保管時の暗号化オプションを指定するには

1. Amazon DocumentDB クラスターの作成は、開始方法 セクションで説明されています。ただし、ステップ 6 では、[クラスターの作成] を選択しないでください。
2. [Authentication (認証)] セクションの下で、[Show advanced settings (詳細設定の表示)] を選択します。
3. [Encryption-at-rest (保管時の暗号化)] セクションまで下にスクロールします。
4. 保存時の暗号化オプションを選択します。どちらのオプションを選択しても、クラスターの作成後に変更することはできません。
   - このクラスターを保管時に暗号化するには、[暗号化の有効化] を選択します。
   - このクラスターで保存されているデータを暗号化しない場合は、[暗号化の無効化] を選択します。
5. 目的のプライマリキーを選択します。Amazon DocumentDB は、 AWS Key Management Service （ AWS KMS) を使用して暗号化キーを取得および管理し、これらのキーの使用方法を制御するポリシーを定義します。 AWS KMS キー識別子を指定しない場合、Amazon DocumentDB はデフォルトの AWS マネージドサービス KMS キーを使用します。詳細については、「AWS Key Management Service の概念」を参照してください。

   > **Note**
   >
   > 暗号化されたクラスターを作成したら、そのクラスターの KMS キーを変更することはできません。暗号化されたクラスターを作成する前に、暗号化キーの要件を確認してください。

6. 必要に応じて他のセクションを完了し、クラスターを作成します。

Using the AWS CLI

を使用して Amazon DocumentDB クラスターを暗号化するには AWS CLI、 `create-db-cluster` コマンドを実行し、 `--storage-encrypted`オプションを指定します。を使用して作成された Amazon DocumentDB クラスターは、デフォルトではストレージ暗号化を有効に AWS CLI しません。

次の例では、ストレージ暗号化を有効にした Amazon DocumentDB クラスターを作成します。

次の例では、各`###############`をクラスターの情報に置き換えます。

Example

Linux、macOS、Unix の場合:

```
aws docdb create-db-cluster \
  --db-cluster-identifier mydocdbcluster \
  --port 27017 \
  --engine docdb \
  --master-username SampleUser1 \
  --master-user-password primaryPassword \
  --storage-encrypted
```

Windows の場合:

```
aws docdb create-db-cluster ^
  --db-cluster-identifier SampleUser1 ^
  --port 27017 ^
  --engine docdb ^
  --master-username SampleUser1 ^
  --master-user-password primaryPassword ^
  --storage-encrypted
```

暗号化された Amazon DocumentDB クラスターを作成するときに、次の例のように AWS KMS キー識別子を指定できます。

Example

Linux、macOS、Unix の場合:

```
aws docdb create-db-cluster \
  --db-cluster-identifier SampleUser1 \
  --port 27017 \
  --engine docdb \
  --master-username primaryUsername \
  --master-user-password yourPrimaryPassword \
  --storage-encrypted \
  --kms-key-id key-arn-or-alias
```

Windows の場合:

```
aws docdb create-db-cluster ^
  --db-cluster-identifier SampleUser1 ^
  --port 27017 ^
  --engine docdb ^
  --master-username SampleUser1 ^
  --master-user-password primaryPassword ^
  --storage-encrypted ^
  --kms-key-id key-arn-or-alias
```

> **Note**
>
> 暗号化されたクラスターを作成したら、そのクラスターの KMS キーを変更することはできません。暗号化されたクラスターを作成する前に、暗号化キーの要件を確認してください。

Amazon DocumentDB 暗号化クラスターに関する制限事項

Amazon DocumentDB の暗号化されたクラスターには、以下の制限事項があります。

- Amazon DocumentDB クラスターのストレージの暗号化を有効または無効にできるのは、クラスターの作成後ではなく作成時のみです。ただし、暗号化されていないクラスターのスナップショットを作成し、暗号化されていないスナップショットを新しいクラスターとして復元する際に保存時の暗号化オプションを指定します。

  詳細については、以下の各トピックを参照してください。

  - 手動クラスタースナップショットの作成
  - クラスタースナップショットからの復元

  - Amazon DocumentDB クラスタースナップショットのコピー
- ストレージの暗号化を有効にした Amazon DocumentDB クラスターを変更して暗号化を無効にすることはできません。
- Amazon DocumentDB クラスターのすべてのインスタンス、自動バックアップ、スナップショット、インデックスは、同じ KMS キーで暗号化されます。

Encrypting data in transit

TLS (Transport Layer Security) を使用して、アプリケーションと Amazon DocumentDB クラスター間の接続を暗号化できます。デフォルトでは、転送時の暗号化は、新しく作成された Amazon DocumentDB クラスターに対して有効になっています。クラスターの作成時、または後で無効にすることもできます。転送中の暗号化が有効になっている場合、クラスターに接続するには TLS を使用したセキュアな接続が必要です。TLS を使用した Amazon DocumentDB への接続の詳細については、「プログラムによる Amazon DocumentDB への接続」を参照してください。

Amazon DocumentDB クラスター TLS 設定の管理

Amazon DocumentDB クラスターの転送中の暗号化は、クラスターパラメータグループ の TLS パラメータを使用して管理されます。Amazon DocumentDB クラスターの TLS 設定は、 AWS Management Console または AWS Command Line Interface () を使用して管理できますAWS CLI。現在の TLS 設定を確認して変更する方法の説明については、次のセクションを参照してください。

Using the AWS Management Console

コンソールを使用して TLS 向けにパフォーマンス管理タスクは、パラメータグループの識別、TLS 値の検証や必要な変更の追加などには、次の手順を使用します。

> **Note**
>
> クラスターの作成時に別の指定がない限り、クラスターはデフォルトのクラスターパラメータグループを使用して作成されます。`default` クラスターパラメータグループのパラメータは変更できません (`tls` の有効化/無効化など)。そのため、クラスターが `default` クラスターパラメータグループを使用している場合は、デフォルト以外のクラスターパラメータグループを使用するようにクラスターを変更する必要があります。まず、カスタムクラスターパラメータグループを作成する必要があります。詳細については、「Amazon DocumentDB クラスターパラメータグループを作成する」を参照してください。

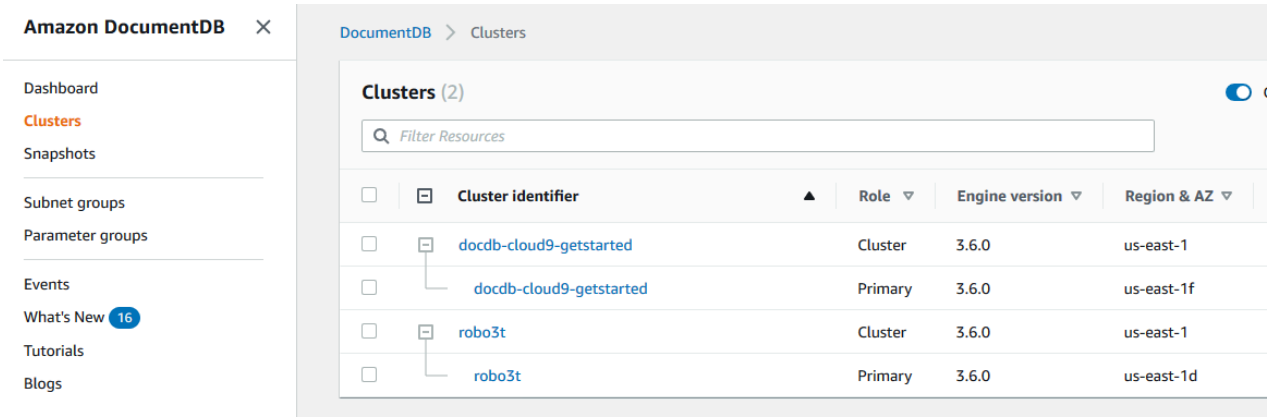
1. クラスターが使用しているクラスターパラメータグループを確認します。

   a. Amazon DocumentDB コンソールを次の場所で開きます。https://console.aws.amazon.com/docdb

   b. ナビゲーションペインで クラスター を選択します。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

   c. [クラスター] ナビゲーションボックスの [クラスター識別子] 列には、クラスターとインスタンスの両方が表示されますのでご注意ください。インスタンスはクラスターの下にリストされます。参考のために下のスクリーンショットを参照してください。



   d. 関心があるクラスターを選択します。

   e. [設定] タブを選択して、[クラスターの詳細] の一番下までスクロールし、[クラスターパラメータグループ] を見つけます。クラスターパラメータグループの名前を書き留めます。

      クラスターのパラメータグループの名前が `default` (たとえば、`default.docdb3.6` など) の場合、続行する前にカスタムのクラスターパラメータグループを作成して、それをクラスターのパラメータグループにする必要があります。詳細については次を参照してください:

      1. Amazon DocumentDB クラスターパラメータグループを作成する – 使用できるカスタムのクラスターパラメータグループがない場合、これを作成します。

2. Amazon DocumentDB クラスターの変更 - カスタムクラスターパラメータグループを使用するようにクラスターを変更します。

2. **tls** クラスターパラメータの現在の値を確認します。

   a. Amazon DocumentDB コンソールを次の場所で開きます。https://console.aws.amazon.com/docdb

   b. ナビゲーションペインで、[パラメータグループ] を選択します。

   c. クラスターパラメータグループの一覧から、使用するクラスターパラメータグループの名前を選択します。

   d. [クラスターパラメータ] セクションを見つけます。クラスターパラメータの一覧で、tls クラスターパラメータ行を見つけます。この時点では、次の 4 つの列が重要です。

      - クラスターパラメータ名 - クラスターパラメータの名前 TLS を管理するには、tls クラスターパラメータを選びます。
      - 値 - 各クラスターパラメータの現在の値
      - 使用できる値 - クラスターパラメータに適用できる値の一覧。
      - 適用タイプ - [静的] または [動的] のいずれか。静的クラスターパラメータへの変更は、インスタンスが再起動されるときにのみ適用されます。動的パラメータへの変更は、すぐに適用されるか、またはインスタンスが再起動されるときに適用されます。

3. **tls** クラスターパラメータの値を設定します。

   tls の値が必要な値ではない場合、このクラスターパラメータグループの値を変更します。tls クラスターパラメータの値を変更するには、前のセクションに続いて次の手順を行います。

   a. クラスターパラメータ名 (tls) の左側のボタンを選択します。

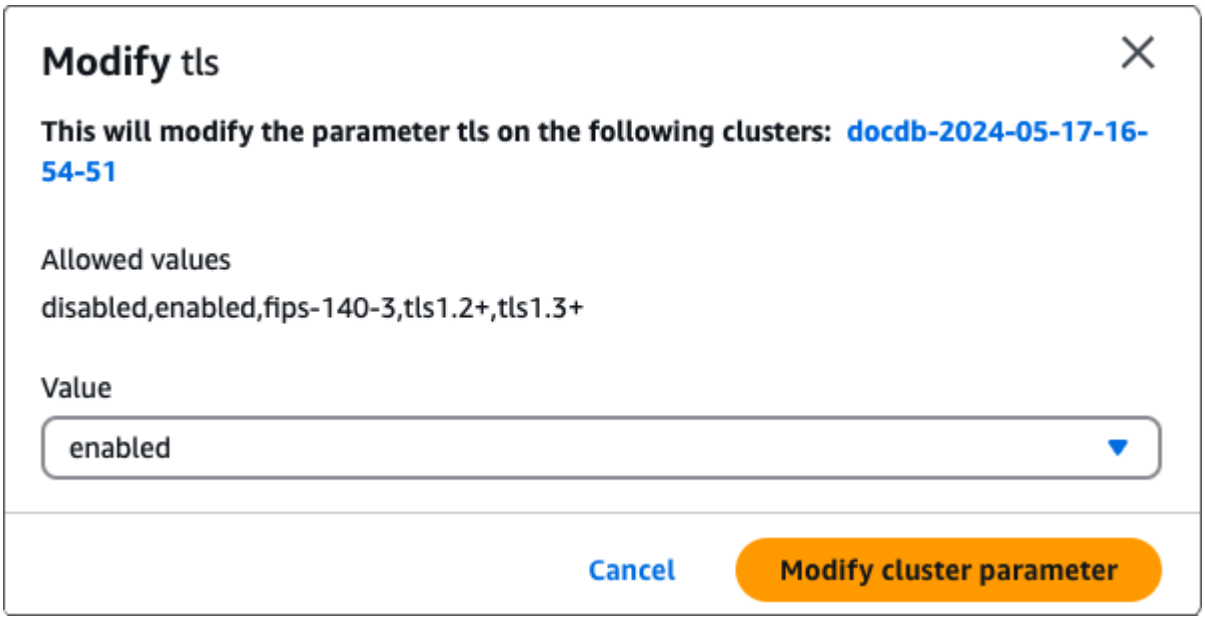
   b. [編集] を選択します。

   c. tls の値を変更するには、[**tls** の変更] ダイアログボックスで、ドロップダウンリストからクラスターパラメータに必要な値を選択します。

      次の値を指定できます:

      - disabled: TLS を無効にします
      - 有効 — TLS バージョン 1.0 から 1.3 を有効にします。

- fips-140-3: FIPS で TLS を有効にします。クラスターは、連邦情報処理標準 (FIPS) 出版物 140-3 の要件に従った安全な接続のみを受け付けます。これは、これらのリージョン (ca-central-1、us-west-2、us-east-1、us-east-2、us-gov-east-1、us-gov-west-1) で、Amazon DocumentDB 5.0 (エンジンバージョン 3.0.3727) 以降のクラスターでのみサポートされます。
- tls1.2+ — TLS バージョン 1.2 以降を有効にします。これは、Amazon DocumentDB 4.0 (エンジンバージョン 2.0.10980) および Amazon DocumentDB (エンジンバージョン 3.0.11051) 以降でのみサポートされています。
- tls1.3+ — TLS バージョン 1.3 以降を有効にします。これは、Amazon DocumentDB 4.0 (エンジンバージョン 2.0.10980) および Amazon DocumentDB (エンジンバージョン 3.0.11051) 以降でのみサポートされています。



d. [Modify cluster parameter (クラスターパラメータの変更)] を選択します。各クラスターインスタンスが再起動されると、変更が適用されます。

4. Amazon DocumentDB インスタンスを再起動します。

   変更がクラスター内のすべてのインスタンスに適用されるように、クラスターの各インスタンスを再起動します。

   a. Amazon DocumentDB コンソールを次の場所で開きます。https://console.aws.amazon.com/docdb

   b. ナビゲーションペインで、[インスタンス] を選択してください。

c. 再起動するインスタンスを指定するには、インスタンスのリストでインスタンスを見つけ、その名前の左側にあるボタンを選択します。

d. [アクション]、[再起動] の順に選択します。[再起動] を選択して、リブートすることを確認します。

Using the AWS CLI

AWS CLIを使用して TLS 向けにパフォーマンス管理タスクは、パラメータグループの識別、TLS 値の検証や必要な変更の追加などには、次の手順を使用します。

> **Note**
>
> クラスターの作成時に別の指定がない限り、クラスターはデフォルトのクラスターパラメータグループを使用して作成されます。default クラスターパラメータグループのパラメータは変更できません (tls の有効化/無効化など)。そのため、クラスターが default クラスターパラメータグループを使用している場合は、デフォルト以外のクラスターパラメータグループを使用するようにクラスターを変更する必要があります。まず、カスタムクラスターパラメータグループを作成する必要があります。詳細については、「Amazon DocumentDB クラスターパラメータグループを作成する」を参照してください。

1. クラスターが使用しているクラスターパラメータグループを確認します。

   次のオプションを指定して describe-db-clusters コマンドを実行します。

   - --db-cluster-identifier
   - --query

   次の例では、各###############をクラスターの情報に置き換えます。

```
aws docdb describe-db-clusters \
  --db-cluster-identifier mydocdbcluster \
  --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

   このオペレーションからの出力は次のようになります (JSON 形式) 。

```
[
    [
        "mydocdbcluster",
        "myparametergroup"
    ]
]
```

クラスターのパラメータグループの名前が default (たとえば、default.docdb3.6 など) の場合、続行する前にカスタムのクラスターパラメータグループを用意して、それをクラスターのパラメータグループにする必要があります。詳細については、以下の各トピックを参照してください。

1. Amazon DocumentDB クラスターパラメータグループを作成する – 使用できるカスタムのクラスターパラメータグループがない場合、これを作成します。
2. Amazon DocumentDB クラスターの変更 - カスタムクラスターパラメータグループを使用するようにクラスターを変更します。

2. **tls** クラスターパラメータの現在の値を確認します。

このクラスターパラメータグループの詳細については、次のオプションを指定して describe-db-cluster-parameters コマンドを実行します。

- --db-cluster-parameter-group-name
- --query

  出力は、、ParameterName、 ParameterValueAllowedValuesのフィールドのみに制限されますApplyType。

次の例では、各###############をクラスターの情報に置き換えます。

```
aws docdb describe-db-cluster-parameters \
  --db-cluster-parameter-group-name myparametergroup \
  --query 'Parameters[*].[ParameterName,ParameterValue,AllowedValues,ApplyType]'
```

このオペレーションからの出力は次のようになります (JSON 形式) 。

```
[
    [
        "audit_logs",
```

```
        "disabled",
        "enabled,disabled",
        "dynamic"
    ],
    [
        "tls",
        "disabled",
        "disabled,enabled,fips-140-3,tls1.2+,tls1.3+",
        "static"
    ],
    [
        "ttl_monitor",
        "enabled",
        "disabled,enabled",
        "dynamic"
    ]
]
```

3. **tls** クラスターパラメータの値を設定します。

   tls の値が必要な値ではない場合、このクラスターパラメータグループの値を変更します。tls クラスターパラメータの値を変更するには、次のオプションを指定して modify-db-cluster-parameter-group コマンドを実行します。

   - --db-cluster-parameter-group-name — 必須。変更するクラスターパラメータグループの名前。これを default.* クラスターパラメータグループにすることはできません。
   - --parameters — 必須。変更するクラスターパラメータグループのパラメーター一覧。
     - ParameterName — 必須。変更するクラスターパラメータの名前。
     - ParameterValue — 必須。このクラスターパラメータの新しい値。クラスターパラメータの AllowedValues のいずれかにする必要があります。
       - enabled — クラスターは TLS バージョン 1.0 から 1.3 を使用した安全な接続を受け入れます。
       - disabled - クラスターは TLS を使用したセキュアな接続を受け入れません。
       - fips-140-3 - クラスターは、連邦情報処理標準 (FIPS) 出版物 140-3 の要件に従った安全な接続のみを受け付けます。これは、これらのリージョン (ca-central-1、us-west-2、us-east-1、us-east-2、us-gov-east-1、us-gov-west-1) で、Amazon DocumentDB 5.0 (エンジンバージョン 3.0.3727) 以降のクラスターでのみサポートされます。

  - `tls1.2+` — クラスターは TLS バージョン 1.2 以降を使用した安全な接続を受け入れます。これは、Amazon DocumentDB 4.0 (エンジンバージョン 2.0.10980) および Amazon DocumentDB 5.0 (エンジンバージョン 3.0.11051) 以降でのみサポートされています。
  - `tls1.3+` — クラスターは TLS バージョン 1.3 以降を使用した安全な接続を受け入れます。これは、Amazon DocumentDB 4.0 (エンジンバージョン 2.0.10980) および Amazon DocumentDB 5.0 (エンジンバージョン 3.0.11051) 以降でのみサポートされています。
- `ApplyMethod` - この変更が適用される場合。`tle` のような静的クラスターパラメータでは、この値を `pending-reboot` にする必要があります。
  - `pending-reboot` - インスタンスの再起動後のみ、変更がインスタンスに適用されます。クラスターのすべてのインスタンスでこの変更を適用するには、各クラスターインスタンスを個別に再起動する必要があります。

次の例では、各*###############*をクラスターの情報に置き換えます。

次のコードは を無効にし`tls`、再起動時に各インスタンスに変更を適用します。

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-
reboot"
```

次のコードでは、再起動時に各インスタンスに変更を適用できます `tls` (バージョン 1.0 ~ 1.3)。

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=enabled,ApplyMethod=pending-
reboot"
```

次のコードは、 で TLS を有効にし`fips-140-3`、再起動時に各インスタンスに変更を適用します。

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup2 \
```

```
  --parameters "ParameterName=tls,ParameterValue=fips-140-3,ApplyMethod=pending-
reboot"
```

このオペレーションからの出力は次のようになります (JSON 形式) 。

```
{
    "DBClusterParameterGroupName": "myparametergroup"
}
```

4. お客様の Amazon DocumentDB インスタンスを再起動

変更がクラスター内のすべてのインスタンスに適用されるように、クラスターの各インスタンスを再起動します。Amazon DocumentDB インスタンスを再起動するには、次のオプションを指定して reboot-db-instance コマンドを実行します。

- `--db-instance-identifier`

次のコードは、インスタンス mydocdbinstance を再起動します。

次の例では、各###############をクラスターの情報に置き換えます。

Example

Linux、macOS、Unix の場合:

```
aws docdb reboot-db-instance \
  --db-instance-identifier mydocdbinstance
```

Windows の場合:

```
aws docdb reboot-db-instance ^
  --db-instance-identifier mydocdbinstance
```

このオペレーションからの出力は次のようになります (JSON 形式) 。

```
{
    "DBInstance": {
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "PreferredMaintenanceWindow": "fri:09:32-fri:10:02",
```

```
        "PendingModifiedValues": {},
        "DBInstanceStatus": "rebooting",
        "DBSubnetGroup": {
            "Subnets": [
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1a"
                    },
                    "SubnetIdentifier": "subnet-4e26d263"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1c"
                    },
                    "SubnetIdentifier": "subnet-afc329f4"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1e"
                    },
                    "SubnetIdentifier": "subnet-b3806e8f"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1d"
                    },
                    "SubnetIdentifier": "subnet-53ab3636"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1b"
                    },
                    "SubnetIdentifier": "subnet-991cb8d0"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1f"
                    },
```

```
                    "SubnetIdentifier": "subnet-29ab1025"
                }
            ],
            "SubnetGroupStatus": "Complete",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-91280df6",
            "DBSubnetGroupName": "default"
        },
        "PromotionTier": 2,
        "DBInstanceClass": "db.r5.4xlarge",
        "InstanceCreateTime": "2018-11-05T23:10:49.905Z",
        "PreferredBackupWindow": "00:00-00:30",
        "KmsKeyId": "arn:aws:kms:us-east-1:012345678901:key/0961325d-a50b-44d4-
b6a0-a177d5ff730b",
        "StorageEncrypted": true,
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-77186e0d"
            }
        ],
        "EngineVersion": "3.6.0",
        "DbiResourceId": "db-SAMPLERESOURCEID",
        "DBInstanceIdentifier": "mydocdbinstance",
        "Engine": "docdb",
        "AvailabilityZone": "us-east-1a",
        "DBInstanceArn": "arn:aws:rds:us-east-1:012345678901:db:sample-cluster-
instance-00",
        "BackupRetentionPeriod": 1,
        "Endpoint": {
            "Address": "mydocdbinstance.corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
            "Port": 27017,
            "HostedZoneId": "Z2R2ITUGPM61AM"
        },
        "DBClusterIdentifier": "mydocdbcluster"
    }
}
```

インスタンスが再起動するまでには数分かかります。インスタンスを使用できるのは、そのステータスが [available] である場合のみです。コンソールまたは AWS CLIを使用して、インスタンスのステータスをモニタリングできます。詳細については、「Amazon DocumentDB インスタンスのステータスのモニタリング」を参照してください。

キー管理

Amazon DocumentDB は AWS Key Management Service （ AWS KMS) を使用して暗号化キーを取得および管理します。安全で可用性の高いハードウェアとソフトウェア AWS KMS を組み合わせて、クラウド向けに拡張されたキー管理システムを提供します。を使用すると AWS KMS、暗号化キーを作成し、これらのキーの使用方法を制御するポリシーを定義できます。 は AWS KMS をサポートしているため AWS CloudTrail、キーの使用を監査して、キーが適切に使用されていることを確認できます。

AWS KMS キーは、Amazon DocumentDB、および Amazon Simple Storage Service (Amazon S3)、Amazon Relational Database Service (Amazon RDS)、Amazon Elastic Block Store (Amazon EBS)、Amazon Redshift などのサポートされている AWS サービスと組み合わせて使用できます。がサポートするサービスのリストについては AWS KMS、「 AWS Key Management Service デベロッパーガイド」の「 AWS サービスが を使用する方法 AWS KMS」を参照してください。詳細については AWS KMS、 「 とは」を参照してください AWS Key Management Service。

Amazon DocumentDB の Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインインを許可) し、誰に Amazon DocumentDB リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- Amazon DocumentDB と IAM との連携方法
- Amazon DocumentDB のアイデンティティベースのポリシー例
- Amazon DocumentDB ID とアクセスのトラブルシューティング
- Amazon DocumentDB リソースへのアクセス許可の管理
- Amazon DocumentDB 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
- AWS Amazon DocumentDB の マネージドポリシー

- Amazon DocumentDB API のアクセス許可: アクション、リソース、条件リファレンス

対象者

AWS Identity and Access Management (IAM) の使用方法は、Amazon DocumentDB で行う作業によって異なります。

サービスユーザー – ジョブを実行するために Amazon DocumentDB サービスを使用する場合は、管理者から必要なアクセス許可と認証情報が与えられます。さらに多くの Amazon DocumentDB の機能を使用して作業を行う場合は、追加のアクセス許可が必要になる場合があります。アクセスの管理方法を理解すると、管理者に適切なアクセス許可をリクエストするのに役に立ちます。Amazon DocumentDB の機能にアクセスできない場合は、「Amazon DocumentDB ID とアクセスのトラブルシューティング」を参照してください。

サービス管理者 – 会社で Amazon DocumentDB リソースを担当している場合は、通常、 Amazon DocumentDB へのフルアクセス権があります。サービスユーザーがどの Amazon DocumentDB 機能やリソースにアクセスするかを決めるのはサービス管理者の仕事です。決定後、IAM 管理者にリクエストを送信して、サービスユーザーのアクセス許可を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社が Amazon DocumentDB で IAM を使用する方法の詳細は、「Amazon DocumentDB と IAM との連携方法」をご参照ください。

IAM 管理者 - 管理者は、Amazon DocumentDB へのアクセス権を管理するポリシーの作成方法の詳細について確認する場合があります。IAM で使用できる Amazon DocumentDB のアイデンティティベースのポリシーの例は、「Amazon DocumentDB のアイデンティティベースのポリシー例」でご確認ください。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けることによって、認証 (にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center （ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook 認証情報は、フェデレーション ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーションを使用して にアクセスすると、間接的 AWS にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 AWS サインイン ユーザーガイド」の「 へのサインイン AWS アカウント方法」を参照してください。

AWS プログラムで にアクセスする場合、 はソフトウェア開発キット (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストを暗号化して署名します。AWS ツールを使用しない場合は、自分でリクエストに署名する必要があります。リクエストに自分で署名する推奨方法の使用については、「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。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。たとえば、 では、アカウントのセキュリティを高めるために多要素認証 (MFA) を使用する AWS ことをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「IAM ユーザーガイド」の「IAM のAWS 多要素認証」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、ID プロバイダーとのフェデレーションを使用して一時的な認証情報 AWS のサービス を使用して にアクセスすることを要求します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリ、または ID ソースを介して提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID がアクセスすると AWS アカウント、ロールを引き受け、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成するか、独自の ID ソースのユーザーとグループのセッ

トに接続して同期し、すべての AWS アカウント とアプリケーションで使用できます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「What is IAM Identity Center?」(IAM Identity Center とは) を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内の ID です。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、「IAM ユーザーガイド」の「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザー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、特定のアクセス許可 AWS アカウント を持つ 内の ID です。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。で IAM ロールを一時的に引き受けるには AWS Management Console、ユーザーから IAM ロール (コンソール) に切り替えることができます。ロールを引き受けるには、 または AWS API オペレーションを AWS CLI 呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロール

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 Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center User Guide」の「Permission sets」を参照してください。

- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、 (プロキシとしてロールを使用する代わりに) リソースに直接ポリシーをアタッチできます。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。
- クロスサービスアクセス — 一部の は他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用してアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする と組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除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 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。
  - サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカ

ウント 、サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。 AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 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。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「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、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、および Amazon VPC は AWS WAF、ACLs。ACL の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的でない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。Principal フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) の最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、ビジネスが所有する複数の AWS アカウント をグループ化して一元管理するためのサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー (SCP)」を参照してください。
- リソースコントロールポリシー (RCP) – RCP は、所有する各リソースにアタッチされた IAM ポリシーを更新することなく、アカウント内のリソースに利用可能な最大数のアクセス許可を設定するために使用できる JSON ポリシーです。RCP は、メンバーアカウントのリソースのアクセス許可を制限し、組織に属しているかどうかにかかわらず AWS アカウントのルートユーザー、 を含む ID の有効なアクセス許可に影響を与える可能性があります。RCP をサポートする のリストを含む Organizations と RCP の詳細については、AWS Organizations RCPs「リソースコントロールポリシー (RCPs」を参照してください。 AWS のサービス
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

Amazon DocumentDB と IAM との連携方法

IAM を使用して Amazon DocumentDB へのアクセスを管理する前に、Amazon DocumentDB で使用できる IAM 機能について理解しておく必要があります。

Amazon DocumentDB で使用できる IAM の機能

IAM の機能	インスタンスベースのクラスター	Elastic クラスター
アイデンティティベースポリシー	はい	はい
リソースベースのポリシー	いいえ	いいえ
ポリシーアクション	はい	はい
ポリシーリソース	はい	はい
ポリシー条件キー (サービス固有)	はい	はい
ACL	いいえ	いいえ
ABAC (ポリシー内のタグ)	部分的	はい
一時的な認証情報	はい	はい
プリンシパル権限	はい	はい
サービスロール	はい	はい
サービスリンクロール	いいえ	はい

Amazon DocumentDB およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要については、IAM ユーザーガイドのAWS 「IAM と連携する のサービス」を参照してください。

Amazon DocumentDB 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 ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 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。ID ベースの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。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。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。

Amazon DocumentDB のアイデンティティベースのポリシー例

Amazon DocumentDB のアイデンティティベースのポリシーの例は、「Amazon DocumentDB のアイデンティティベースのポリシー例」でご確認ください。

Amazon DocumentDB 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、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、アカウント全体、または別のアカウントの IAM エンティティをリソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も付与する必要があります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーをさらに付与する必

要はありません。詳細については、「IAM ユーザーガイド」の「IAM 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。

Amazon DocumentDB 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 JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは依存アクションと呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための許可を付与するポリシーで使用されます。

> **Note**
>
> 特定の管理機能では、Amazon DocumentDB は Amazon Relational Database Service (Amazon RDS) と共有の運用テクノロジーを使用します。
> RDS アクションの一覧については、「サービス認可リファレンス」の「Amazon Relational Database Service で定義されるアクション」を参照してください。
> Amazon DocumentDB エラスティッククラスターのポリシーアクションを表示するには、「サービス認可リファレンス」のAmazon DocumentDB エラスティッククラスターで定義されるアクション」を参照してください。

Amazon DocumentDB のポリシーアクションでは、アクションの前に以下のプレフィックスが使用されます。

```
aws
```

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": [
```

```
      "aws:action1",
      "aws:action2"
         ]
```

Amazon DocumentDB のアイデンティティベースのポリシーの例は、「Amazon DocumentDB のアイデンティティベースのポリシー例」でご確認ください。

Amazon DocumentDB 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 JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Resource` JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ステートメントには`Resource` または `NotResource` 要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの許可をサポートしないアクションの場合は、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": "*"
```

> **Note**
>
> 特定の管理機能では、Amazon DocumentDB は Amazon Relational Database Service (Amazon RDS) と共有の運用テクノロジーを使用します。
> RSD リソースのタイプと 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「Amazon Relational Database Service で定義されるリソース」を参照してください。各リソースの ARN を指定できるアクションについては、「Amazon Relational Database Service で定義されるアクション」を参照してください。
> Amazon DocumentDB Elastic クラスターのリソースタイプを確認するには、「サービス認可リファレンス」の「Amazon DocumentDB Elastic クラスターで定義されるリソースタイプ」を参照してください。

Amazon DocumentDB のアイデンティティベースのポリシーの例は、「Amazon DocumentDB のアイデンティティベースのポリシー例」でご確認ください。

Amazon DocumentDB 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 JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Condition` 要素 (または `Condition` ブロック) を使用すると、ステートメントが有効な条件を指定できます。`Condition` 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の `Condition` 要素を指定する場合、または 1 つの `Condition` 要素に複数のキーを指定する場合、 AWS では `AND` 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理`OR`オペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

> **Note**
>
> 特定の管理機能では、Amazon DocumentDB は Amazon Relational Database Service (Amazon RDS) と共有の運用テクノロジーを使用します。
> RDS 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「Amazon Relational Database Service の条件キー」を参照してください。条件キーを使用できるアクションとリソースについては、「Amazon Relational Database Service で定義するアクション」を参照してください。

Amazon DocumentDB Elastic クラスターの条件キーを確認するには、「サービス認可リファレンス」の「Amazon DocumentDB Elastic クラスターの条件キー」を参照してください。

Amazon DocumentDB のアイデンティティベースのポリシーの例は、「Amazon DocumentDB のアイデンティティベースのポリシー例」でご確認ください。

Amazon DocumentDB の ACL

ACL のサポート: なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon DocumentDB の ABAC

> **Note**
>
> ABAC はインスタンスベースのクラスターでは部分的にしかサポートされていませんが、Elastic クラスターではサポートされています。

属性ベースのアクセス制御 (ABAC) は、属性に基づいてアクセス許可を定義する認可戦略です。では AWS、これらの属性はタグと呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合にオペレーションを許可するように ABAC ポリシーをします。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

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。

Amazon DocumentDB で一時的な認証情報を使用する

一時的な認証情報のサポート: あり

一部の AWS のサービス は、一時的な認証情報を使用してサインインすると機能しません。一時的な認証情報 AWS のサービス を使用する場合などの詳細については、IAM ユーザーガイドAWS のサービス の「IAM と連携する 」を参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合、一時的な認証情報を使用します。たとえば、会社のシングルサインオン (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ユーザーから IAM ロールに切り替える (コンソール)」を参照してください。

一時的な認証情報は、 AWS CLI または AWS API を使用して手動で作成できます。その後、これらの一時的な認証情報を使用してアクセスすることができます AWS。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 AWS することをお勧めします。詳細については、「IAM の一時的セキュリティ認証情報」を参照してください。

Amazon DocumentDB 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

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする と組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

Amazon DocumentDB 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、Amazon DocumentDB の機能が破損する可能性があります。Amazon DocumentDB 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。

Amazon DocumentDB のサービスにリンクされたロール

> **Note**
>
> サービスにリンクされたロールはインスタンスベースのクラスターではサポートされていませんが、Elastic クラスターではサポートされています。

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービス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。

Amazon DocumentDB のアイデンティティベースのポリシー例

デフォルトでは、ユーザーおよびロールには Amazon DocumentDB リソースを作成または変更する許可はありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

これらのサンプル JSON ポリシードキュメントを使用して IAM ID 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。

Amazon DocumentDB 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「Amazon Relational Database Service 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。

> **Note**
>
> 特定の管理機能では、Amazon DocumentDB は Amazon Relational Database Service (Amazon RDS) と共有の運用テクノロジーを使用します。
> Amazon DocumentDB エラスティッククラスターのポリシーアクションについては、「サービス認可リファレンス」のAmazon DocumentDB エラスティッククラスターのアクション、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
- Amazon DocumentDB コンソールを使用する
- 自分の権限の表示をユーザーに許可する

ポリシーに関するベストプラクティス

アイデンティティベースのポリシーは、ユーザーのアカウント内で誰かが Amazon DocumentDB リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」または「ジョブ機能のAWS マネージドポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定

義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーとアクセス許可」を参照してください。

- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます AWS CloudFormation。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素:条件」を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer でポリシーを検証する」を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA を使用した安全な API アクセス」を参照してください。

IAM でのベストプラクティスの詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

Amazon DocumentDB コンソールを使用する

Amazon DocumentDB (MongoDB 互換) コンソールにアクセスするには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、 の Amazon DocumentDB リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な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) に対して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アクセス許可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が許可されます。

ユーザーとロールが引き続き Amazon DocumentDB 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 Amazon DocumentDB *ConsoleAccess*または *ReadOnly* AWS 管理ポリシーもアタッ

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 または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon DocumentDB ID とアクセスのトラブルシューティング

以下の情報を使用すると、Amazon DocumentDB および IAM での作業中に直面する可能性がある一般的な問題の診断や修正に役立ちます。

トピック

- Amazon DocumentDB でアクションを実行することを認可されていません
- iam:PassRole を実行する権限がありません
- 自分の 以外のユーザーに Amazon DocumentDB リソース AWS アカウント へのアクセスを許可したい

Amazon DocumentDB でアクションを実行することを認可されていません

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、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要なaws:*GetWidget* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 aws:GetWidget on resource: my-example-widget
```

この場合、aws:*GetWidget* アクションを使用して *my-example-widget*リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

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 Amazon DocumentDB にロールを渡せるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、既存のロールをそのサービスに渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、 marymajor という名前の IAM ユーザーがコンソールを使用して Amazon DocumentDB でアクションを実行しようとする際に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに Amazon DocumentDB リソース AWS アカウント 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:

- Amazon DocumentDB がこれらの機能をサポートしているかどうかは「Amazon DocumentDB と IAM との連携方法」を参照してください。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、「IAM ユーザーガイド」の「外部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。

Amazon DocumentDB リソースへのアクセス許可の管理

すべての AWS リソースは によって所有され AWS アカウント、リソースを作成またはアクセスするアクセス許可はアクセス許可ポリシーによって管理されます。アカウント管理者は、アクセス許可ポリシーを IAM ID (ユーザー、グループ、ロール) にアタッチできます。一部のサービス (など AWS Lambda) 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、管理者アクセス許可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。

トピック

- Amazon DocumentDB 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
- リソースへのアクセスの管理
- 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

Amazon DocumentDB リソースおよびオペレーション

Amazon DocumentDB では、プライマリリソースは クラスター です。Amazon DocumentDB は、インスタンス、パラメータグループ、イベントサブスクリプション などのプライマリリソースで使用できる他のリソースをサポートします。これらのリソースはサブリソースと呼ばれます。

これらのリソースとサブリソースには、次の表に示すとおり、一意の Amazon リソースネーム (ARN) が関連付けられています。

リソースタイプ	ARN 形式
クラスター	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster:` *`db-cluster-name`*
クラスターパラメータグループ	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster-pg:` *`cluster-parameter-group-name`*

リソースタイプ	ARN 形式
クラススナップショット	arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*
インスタンス	arn:aws:rds: *region*:*account-id* :db:*db-instance-name*
セキュリティグループ	arn:aws:rds: *region*:*account-id* :secgrp:*security-group-name*
サブネットグループ	arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*

Amazon DocumentDB には、Amazon DocumentDB のリソースを操作するための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、「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

リソース所有者は、リソースを AWS アカウント 作成した です。つまり、リソース所有者は、リソースを作成するリクエスト AWS アカウント を認証するプリンシパルエンティティ (ルートアカウント、IAM ユーザー、または IAM ロール) の です。次の例は、この仕組みを示しています。

- のルートアカウントの認証情報を使用してインスタンスなどの Amazon DocumentDB リソース AWS アカウント を作成する場合、 AWS アカウント は Amazon DocumentDB リソースの所有者です。
- で IAM ユーザーを作成し AWS アカウント 、そのユーザーに Amazon DocumentDB リソースを作成するアクセス許可を付与すると、そのユーザーは Amazon DocumentDB リソースを作成できます。ただし、ユーザーが属 AWS アカウントする は Amazon DocumentDB リソースを所有しています。
- Amazon DocumentDB リソースを作成するアクセス許可 AWS アカウント を持つ で IAM ロールを作成する場合、ロールを引き受けることができるすべてのユーザーが Amazon DocumentDB リソースを作成できます。ロールが属 AWS アカウントする は、Amazon DocumentDB 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、Amazon DocumentDB 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 ポリシーの構文の詳細と説明については、「IAM ユーザーガイド」の「AWSIAM IAM ポリシーリファレンス」を参照してください。

IAM ID にアタッチされているポリシーは、ID ベース のポリシー (IAM ポリシー) と呼ばれます。リソースにアタッチされているポリシーは、リソースベース のポリシーと呼ばれます。Amazon DocumentDB では、アイデンティティベースのポリシー (IAM ポリシー) のみをサポートしています。

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)
- リソースベースのポリシー

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付与する - アカウント管理者は、特定のユーザーに関連付けられているアクセス許可ポリシーを使用して、そのユーザーにインスタンスなどの Amazon DocumentDB リソースを作成するためのアクセス許可を付与できます。
- アクセス権限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 ベースのアクセス権限ポリシーを IAM ロールにアタッチして、クロスアカウントの権限を付与することができます。たとえば、管理者は次のように、別の AWS アカウント 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 のリソースに許可を付与するロールに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、アカウント B のユーザーにロールを引き受けるアクセス許可を委任できます。これにより、アカウント B のユーザーはアカウント A のリソースを作成またはアクセスできます。ロールを引き受けるアクセス許可を AWS サービスに付与する場合、信頼ポリシーのプリンシパルは AWS サービスプリンシパルになることもできます。

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。

以下は、ID 123456789012 を持つユーザーが AWS アカウントのインスタンスを作成できるようにするポリシーの例です。新しいインスタンスでは、オプショングループと `default` で始まるパラメータグループ、および `default` サブネットグループを使用する必要があります。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateDBInstanceOnly",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBInstance"
            ],
            "Resource": [
                "arn:aws:rds:*:123456789012:db:test*",
                "arn:aws:rds:*:123456789012:pg:cluster-pg:default*",
                "arn:aws:rds:*:123456789012:subgrp:default"
            ]
        }
    ]
}
```

Amazon DocumentDB 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、Amazon DocumentDB 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、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。

リソースベースのポリシー

Amazon S3 (Amazon Simple Storage Service) などその他のサービスでは、リソースベースの権限ポリシーがサポートされています。例えば、ポリシーを Amazon S3 バケットにアタッチして、そのバケットに対するアクセス許可を管理できます。Amazon DocumentDB では、リソースベースのポリシーはサポートされていません。

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

Amazon DocumentDB リソースごとに (Amazon DocumentDB リソースおよびオペレーション を参照)、そのサービスは、API オペレーションの設定を定義します。詳細については、「アクション」を参照してください。これらのAPIオペレーションの権限を付与するために、Amazon DocumentDB はポリシーで指定できる一連のアクションを定義します。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。

以下は、基本的なポリシーの要素です。

- 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。
- [Action] (アクション) - アクションのキーワードを使用して、許可または拒否するリソースオペレーションを識別します。例えば、`rds:DescribeDBInstances` 権限は、`DescribeDBInstances` オペレーションの実行をユーザーに許可します。
- 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。また、明示的にリソースへのアクセスを拒否すると、別のポリシーによってアクセスが許可されている場合でも、ユーザーはそのリソースにアクセスできなくなります。
- 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、権限(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。Amazon DocumentDB では、リソースベースのポリシーはサポートされていません。

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 Amazon DocumentDB API アクションとそれらが適用されるリソースを示す表については、Amazon DocumentDB API のアクセス許可: 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。

条件を表すには、あらかじめ定義された条件キーを使用します。Amazon DocumentDB には、IAM ポリシーで使用できるサービス固有のコンテキストキーはありません。すべてのサービスで使用できるグローバル条件コンテキストキーのリストについては、IAM ユーザーガイド の条件に使用可能なキー を参照してください。

Amazon DocumentDB 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

> **Important**
>
> 特定の管理機能について、Amazon DocumentDB は Amazon RDS と共有されるオペレーショナルテクノロジーを使用します。Amazon DocumentDB コンソール、、および API コールは AWS CLI、Amazon RDS API に対する呼び出しとして ログに記録されます。
> 最初に、Amazon DocumentDB リソースへのアクセスを管理するために利用できる基本的な概念とオプションを説明する紹介トピックを確認することをお勧めします。詳細については、「Amazon DocumentDB リソースへのアクセス許可の管理」を参照してください。

このトピックでは、アカウント管理者が IAM アイデンティティ(ユーザー、グループ、ロール)へのアクセス権限ポリシーをアタッチする、アイデンティティベースのポリシーの例を示します。

IAM ポリシーの例を次に示します。

JSON

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

```
    "Statement": [
        {
            "Sid": "AllowCreateDBInstanceOnly",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBInstance"
            ],
            "Resource": [
                "arn:aws:rds:*:123456789012:db:test*",
                "arn:aws:rds:*:123456789012:pg:cluster-pg:default*",
                "arn:aws:rds:*:123456789012:subgrp:default"
            ]
        }
    ]
}
```

ポリシーには、IAM ユーザー用の以下のアクセス許可を指定する単一のステートメントが含まれます。

- このポリシーにより、IAM ユーザーは CreateDBInstance アクションを使用してインスタンスを作成できます (これは create-db-instance AWS CLI オペレーションと にも適用されます AWS Management Console)。
- Resource 要素では、ユーザーがリソースでアクションを実行できることを指定できます。Amazon リソースネーム (ARN) を使用してリソースを指定します。この ARN には、リソースが属するサービスの名前 (rds)、 AWS リージョン (* はこの例の任意のリージョンを示します)、ユーザーアカウント番号 (123456789012 はこの例のユーザー ID です)、およびリソースのタイプが含まれます。

  例の Resource 要素は、ユーザーのリソースで、以下のポリシーの制約を指定します。

  - 新しいインスタンスのインスタンス識別子は、test で始まる必要があります (例: testCustomerData1、test-region2-data)。
  - 新しいインスタンスのクラスターパラメータグループは、default で始まる必要があります。
  - 新しいインスタンスのサブネットグループは、default サブネットグループである必要があります。

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 Amazon DocumentDB API オペレーションとそれらが適用されるリソースを示す表については、Amazon DocumentDB API のアクセス許可: 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。

Amazon DocumentDB コンソールの使用に必要な許可

Amazon DocumentDB コンソールを使用するユーザーには、最小限の権限のセットが必要です。これらのアクセス許可により、ユーザーは の Amazon DocumentDB リソースを記述 AWS アカウントし、Amazon EC2 セキュリティやネットワーク情報などの他の関連情報を提供できます。

これらの最小限必要なアクセス許可よりも制限された IAM ポリシーを作成している場合、その IAM ポリシーを使用するユーザーに対してコンソールは意図したとおりには機能しません。これらのユーザーが引き続き Amazon DocumentDB コンソールを使用できるようにするには、`AmazonDocDBConsoleFullAccess` で説明されているように AWS Amazon DocumentDB のマネージドポリシー マネージドポリシーをユーザーにアタッチします。

AWS CLI または Amazon DocumentDB API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。

カスタマーマネージドポリシーの例

このセクションでは、さまざまな Amazon DocumentDB アクションの許可を付与するユーザーポリシー例を示しています。これらのポリシーは、Amazon DocumentDB API アクション、 AWS SDKs、または を使用している場合に機能します AWS CLI。コンソールを使用している場合は、「Amazon DocumentDB コンソールの使用に必要な許可」で説明しているコンソールに固有の追加のアクセス権限を付与する必要があります。

特定の管理機能について、Amazon DocumentDB は、Amazon RDS (Amazon Relational Database Service) および AmazonNeptune と共有される運用テクノロジーを使用します。

> **Note**
>
> すべての例で、米国東部 (バージニア北部) リージョン (`us-east-1`) を使用し、架空のアカウント ID を使用しています。

例

- 例 1: ユーザーが任意の Amazon DocumentDB リソースに対して任意の Describe アクションを実行できるようにする
- 例 2: ユーザーによるインスタンスの削除を禁止する
- 例 3: ストレージ暗号化が有効になっている場合を除き、ユーザーがクラスターを作成できないようにする

例 1: ユーザーが任意の Amazon DocumentDB リソースに対して任意の Describe アクションを実行できるようにする

次のアクセス権限ポリシーは、Describe で始まるすべての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。これらのアクションは、インスタンスなど Amazon DocumentDB リソースに関する情報を表示します。Resource 要素内のワイルドカード文字 (*) は、アカウントによって所有されるすべての Amazon DocumentDB リソースに対してそれらのアクションが許可されることを示します。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowRDSDescribe",
         "Effect":"Allow",
         "Action":"rds:Describe*",
         "Resource":"*"
      }
   ]
}
```

例 2: ユーザーによるインスタンスの削除を禁止する

以下のアクセス権限ポリシーは、特定のインスタンスを削除することをユーザーに禁止するためのアクセス権限を付与します。例えば、管理者ではないすべてのユーザーに対して、本稼働インスタンスの削除を拒否することもできます。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyDelete1",
         "Effect":"Deny",
         "Action":"rds:DeleteDBInstance",
         "Resource":"arn:aws:rds:us-east-1:123456789012:db:my-db-instance"
      }
   ]
}
```

例 3: ストレージ暗号化が有効になっている場合を除き、ユーザーがクラスターを作成できないようにする

次のアクセス許可ポリシーは、ストレージの暗号化が有効になっていない限り、ユーザーが Amazon DocumentDB クラスターを作成することを許可しません。

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "PreventUnencryptedDocumentDB",
         "Effect": "Deny",
         "Action": "RDS:CreateDBCluster",
         "Condition": {
         "Bool": {
         "rds:StorageEncrypted": "false"
      },
         "StringEquals": {
         "rds:DatabaseEngine": "docdb"
         }
      },
      "Resource": "*"
      }
   ]
```

```
}
```

AWS Amazon DocumentDB の 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、自分でポリシーを記述するよりも AWS 管理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。すぐに開始するには、 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としており、 AWS アカウントで利用できます。 AWS 管理ポリシーの詳細については、AWS 「 Identity and Access Management ユーザーガイド」のAWS 「 管理ポリシー」を参照してください。

AWS サービスは、 AWS 管理ポリシーを維持および更新します。 AWS 管理ポリシーのアクセス許可は変更できません。サービスは、新機能をサポートするために、 AWS 管理ポリシーに追加のアクセス許可を追加することがあります。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。サービスは、新機能が起動されたとき、または新しいオペレーションが利用可能になったときに、 AWS マネージドポリシーを更新する可能性が最も高いです。サービスは AWS マネージドポリシーからアクセス許可を削除しないため、ポリシーの更新によって既存のアクセス許可が損なわれることはありません。

さらに、 は、複数のサービスにまたがるジョブ関数の 管理ポリシー AWS をサポートしています。たとえば、 `ViewOnlyAccess` AWS マネージドポリシーは、多くの AWS 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。サービスが新機能を起動すると、 は新しいオペレーションとリソースの読み取り専用アクセス許可 AWS を追加します。職務機能ポリシーのリストと説明については、「AWS Identity and Access Management ユーザーガイド」の「 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。

アカウントのユーザーにアタッチできる以下の AWS 管理ポリシーは、Amazon DocumentDB に固有のものです。

- AmazonDocDBFullAccess – ルート AWS アカウントのすべての Amazon DocumentDB リソースへのフルアクセスを許可します。
- AmazonDocDBReadOnlyAccess – ルート AWS アカウントのすべての Amazon DocumentDB リソースへの読み取り専用アクセスを許可します。
- AmazonDocDBConsoleFullAccess - AWS Management Consoleを使用して、Amazon DocumentDB および Amazon DocumentDB Elastic クラスターリソースを管理するためのフルアクセスを許可します。

- AmazonDocDBElasticReadOnlyAccess – ルート AWS アカウントのすべての Amazon DocumentDB Elastic クラスターリソースへの読み取り専用アクセスを許可します。
- AmazonDocDBElasticFullAccess – ルート AWS アカウントのすべての Amazon DocumentDB Elastic クラスターリソースへのフルアクセスを許可します。

AmazonDocDBFullAccess

このポリシーは、プリンシパルにすべての Amazon DocumentDB アクションへのフルアクセスを許可する管理者権限を付与します。このポリシーの権限は、次のようにグループ化されています。

- Amazon DocumentDB の権限は、Amazon DocumentDB アクションをすべて許可します。
- このポリシーの Amazon EC2 権限の一部は、API リクエストで渡されたリソースを検証するために必要です。これは、Amazon DocumentDB がクラスターでリソースを正常に使用できるようにするためです。このポリシーの残りの Amazon EC2 アクセス許可により、Amazon DocumentDB はクラスターへの接続を可能にするために必要な AWS リソースを作成できます。
- Amazon DocumentDB 権限は、リクエストで渡されたリソースを検証するために API コール中に使用されます。これらは、Amazon DocumentDB が、渡されたキーを Amazon DocumentDB クラスターで使用できるようにするために必要です。
- CloudWatch Logsは、Amazon DocumentDB がログ配信先に到達可能であり、ブローカーログの使用に有効であることを確認するために必要です。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "rds:AddRoleToDBCluster",
                "rds:AddSourceIdentifierToSubscription",
                "rds:AddTagsToResource",
                "rds:ApplyPendingMaintenanceAction",
                "rds:CopyDBClusterParameterGroup",
                "rds:CopyDBClusterSnapshot",
                "rds:CopyDBParameterGroup",
                "rds:CreateDBCluster",
                "rds:CreateDBClusterParameterGroup",
                "rds:CreateDBClusterSnapshot",
```

```
                "rds:CreateDBInstance",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSubnetGroup",
                "rds:CreateEventSubscription",
                "rds:DeleteDBCluster",
                "rds:DeleteDBClusterParameterGroup",
                "rds:DeleteDBClusterSnapshot",
                "rds:DeleteDBInstance",
                "rds:DeleteDBParameterGroup",
                "rds:DeleteDBSubnetGroup",
                "rds:DeleteEventSubscription",
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEngineDefaultClusterParameters",
                "rds:DescribeEngineDefaultParameters",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeOptionGroups",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DescribeValidDBInstanceModifications",
                "rds:DownloadDBLogFilePortion",
                "rds:FailoverDBCluster",
                "rds:ListTagsForResource",
                "rds:ModifyDBCluster",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyDBSubnetGroup",
                "rds:ModifyEventSubscription",
```

```
                "rds:PromoteReadReplicaDBCluster",
                "rds:RebootDBInstance",
                "rds:RemoveRoleFromDBCluster",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:RemoveTagsFromResource",
                "rds:ResetDBClusterParameterGroup",
                "rds:ResetDBParameterGroup",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:RestoreDBClusterToPointInTime"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs",
                "kms:ListAliases",
                "kms:ListKeyPolicies",
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "sns:ListSubscriptions",
                "sns:ListTopics",
                "sns:Publish"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Action": "iam:CreateServiceLinkedRole",
            "Effect": "Allow",
```

```
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
                    "iam:AWS ServiceName": "rds.amazonaws.com"
                }
            }
        }
    ]
}
```

AmazonDocDBReadOnlyAccess

このポリシーは、ユーザーが Amazon DocumentDB で情報を表示できるようにする読み取り専用の権限を付与します。このポリシーが添付されているプリンシパルは、既存のリソースを更新または削除したり、新しい Amazon DocumentDB リソースを作成したりすることはできません。例えば、これらの権限を持つプリンシパルは、自分のアカウントに関連付けられているクラスターと構成のリストを表示できますが、クラスターの構成や設定を変更することはできません。このポリシーの権限は、次のようにグループ化されています。

- Amazon DocumentDB 権限を使用すると、Amazon DocumentDB リソースを一覧表示し、それらを記述し、それらに関する情報を取得できます。
- Amazon EC2 権限は、クラスターに関連付けられている Amazon VPC、サブネット、セキュリティグループ、および ENI を記述するために使用されます。
- Amazon DocumentDB 権限は、クラスターに関連付けられているキーを記述するために使用されます。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
```

```
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DownloadDBLogFilePortion",
                "rds:ListTagsForResource"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "kms:ListKeys",
```

```
                "kms:ListRetirableGrants",
                "kms:ListAliases",
                "kms:ListKeyPolicies"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "logs:DescribeLogStreams",
                "logs:GetLogEvents"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*",
                "arn:aws:logs:*:*:log-group:/aws/docdb/*:log-stream:*"
            ]
        }
    ]
}
```

AmazonDocDBConsoleFullAccess

以下 AWS Management Console に対して を使用して Amazon DocumentDB リソースを管理するためのフルアクセスを付与します。

- すべての Amazon DocumentDB アクションと Amazon DocumentDB クラスターアクションを許可する Amazon DocumentDB のアクセス許可。
- このポリシーの Amazon EC2 権限の一部は、API リクエストで渡されたリソースを検証するために必要です。これは、Amazon DocumentDB がリソースを正常に使用して、クラスターをプロビジョンおよび維持できるようにするためです。このポリシーの残りの Amazon EC2 アクセス許可により、Amazon DocumentDB は VPCEndpoint などのクラスターに接続するために必要な AWS リソースを作成できます。
- AWS KMS アクセス許可は、 への API コール中に使用され AWS KMS 、リクエストで渡されたリソースを検証します。これらは、Amazon DocumentDB が、渡されたキーを使用して、Amazon DocumentDB エラスティッククラスターで保存されている保管中のデータを暗号化および復号化できるようにするために必要です。
- CloudWatch Logs は、Amazon DocumentDB が口グ配信先に到達可能であり、口グ使用の監査およびプロファイリングに有効であることを確認するために必要です。

- 特定のシークレットを検証し、それを使用して Amazon DocumentDB Elastic クラスターの管理者ユーザーをセットアップするには、Secrets Manager 権限が必要です。
- Amazon DocumentDB クラスター管理アクションには Amazon RDS アクセス権限が必要です。特定の管理機能について、Amazon DocumentDB は Amazon RDS と共有されるオペレーショナルテクノロジーを使用します。
- SNS 権限は、プリンシパルが Amazon Simple Notification Service (Amazon SNS) のサブスクリプションとトピックにアクセスして、Amazon SNS メッセージを発行することを許可します。
- メトリクスとログの公開に必要なサービスにリンクされたロールを作成するには IAM 権限が必要です。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DocdbSids",
            "Effect": "Allow",
            "Action": [
                "docdb-elastic:CreateCluster",
                "docdb-elastic:UpdateCluster",
                "docdb-elastic:GetCluster",
                "docdb-elastic:DeleteCluster",
                "docdb-elastic:ListClusters",
                "docdb-elastic:CreateClusterSnapshot",
                "docdb-elastic:GetClusterSnapshot",
                "docdb-elastic:DeleteClusterSnapshot",
                "docdb-elastic:ListClusterSnapshots",
                "docdb-elastic:RestoreClusterFromSnapshot",
                "docdb-elastic:TagResource",
                "docdb-elastic:UntagResource",
                "docdb-elastic:ListTagsForResource",
                "docdb-elastic:CopyClusterSnapshot",
                "docdb-elastic:StartCluster",
                "docdb-elastic:StopCluster",
                "docdb-elastic:GetPendingMaintenanceAction",
                "docdb-elastic:ListPendingMaintenanceActions",
                "docdb-elastic:ApplyPendingMaintenanceAction",
                "rds:AddRoleToDBCluster",
                "rds:AddSourceIdentifierToSubscription",
```

```
                "rds:AddTagsToResource",
                "rds:ApplyPendingMaintenanceAction",
                "rds:CopyDBClusterParameterGroup",
                "rds:CopyDBClusterSnapshot",
                "rds:CopyDBParameterGroup",
                "rds:CreateDBCluster",
                "rds:CreateDBClusterParameterGroup",
                "rds:CreateDBClusterSnapshot",
                "rds:CreateDBInstance",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSubnetGroup",
                "rds:CreateEventSubscription",
                "rds:CreateGlobalCluster",
                "rds:DeleteDBCluster",
                "rds:DeleteDBClusterParameterGroup",
                "rds:DeleteDBClusterSnapshot",
                "rds:DeleteDBInstance",
                "rds:DeleteDBParameterGroup",
                "rds:DeleteDBSubnetGroup",
                "rds:DeleteEventSubscription",
                "rds:DeleteGlobalCluster",
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEngineDefaultClusterParameters",
                "rds:DescribeEngineDefaultParameters",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOptionGroups",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
```

```
                "rds:DescribeValidDBInstanceModifications",
                "rds:DownloadDBLogFilePortion",
                "rds:FailoverDBCluster",
                "rds:ListTagsForResource",
                "rds:ModifyDBCluster",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyDBSubnetGroup",
                "rds:ModifyEventSubscription",
                "rds:ModifyGlobalCluster",
                "rds:PromoteReadReplicaDBCluster",
                "rds:RebootDBInstance",
                "rds:RemoveFromGlobalCluster",
                "rds:RemoveRoleFromDBCluster",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:RemoveTagsFromResource",
                "rds:ResetDBClusterParameterGroup",
                "rds:ResetDBParameterGroup",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:RestoreDBClusterToPointInTime"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "DependencySids",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "ec2:AllocateAddress",
                "ec2:AssignIpv6Addresses",
                "ec2:AssignPrivateIpAddresses",
                "ec2:AssociateAddress",
                "ec2:AssociateRouteTable",
                "ec2:AssociateSubnetCidrBlock",
                "ec2:AssociateVpcCidrBlock",
                "ec2:AttachInternetGateway",
                "ec2:AttachNetworkInterface",
```

```
                "ec2:CreateCustomerGateway",
                "ec2:CreateDefaultSubnet",
                "ec2:CreateDefaultVpc",
                "ec2:CreateInternetGateway",
                "ec2:CreateNatGateway",
                "ec2:CreateNetworkInterface",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:CreateSecurityGroup",
                "ec2:CreateSubnet",
                "ec2:CreateVpc",
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddresses",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeCustomerGateways",
                "ec2:DescribeInstances",
                "ec2:DescribeNatGateways",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribePrefixLists",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroupReferences",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcs",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:ModifySubnetAttribute",
                "ec2:ModifyVpcAttribute",
                "ec2:ModifyVpcEndpoint",
                "kms:DescribeKey",
                "kms:ListAliases",
                "kms:ListKeyPolicies",
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "sns:ListSubscriptions",
                "sns:ListTopics",
                "sns:Publish"
            ],
            "Resource": [
                "*"
```

```
            ]
        },
        {
            "Sid": "DocdbSLRSid",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "rds.amazonaws.com"
                }
            }
        },
        {
            "Sid": "DocdbElasticSLRSid",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/docdb-
elastic.amazonaws.com/AWSServiceRoleForDocDB-Elastic",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "docdb-elastic.amazonaws.com"
                }
            }
        }
    ]
}
```

AmazonDocDBElasticReadOnlyAccess

このポリシーは、ユーザーが Amazon DocumentDB で Elastic クラスター情報を表示できるようにする読み取り専用の権限を付与します。このポリシーが添付されているプリンシパルは、既存のリソースを更新または削除したり、新しい Amazon DocumentDB リソースを作成したりすることはできません。例えば、これらの権限を持つプリンシパルは、自分のアカウントに関連付けられているクラスターと構成のリストを表示できますが、クラスターの構成や設定を変更することはできません。このポリシーの権限は、次のようにグループ化されています。

- Amazon DocumentDB Elastic クラスター権限を使用すると、Amazon DocumentDB Elastic クラスターリソースを一覧表示し、それらを記述し、それらに関する情報を取得できます。
- CloudWatch 権限は、サービスメトリクスを検証するために使用します。

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "docdb-elastic:ListClusters",
            "docdb-elastic:GetCluster",
            "docdb-elastic:ListClusterSnapshots",
            "docdb-elastic:GetClusterSnapshot",
            "docdb-elastic:ListTagsForResource"
         ],
         "Resource": "*"
      },
      {
         "Effect": "Allow",
         "Action": [
            "cloudwatch:GetMetricData",
            "cloudwatch:ListMetrics",
            "cloudwatch:GetMetricStatistics"
         ],
         "Resource": "*"
      }
   ]
}
```

AmazonDocDBElasticFullAccess

このポリシーは、Amazon DocumentDB Elastic クラスターに関するすべての Amazon DocumentDB アクションへのフルアクセスをプリンシパルに許可する管理者権限を付与します。

このポリシーは、条件内で AWS タグ (https://docs.aws.amazon.com/tag-editor/latest/userguide/tagging.html) を使用して、リソースへのアクセスの範囲を設定します。シークレットを使用する場合は、タグキー `DocDBElasticFullAccess` とタグ値でタグ付けする必要があります。カスタマー管理キーを使用する場合は、タグキー `DocDBElasticFullAccess` とタグ値でタグ付けする必要があります。

このポリシーの権限は、次のようにグループ化されています。

- Amazon DocumentDB Elastic クラスターアクセス権限では、すべての Amazon DocumentDB アクションが許可されます。
- このポリシーの Amazon EC2 権限の一部は、API リクエストで渡されたリソースを検証するために必要です。これは、Amazon DocumentDB がリソースを正常に使用して、クラスターをプロビジョンおよび維持できるようにするためです。このポリシーの残りの Amazon EC2 アクセス許可により、Amazon DocumentDB は VPC エンドポイントなどのクラスターに接続するために必要な AWS リソースを作成できます。
- AWS KMS Amazon DocumentDB が渡されたキーを使用して Amazon DocumentDB エラスティッククラスター内の保管中のデータを暗号化および復号化できるようにするには、 アクセス許可が必要です。

> **Note**
>
> カスタマー管理キーには、キー `DocDBElasticFullAccess` が付いたタグとタグ値が必要です。

- 特定のシークレットを検証し、それを使用して Amazon DocumentDB Elastic クラスターの管理者ユーザーをセットアップするには、SecretsManager の権限が必要です。

> **Note**
>
> 使用するシークレットには、キー `DocDBElasticFullAccess` が付いたタグとタグ値が必要です。

- メトリクスとログの公開に必要なサービスにリンクされたロールを作成するには IAM 権限が必要です。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DocdbElasticSid",
            "Effect": "Allow",
            "Action": [
                "docdb-elastic:CreateCluster",
                "docdb-elastic:UpdateCluster",
```

```
                "docdb-elastic:GetCluster",
                "docdb-elastic:DeleteCluster",
                "docdb-elastic:ListClusters",
                "docdb-elastic:CreateClusterSnapshot",
                "docdb-elastic:GetClusterSnapshot",
                "docdb-elastic:DeleteClusterSnapshot",
                "docdb-elastic:ListClusterSnapshots",
                "docdb-elastic:RestoreClusterFromSnapshot",
                "docdb-elastic:TagResource",
                "docdb-elastic:UntagResource",
                "docdb-elastic:ListTagsForResource",
                "docdb-elastic:CopyClusterSnapshot",
                "docdb-elastic:StartCluster",
                "docdb-elastic:StopCluster",
                "docdb-elastic:GetPendingMaintenanceAction",
                "docdb-elastic:ListPendingMaintenanceActions",
                "docdb-elastic:ApplyPendingMaintenanceAction"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "EC2Sid",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeVpcEndpoints",
                "ec2:DeleteVpcEndpoints",
                "ec2:ModifyVpcEndpoint",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeAvailabilityZones",
                "secretsmanager:ListSecrets"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "docdb-elastic.amazonaws.com"
                }
```

```
            }
        },
        {
            "Sid": "KMSSid",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "docdb-elastic.*.amazonaws.com"
                    ],
                    "aws:ResourceTag/DocDBElasticFullAccess": "*"
                }
            }
        },
        {
            "Sid": "KMSGrantSid",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/DocDBElasticFullAccess": "*",
                    "kms:ViaService": [
                        "docdb-elastic.*.amazonaws.com"
                    ]
                },
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        },
        {
            "Sid": "SecretManagerSid",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:ListSecretVersionIds",
```

```
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:GetResourcePolicy"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "secretsmanager:ResourceTag/DocDBElasticFullAccess": "*"
                },
                "StringEquals": {
                    "aws:CalledViaFirst": "docdb-elastic.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudwatchSid",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "SLRSid",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/docdb-
elastic.amazonaws.com/AWSServiceRoleForDocDB-Elastic",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "docdb-elastic.amazonaws.com"
                }
            }
        }
    ]
}
```

AmazonDocDB-ElasticServiceRolePolicy

AWS Identity and Access Management エンティティ`AmazonDocDBElasticServiceRolePolicy`に をアタッチすることはできません。このポリシーは、Amazon DocumentDB がユーザーに代わってアクションを実行することを許可するサービスリンクロールにアタッチされます。詳細については、「エラスティッククラスター上のサービスにリンクされたロール」を参照してください。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

Amazon DocumentDB の AWS マネージドポリシーの更新

変更	説明	日付
AmazonDocDBElasticFullAccess、AmazonDocDBConsoleFullAccess - 変更	ポリシーが更新され、保留中のメンテナンスアクションが追加されました。	2/11/2025

変更	説明	日付
AmazonDocDBElasticFullAccess、AmazonDocDBConsoleFullAccess - 変更	ポリシーが更新され、クラスターの開始/停止とクラスタースナップショットアクションのコピーが追加されました。	2/21/2024
AmazonDocDBElasticReadOnlyAccess、AmazonDocDBElasticFullAccess - 変更	ポリシーを更新し、`cloudwatch:GetMetricData` アクションを追加。	6/21/2023
AmazonDocDBElasticReadOnlyAccess – 新しいポリシー	Amazon DocumentDB エラスティッククラスターの新しいマネージドポリシー。	6/8/2023
AmazonDocDBElasticFullAccess – 新しいポリシー	Amazon DocumentDB エラスティッククラスターの新しいマネージドポリシー。	6/5/2023
AmazonDocDB-ElasticServiceRolePolicy - 新しいポリシー	Amazon DocumentDB は、Amazon DocumentDB エラスティッククラスター用の新しい AWS ServiceRoleForDocDB-Elastic サービスリンクロールを作成します。Amazon DocumentDB	11/30/2022
AmazonDocDBConsoleFullAccess - 変更	Amazon DocumentDB グローバルクラスターとエラスティッククラスターのアクセス許可を追加するためにポリシーが更新されました。	11/30/2022

変更	説明	日付
AmazonDocDBConsoleFullAccess, AmazonDocDBFullAccess, AmazonDocDBReadOnlyAccess - 新しいポリシー	サービス起動。	1/19/2017

Amazon DocumentDB API のアクセス許可: アクション、リソース、条件リファレンス

次のセクションは、IAM アイデンティティに付加できる Amazon DocumentDB 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 およびアクセス許可ポリシー (アイデンティティベースのポリシー) を設定および書き込む際の参照として使用してください。

以下に各 Amazon DocumentDB API オペレーションを示します。リストには、アクションを実行するためのアクセス許可を付与できる対応するアクション、アクセス許可を付与できる AWS リソース、きめ細かなアクセスコントロールに含めることができる条件キーが含まれています。ポリシーの `Action` フィールドにアクションを、ポリシーの `Resource` フィールドにリソース値を、ポリシーの `Condition` フィールドに条件を指定します。条件の詳細については、「ポリシーでの条件の指定」を参照してください。

Amazon DocumentDB ポリシーで AWS全体の条件キーを使用して、条件を表現できます。 AWS全体のキーの完全なリストについては、IAM ユーザーガイドの「使用可能なキー」を参照してください。

IAM ポリシーシミュレーターを使用して IAM ポリシーをテストできます。Amazon DocumentDB AWS アクションなど、各アクションに必要なリソースとパラメータのリストが自動的に提供されます。IAM ポリシーシミュレーター により、指定する各アクションに必要な許可が決定されます。IAM ポリシーシミュレーターの詳細については、IAMユーザーガイド の IAM Policy Simulatorを使用したIAMポリシーのテスト を参照してください。

> **Note**
>
> アクションを指定するには、API オペレーション名 (`rds:CreateDBInstance` など) の前に `rds:` プレフィックスを使用します。

以下に、Amazon RDS API のオペレーションとそれに関連するアクション、リソース、および条件キーを示します。

トピック

- リソースレベルの許可をサポートする Amazon DocumentDB アクション
- リソースレベルの権限をサポートしない Amazon DocumentDB アクション

リソースレベルの許可をサポートする Amazon DocumentDB アクション

リソースレベルの許可とは、ユーザーがアクションを実行できるリソースを指定できる機能を提供します。Amazon DocumentDB は、リソースレベルのアクセス許可を部分的にサポートします。これは、特定の Amazon DocumentDB アクションについて、満たす必要のある条件、またはユーザーが使用を許可されている特定のリソースに基づいて、ユーザーがそれらのアクションをいつ使用できるかを制御できることを意味します。例えば、特定のインスタンスのみ変更するアクセス許可をユーザーに付与できます。

Amazon DocumentDB API オペレーションと関連するアクション、リソース、および条件キーを以下に示します。

> **Note**
>
> 特定の管理機能について、Amazon DocumentDB は Amazon RDS と共有されるオペレーショナルテクノロジーを使用します。Amazon DocumentDB アクションおよびアクセス許可の詳細については、「サービス認可リファレンス」の「Amazon RDS のアクション、リソース、および条件キー」を参照してください。

Amazon DocumentDB API オペレーションとアクション	リソース	条件キー
AddTagsToResource rds:AddTagsToResource	インスタンス arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
	サブネットグループ	rds:subgrp-tag

Amazon DocumentDB API オペレーションとアクション	リソース	条件キー
	arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	
ApplyPendingMaintenanceAction rds:ApplyPendingMaintenanceAction	インスタンス arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
CopyDBClusterSnapshot rds:CopyDBClusterSnapshot	クラススナップショット arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
CreateDBCluster rds:CreateDBCluster	クラスター arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	クラスターパラメータグループ arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
	サブネットグループ arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag

Amazon DocumentDB API オペレーションとアクション	リソース	条件キー
CreateDBClusterParameterGroup rds:CreateDBClusterParameterGroup	クラスターパラメータグループ arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
CreateDBClusterSnapshot rds:CreateDBClusterSnapshot	クラスター arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	クラスタースナップショット arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
CreateDBInstance rds:CreateDBInstance	インスタンス arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
	クラスター arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
CreateDBSubnetGroup rds:CreateDBSubnetGroup	サブネットグループ arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag

Amazon DocumentDB API オペレーションとアクション	リソース	条件キー
DeleteDBInstance rds:DeleteDBInstance	インスタンス arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
DeleteDBSubnetGroup rds:DeleteDBSubnetGroup	サブネットグループ arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribeDBClusterParameterGroups rds:DescribeDBClusterParameterGroups	クラスターパラメータグループ arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
DescribeDBClusterParameters rds:DescribeDBClusterParameters	クラスターパラメータグループ arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
DescribeDBClusters rds:DescribeDBClusters	クラスター arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag

Amazon DocumentDB API オペレーションとアクション	リソース	条件キー
DescribeDBClusterSnapshotAttributes rds:DescribeDBClusterSnapshotAttributes	クラスタースナップショット arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
DescribeDBSubnetGroups rds:DescribeDBSubnetGroups	サブネットグループ arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribePendingMaintenanceActions rds:DescribePendingMaintenanceActions	インスタンス arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
FailoverDBCluster rds:FailoverDBCluster	クラスター arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
ListTagsForResource rds:ListTagsForResource	インスタンス arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag

Amazon DocumentDB API オペレーションとアクション	リソース	条件キー
	サブネットグループ arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ModifyDBCluster rds:ModifyDBCluster	クラスター arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	クラスターパラメータグループ arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
ModifyDBClusterParameterGroup rds:ModifyDBClusterParameterGroup	クラスターパラメータグループ arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
ModifyDBClusterSnapshotAttribute rds:ModifyDBClusterSnapshotAttribute	クラスタースナップショット arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag

Amazon DocumentDB API オペレーションとアクション	リソース	条件キー
ModifyDBInstance rds:ModifyDBInstance	インスタンス arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
RebootDBInstance rds:RebootDBInstance	インスタンス arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
RemoveTagsFromResource rds:RemoveTagsFromResource	インスタンス arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
	サブネットグループ arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ResetDBClusterParameterGroup rds:ResetDBClusterParameterGroup	クラスターパラメータグループ arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
RestoreDBClusterFromSnapshot rds:RestoreDBClusterFromSnapshot	クラスター arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag

Amazon DocumentDB API オペレーションとアクション	リソース	条件キー
	クラススナップショット arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
RestoreDBClusterToPointInTime rds:RestoreDBClusterToPointInTime	クラスター arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
	サブネットグループ arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag

リソースレベルの権限をサポートしない Amazon DocumentDB アクション

Amazon DocumentDB ポリシーのすべての IAM アクションを使用して、そのアクションを使用する許可をユーザーに付与または拒否できます。ただし、すべての Amazon DocumentDB アクションが、アクションを実行することができるリソースを指定できる、リソースレベルのアクセス許可をサポートしているわけではありません。現在、次の Amazon DocumentDB API アクションは、リソースレベルの許可をサポートしていません。したがって、IAM ポリシーでこれらのアクションを使用するには、ステートメントで * 要素に Resource ワイルドカードを使って、アクションに対してすべてのリソースを使用する許可をユーザーに付与する必要があります。

- rds:DescribeDBClusterSnapshots
- rds:DescribeDBInstances

IAM ID を使用した認証

Amazon DocumentDB ユーザーやアプリケーションは、IAM ユーザーとロールを使用して Amazon DocumentDB クラスターに対する認証を行うことができます。Amazon DocumentDB IAM 認証は、パスワードレスな認証方法です。IAM ロール/ユーザーを使用する場合、クライアントアプリケーションはパスワードシークレットを Amazon DocumentDB クラスターに送信しません。代わりに、クライアント接続は一時的なセキュリティトークン AWS STS を使用して認証されます。管理者以外のユーザーとアプリケーションは、異なる Amazon DocumentDB クラスターやその他の AWS サービスに接続するときに、同じ IAM ID ARN を使用できるようになりました。

また、パスワードベース認証と IAM 認証を併用して、Amazon DocumentDB クラスターに対してユーザーやアプリケーションを認証することも可能です。IAM 認証は、Amazon DocumentDB インスタンスベースのクラスターバージョン 5.0 でのみ使用できます。なお、IAM ID ARN を用いた IAM 認証は、Amazon DocumentDB プライマリユーザーに対してはサポートしています。

> **Note**
>
> プライマリユーザーは、既存のパスワードベースの認証を使用してのみ認証できます。

トピック

- IAM ユーザーとロールを使用した認証の開始方法
- IAM を使用して Amazon DocumentDB AWS を認証するように AWS コンピューティングタイプを設定する
- IAM 認証リクエストのモニタリング
- IAM 認証の使用
- IAM をサポートするドライバー
- IAM ID 認証に関するよくある質問

IAM ユーザーとロールを使用した認証の開始方法

IAM ID を持つ Amazon DocumentDB ユーザーとロールは、`$external` データベースで作成および管理されます。

ユーザーの作成

プライマリユーザーとして接続し、IAM ユーザーとロールを作成します。

```
use $external;
db.createUser(
    {
        user: "arn:aws:iam::123456789123:user/iamuser",
        mechanisms: ["MONGODB-AWS"],
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
);
```

または、IAM ロールを使用して Amazon DocumentDB ユーザーを追加します。

```
use $external;
db.createUser(
    {
        user: "arn:aws:iam::123456789123:role/iamrole",
        mechanisms: ["MONGODB-AWS"],
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
);
```

IAM ユーザーまたはロールの変更

既存の IAM ユーザーの変更:

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:user/iamuser",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
);
```

既存の IAM ロールの変更:

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:role/iamrole",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
```

```
);
```

特定の IAM ユーザーに対してロールを付与または取り消すには:

```
use $external;
db.grantRolesToUser(
  "arn:aws:iam::123456789123:user/iamuser",
  [ { db: "admin", role: "readWriteAnyDatabase" } ]
);
```

```
use $external;
db.revokeRolesFromUser(
  "arn:aws:iam::123456789123:user/iamuser",
  [ { db: "admin", role: "readWriteAnyDatabase" } ]
);
```

特定の IAM ロールに対してロールを付与または取り消すには:

```
use $external;
db.grantRolesToUser(
  "arn:aws:iam::123456789123:user/iamrole",
  [ { db: "admin", role: "readWriteAnyDatabase" } ]
);
```

```
use $external;
db.revokeRolesFromUser(
  "arn:aws:iam::123456789123:user/iamrole",
  [ { db: "admin", role: "readWriteAnyDatabase" } ]
);
```

IAM ユーザーまたはロールの削除

既存の IAM ユーザーを削除するには:

```
use $external;
db.dropUser("arn:aws:iam::123456789123:user/iamuser");
```

既存の IAM ロールを削除するには:

```
use $external;
```

```
db.dropUser("arn:aws:iam::123456789123:role/iamrole");
```

IAM を使用して認証するように接続 URI AWS を設定する

AWS IAM を使用して認証するには、 authSourceとして $external、 authMechanismとして URI パラメータを使用しますMONGODB-AWS。IAM ユーザーを使用する場合、ユーザー名フィールドとパスワードフィールドはそれぞれアクセスキーとシークレットキーに置き換えられます。IAM ロールを引き受けている場合、現在の環境 (AWS Lambda 関数、Amazon EC2 インスタンスなど) にアタッチされます。MONGODB-AWS メカニズムを使用して認証する場合、特に認証情報を渡す必要はありません。MONGODB-AWS 認証メカニズムをサポートする MongoDB ドライバーを使用している場合、これらのドライバーにはコンピューティングインスタンス (Amazon EC2、Lambda 関数など) から IAM ロール認証情報を取得する能力もあります。次の例は、(IAM ユーザーの) アクセスキーとシークレットキーを手動で渡して行う、mongo シェル による MONGODB-AWS を使用した Amazon DocumentDB に対する認証プロセスを示します。

こちらの例は、(環境にアタッチされた IAM ロール を利用することで)、認証情報を明示的に渡さずに Python コードを使用して行う、MONGODB-AWS による Amazon DocumentDB に対する認証プロセスを示します。

```
##Create a MongoDB client, open a connection to Amazon DocumentDB using an IAM role
    client = pymongo.MongoClient('mongodb://<DocDBEndpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false&authSource=
%24external&authMechanism=MONGODB-AWS')
```

次の例は、(IAM ユーザーの) アクセスキーとシークレットキーを手動で渡して行う、mongo シェルによる MONGODB-AWS メカニズムを使用した Amazon DocumentDB に対する認証プロセスを示します。

```
$ mongo 'mongodb://<access_key>:<secret_key>@<cluster_endpoint>:<db_port>/test?
authSource=%24external&authMechanism=MONGODB-AWS'
```

こちらの例は、(環境にアタッチされた IAM ロール を利用することで)、認証情報を明示的に渡さずに mongo シェルを使用して行う、MONGODB-AWS による Amazon DocumentDB に対する認証プロセスを示します。

```
$ mongo 'mongodb://<cluster_endpoint>:<db_port>/test?authSource=
%24external&authMechanism=MONGODB-AWS'
```

IAM を使用して Amazon DocumentDB AWS を認証するように AWS コンピューティングタイプを設定する

Amazon EC2/AWS Lambda/AWS Fargateの使用

Amazon EC2 は次の環境変数を使用します。EC2 インスタンスにアタッチされた IAM ロール、または Lambda 関数または Amazon ECS タスクに関連付けられた実行 IAM ロールがある場合、これらの変数は自動的に入力され、ドライバーは環境からこれらの値を取得できます。

```
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_SESSION_TOKEN
```

環境変数に関する詳細については、「AWS Lambda 開発者ガイド」の「Lambda 環境関数の使用」を参照してください。

Amazon EKS の使用

Amazon Elastic Kubernetes Service (Amazon EKS) ポッドにロールを割り当てると、次の 2 つの環境変数が自動的に設定されます。

```
AWS_WEB_IDENTITY_TOKEN_FILE - path of web identity token file
AWS_ROLE_ARN - Name of IAM role to connect with
```

これらの変数を使用して、 の AWS SDK 呼び出しを使用してコードからロールを手動で引き受けますAssumeRoleWithWebIdentity。

- ProviderID のパラメータは省略します。
- AWS_WEB_IDENTITY_TOKEN_FILE 環境変数で説明されている ファイルで、WebIdentityToken パラメータの値を確認します。

詳細については、「Amazon EKS ユーザーガイド」の「Amazon EKS とは」を参照してください。

IAM 認証リクエストのモニタリング

Amazon DocumentDB 監査の使用

Amazon CloudWatch の監査ログフォルダに移動し、さまざまな検索パターンを使用して IAM 認証のログを取得することができます。例えば、「すべてのログストリームを検索する」の検索パターンとして { $.param.mechanism = "MONGODB-AWS" } を使用できます。

監査でサポートされているイベントの詳細については、「Amazon DocumentDB イベントの監査」を参照してください。

Amazon CloudWatch メトリクスを使用する

`StsGetCallerIdentityCalls`: このメトリクスは、Amazon DocumentDB インスタンスが Regionalized AWS Security Token Service （ AWS STS) エンドポイントに対して行っている`GetCallerIdentity`呼び出しの数を示します。データベースインスタンスが STS `GetCallerIdentity` コールを行う理由については、`MONGODB-AWS` 認証仕様を参照してください。

IAM 認証の使用

独自のデータベースでユーザー名やパスワードを管理したくない場合は、IAM 認証を使用できます。IAM 認証は、Amazon DocumentDB インスタンスベースのクラスターバージョン 5.0 でのみ使用できます。

IAM 認証は STS サービスに依存します。接続に IAM 認証を使用して STS スロットリング例外を取得することにより接続料金を下げることができるかどうかを、あらかじめ評価することをお勧めします。

IAM クォータについては、IAM ユーザーガイドの「IAM と AWS STS クォータ」を参照してください。

IAM をサポートするドライバー

Amazon DocumentDB 5.0 と `MONGODB-AWS` 認証メカニズムをサポートするドライバーは、Amazon DocumentDB の IAM 認証実装とも連携するはずです。

> **Important**
>
> バージョン 6.13.1 より古い Node.js ドライバーには既知の制限がありますが、現在 Amazon DocumentDB の IAM ID 認証ではサポートされていません。Node.js ドライバーを使用する Node.js ドライバーとツール (mongosh など) は、Node.js ドライバーバージョン 6.13.1 以降を使用するようにアップグレードする必要があります。

IAM ID 認証に関するよくある質問

参照できるサンプルはありますか？

サンプルユースケースと設定については、以下のページを参照してください。

- 人間のユーザーが、IAM ユーザーと IAM ロールを使用して Amazon DocumentDB に対して認証する方法
- IAM ロールを使用した、Amazon DocumentDB に対するパスワードレス認証

Python ドライバーの使用中にエラーが発生しました：「pymongo.errors.ConfigurationError: MONGODB-AWS authentication requires pymongo-auth-aws」。このエラーを解消するにはどうすればよいですか？

IAM 認証で Python ドライバーをインストールする際には、必ず次のステートメントを使用してください。

```
pip install 'pymongo[aws]'
```

これにより、IAM 認証が機能するために必要な追加の AWS 依存関係がインストールされます。

IAM ロールの一時的な認証情報の有効期限が切れると、接続は切断されますか?

いいえ。一時的な IAM 認証情報は、接続と認証の確立にのみ使用されます。その後のすべての承認は Amazon DocumentDB クラスター内で行われます。IAM 認証情報がローテーションされたり、あるいは期限切れになっても、接続が削除されたり古くなることはありません。

Amazon DocumentDB ユーザーの管理

Amazon DocumentDB の場合、ユーザーはクラスターへの認証にパスワードを接続します。各クラスターには、クラスターの作成時に設定されるプライマリサインイン認証情報があります。

> **Note**
>
> 2020 年 3 月 26 日 より前に作成されたすべての新規ユーザーには `dbAdminAnyDatabase`、`readWriteAnyDatabase`、および `clusterAdmin` のロールが付与されています。クラスター内のすべてのユーザーに最小限の特権を適用するために、すべてのユーザーを再評価し、必要に応じてロールを変更することをお勧めします。
> 詳細については、「ロールベースのアクセス制御 (RBAC) を使用したデータベースへのアクセス」を参照してください。

プライマリおよび serviceadmin ユーザ

新しく作成された Amazon DocumentDB クラスターには、マスターユーザーと serviceadmin ユーザーの 2 つのユーザーがあります。

マスターユーザー は単一の特権ユーザーであり、管理タスクの実行と、ロールを持つ追加のユーザーの作成を行うことができます。Amazon DocumentDB クラスターに初めて接続するときは、マスターユーザー名およびパスワードを使用して認証する必要があります。マスターユーザーには、Amazon DocumentDB クラスターの作成時に該当する管理権限が提供され、さらに root のロールが付与されます。

serviceadmin ユーザーは、クラスターの作成時に暗黙で作成されます。すべての Amazon DocumentDB クラスターには、クラスターを管理する AWS 機能を提供するserviceadminユーザーがあります。パスワードでのログイン、パスワードの削除や名前変更、または serviceadmin のアクセス許可の変更を行うことはできません。それらの操作を試みると、エラーが発生します。

> **Note**
>
> Amazon DocumentDB クラスターのマスターユーザーや serviceadmin ユーザーを削除したり、マスターユーザーの root のロールを無効にしたりすることはできません。
> プライマリユーザーのパスワードを忘れた場合は、 AWS Management Console または を使用してリセットできます AWS CLI。

追加ユーザーの作成

マスターユーザー (または createUser ロールを持つユーザー) として接続すると、次に示すように、新しいユーザーを作成できます。

```
db.createUser(
    {
        user: "sample-user-1",
        pwd: "password123",
        roles:
            [{"db":"admin", "role":"dbAdminAnyDatabase" }]
    }
)
```

ユーザの詳細を表示するには、次のように show users コマンドを使用できます。また、dropUser コマンドを使用してユーザーを削除することもできます。詳細については、「一般的なコマンド」を参照してください。

```
show users
{
    "_id" : "serviceadmin",
    "user" : "serviceadmin",
    "db" : "admin",
    "roles" : [
     {
            "role" : "root",
            "db" : "admin"
        }
    ]
},

{
    "_id" : "myPrimaryUser",
    "user" : "myPrimaryUser",
    "db" : "admin",
    "roles" : [
     {
            "role" : "root",
            "db" : "admin"
        }
    ]
},

{
    "_id" : "sample-user-1",
    "user" : "sample-user-1",
    "db" : "admin",
    "roles" : [
     {
            "role" : "dbAdminAnyDatabase",
            "db" : "admin"
     }
    ]
}
```

この例では、新しいユーザー sample-user-1 を admin データベースに帰属させています。これは、新しいユーザーには常に当てはまります。Amazon DocumentDB には、authenticationDatabase のコンセプトを含んでおらず、したがって、すべての認証は admin データベースで実施されます。

ユーザーの作成時に、db フィールドを省略してロールを指定すると、Amazon DocumentDB は接続の発行先であるデータベースに暗黙でロールを帰属させます。たとえば、データベース sample-database に対して接続を発行し、次のコマンドを実行すると、ユーザー sample-user-2 が admin データベース内に作成され、データベース sample-database への readWrite アクセス許可が付与されます。

```
db.createUser(
    {
        user: "sample-user-2",
        pwd: "password123",
        roles:
            ["readWrite"]
    }
)
```

すべてのデータベースを対象とするロール (readInAnyDatabase など) を持つユーザーを作成するには、ユーザーの作成時に admin データベースのコンテキストで操作をしているか、ユーザーの作成時にロールのデータベースを明示的に指定する必要があります。

データベースのコンテキストを切り替えるには、次のコマンドを使用できます。

```
use admin
```

ロールベースのアクセスコントロールや、クラスター内のすべてのユーザーに最小限の特権を適用する方法の詳細については、「ロールベースのアクセス制御 (RBAC) を使用したデータベースへのアクセス」を参照してください。

Amazon DocumentDB のパスワードの自動ローテーション

を使用すると AWS Secrets Manager、コード内のハードコードされた認証情報 (パスワードを含む) を Secrets Manager への API コールに置き換えて、プログラムでシークレットを取得できます。シークレットはそこに存在しないため、これは、あなたのコードを調べている誰かがシークレットを漏らさないようにするのに役立ちます。また、指定したスケジュールに従って自動的にシークレット

を更新するように Secrets Manager を設定することができます。これにより、長期のシークレットを短期のシークレットに置き換えることが可能となり、侵害されるリスクが大幅に減少します。

Secrets Manager を使用すると、Secrets Manager が提供する AWS Lambda 関数を使用して、Amazon DocumentDB パスワード (つまり、シークレット) を自動的にローテーションできます。

AWS Secrets Manager および Amazon DocumentDB とのネイティブ統合の詳細については、以下を参照してください。

- ブログ: How to rotate Amazon DocumentDB and Amazon Redshift credentials in AWS Secrets Manager
- AWS Secrets Manager とは
- AWS Secrets Manager シークレットのローテーション
- Secrets Manager の Amazon DocumentDB 認証情報

ロールベースのアクセス制御 (RBAC) を使用したデータベースへのアクセス

Amazon DocumentDB (MongoDB 互換) の ロールベースアクセス制御 (RBAC)を使用して、ユーザーがデータベースで実行できるアクションへのアクセスを制限できます。RBAC は、1 つ以上のロールをユーザーに付与することで機能します。これらのロールは、ユーザーがデータベースリソースに対して実行できるオペレーションを決定します。Amazon DocumentDB は、現在、`read`、`readWrite`、`readAnyDatabase`、`clusterAdmin` のようなデータベースレベルを対象とする組み込みロール、および要件に基づいて特定のアクションと詳細なリソースの両方をサポートしています。

RBAC の一般的ユースケースには、クラスター内のデータベースへの読み取り専用アクセス権を持つユーザーを作成して最小特権を強制したり、クラスター内の特定のデータベースにアクセスすることを単一のユーザーに許可するマルチテナントアプリケーションを設計したりすることが含まれます。

> **Note**
>
> 2020 年 3 月 26 日 より前に作成されたすべての新規ユーザーには `dbAdminAnyDatabase`、`readWriteAnyDatabase`、および `clusterAdmin` のロールが

付与されています。すべての既存ユーザーを再評価し、必要に応じてロールを変更してクラスターに最小特権を適用することをお勧めします。

トピック

- RBAC の概念
- RBAC 組み込みロールの開始方法
- RBAC ユーザー定義ロールの開始方法
- ユーザーとして Amazon DocumentDB に接続する
- 一般的なコマンド
- 機能的な違い
- 制限
- ロールベースのアクセス制御 (RBAC) を使用したデータベースへのアクセス

RBAC の概念

ロールベースのアクセスコントロールに関連する重要な用語と概念を以下に示します。Amazon DocumentDB ユーザーの詳細については、Amazon DocumentDB ユーザーの管理 を参照してください。

- ユーザー - データベースに対して認証し、オペレーションを実行できる個別のエンティティ。
- パスワード - ユーザーの認証に使用されるシークレット。
- ロール - 1 つ以上のデータベースに対してアクションを実行することをユーザーに許可します。
- 管理データベース - ユーザーを保存して認証する先のデータベース。
- データベース (**db**) - ドキュメントを保存するためのコレクションを含むクラスター内の名前空間。

次のコマンドは、`sample-user` というユーザーを作成します。

```
db.createUser({user: "sample-user", pwd: "abc123", roles: [{role: "read", db: "sample-database"}]})
```

この例では、以下のようになっています：

- `user: "sample-user"` - ユーザー名を示します。

- pwd: "abc123" - ユーザーのパスワードを示します。
- role: "read", "db: "sample-database" - ユーザー sample-user に sample-database への読み取り許可を付与することを示します。



次の例は、db.getUser(sample-user) を使用してユーザー sample-user を取得した後の出力を示しています。この例では、ユーザー sample-user は admin データベース内に存在しますが、データベース sample-database の読み取りロールを持っています。



ユーザーの作成時に、db フィールドを省略してロールを指定すると、Amazon DocumentDB は接続の発行先であるデータベースに暗黙でロールを帰属させます。たとえば、データベース sample-database に対して接続を発行し、次のコマンドを実行すると、ユーザー sample-user が admin データベース内に作成され、データベース sample-database への readWrite アクセス許可が付与されます。

```
db.createUser({user: "sample-user", pwd: "abc123", roles: ["readWrite"]})
```

このオペレーションによる出力は、次のようになります。

```
{
   "user":"sample-user",
   "roles":[
```

```
      {
         "db":"sample-database",
         "role":"readWrite"
      }
   ]
}
```

すべてのデータベースを対象とするロール (readAnyDatabase など) を持つユーザーを作成するには、ユーザーの作成時に admin データベースのコンテキストで操作をしているか、ユーザーの作成時にロールのデータベースを明示的に指定する必要があります。admin データベースに対してコマンドを発行するには、コマンド use admin を使用します。詳細については、「一般的なコマンド」を参照してください。

RBAC 組み込みロールの開始方法

ロールベースのアクセスコントロールの使用を簡単に開始できるように、このセクションでは、異なるジョブ機能を持つ 3 人のユーザーのロールを作成して最小特権を強制するシナリオの例を示します。

- user1 は、クラスター内のすべてのデータベースの表示とアクセスを必要とする新しいマネージャーです。
- user2 は、同じクラスター内の 1 つのデータベース (sample-database-1) にのみアクセスする必要がある新しい従業員です。
- user3 は、同じクラスター内の、以前はアクセスできなかった別のデータベース (sample-database-2) の表示とアクセを必要とする既存の従業員です。

後で、user1 と user2 が退職し、この両方のユーザーのアクセス権を取り消す必要があります。

ユーザーを作成してロールを付与する場合、クラスターに対して認証を行うユーザーには、createUser と grantRole のアクションを実行できるロールが関連付けられている必要があります。たとえば、admin ロールと userAdminAnyDatabase ロールは両方とも、該当するアクションのアクセス許可を付与できます。ロール別のアクションについては、「ロールベースのアクセス制御 (RBAC) を使用したデータベースへのアクセス」を参照してください。

> **Note**
>
> Amazon DocumentDB では、create データベースに対してコマンドを発行するかどうかにかかわらず、ユーザーおよびロールのすべてのオペレーション

(get、drop、grant、revoke、admin など) が admin データベース内で暗黙で実行されます。

まず、クラスター内の現在のユーザーとロールを確認するために、次の例に示すように show users コマンドを実行します。クラスター内の 2 人のユーザー、serviceadminとプライマリユーザーが表示されます。これらの 2 人のユーザーは常に存在し、削除することはできません。詳細については、「Amazon DocumentDB ユーザーの管理」を参照してください。

```
show users
```

user1 に対して、次のコマンドを使用し、クラスター全体のすべてのデータベースに対する読み取りおよび書き込みアクセス権を持つロールを作成します。

```
db.createUser({user: "user1", pwd: "abc123", roles: [{role: "readWriteAnyDatabase", db:
 "admin"}]})
```

このオペレーションによる出力は、次のようになります。

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

user2 に対して、次のコマンドを使用し、データベース sample-database-1 への読み取り専用アクセス権を持つロールを作成します。

```
db.createUser({user: "user2", pwd: "abc123", roles: [{role: "read", db: "sample-
database-1"}]})
```

このオペレーションによる出力は、次のようになります。

```
{
```

```
   "user":"user2",
   "roles":[
      {
         "role":"read",
         "db":"sample-database-1"
      }
   ]
}
```

user3 を既存のユーザーとするシナリオをシミュレートするには、まずユーザー user3 を作成し、次に user3 新しいロールを割り当てます。

```
db.createUser({user: "user3", pwd: "abc123", roles: [{role: "readWrite", db: "sample-
database-1"}]})
```

このオペレーションによる出力は、次のようになります。

```
{
   "user":"user3",
   "roles":[
      {
         "role":"readWrite",
         "db":"sample-database-1"
      }
   ]
}
```

ユーザー user3 を作成したので、この user3 に対して sample-database-2 の read ロールを割り当てます。

```
db.grantRolesToUser("user3", [{role: "read", db: "sample-database-2"}])
```

最後に、user1 と user2 が退職し、この両ユーザーのクラスターに対するアクセス権を取り消す必要があります。これを行うには、次のようにユーザーを削除します。

```
db.dropUser("user1")
db.dropUser("user2")
```

すべてのユーザーが適切なロールを持っていることを確認するには、次のコマンドを使用してすべてのユーザーを一覧表示します。

```
show users
```

このオペレーションによる出力は、次のようになります。

```
{
   "_id":"serviceadmin",
   "user":"serviceadmin",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"root"
      }
   ]
}
{
   "_id":"master-user",
   "user":"master-user",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"root"
      }
   ]
}
{
   "_id":"user3",
   "user":"user3",
   "db":"admin",
   "roles":[
      {
         "db":"sample-database-2",
         "role":"read"
      },
      {
         "db":"sample-database-1",
         "role":"readWrite"
      }
   ]
}
```

RBAC ユーザー定義ロールの開始方法

このセクションでは、ユーザー定義の役割を開始するのに役立つように、ジョブ機能が異なる 3 人のユーザーにロールを作成して最小特権を適用するシナリオの例について説明します。

この例では、以下が適用されます。

- user1 は、クラスター内のすべてのデータベースの表示とアクセスを必要とする新しいマネージャーです。
- user2 は、同じクラスター内の 1 つのデータベース sample-database-1 にのみアクセスする必要がある新しい従業員です。
- user3 は、同じクラスター内で以前はアクセスできなかった sample-database-2、別のデータベースの col2 という特定のコレクションを表示してアクセスする必要がある既存の従業員です。
- user1 に対して、次のコマンドを使用し、クラスター全体のすべてのデータベースに対する読み取りおよび書き込みアクセス権を持つロールを作成します。

```
db.createUser(
 {
     user: "user1", pwd: "abc123",
     roles: [{role: "readWriteAnyDatabase", db: "admin"}]
 }
 )
```

このオペレーションによる出力は、次のようになります。

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

user2 の場合、次のコマンドを使用して、データベース sample-database-1 内のすべてのコレクションに対する「検索」権限を持つロールを作成します。このロールは、関連付けられたすべてのユーザーが検索クエリのみを実行できるようにすることに注意してください。

```
db.createRole(
{
    role: "findRole",
    privileges: [
    {
        resource: {db: "sample-database-1", collection: ""}, actions: ["find"]
    }],
    roles: []
}
)
```

このオペレーションによる出力は、次のようになります。

```
{
   "role":"findRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-1",
            "collection":""
         },
         "actions":[
            "find"
         ]
      }
   ],
   "roles":[

   ]
}
```

次に、ユーザー (user2) を作成し、最近作成したロール findRole をユーザーに接続します。

```
db.createUser(
{
    user: "user2",
    pwd: "abc123",
    roles: []
})

 db.grantRolesToUser("user2",["findRole"])
```

user3 を既存のユーザーとするシナリオをシミュレートするには、まずユーザー user3 を作成し、次のステップで user3 に割り当てる collectionRole という新しいロールを作成します。

そして、新しいロールを user3 に割り当てることができます。この新しいロールにより、user3 の 1 つの特定のコレクション col2 へのアクセスを挿入、更新、削除、および検索できる sample-database-2 を許可します。

```
db.createUser(
{
    user: "user3",
    pwd: "abc123",
    roles: []
})

db.createRole(
{
    role: "collectionRole",
    privileges: [
    {
        resource: {db: "sample-database-2", collection: "col2"}, actions: ["find",
 "update", "insert", "remove"]
    }],
    roles: []
}
)
```

このオペレーションによる出力は、次のようになります。

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
            "collection":"col2"
         },
         "actions":[
            "find",
            "update",
            "insert",
            "remove"
         ]
      }
```

```
   ],
   "roles":[

   ]
}
```

ユーザー user3 を作成したので、user3 にロール collectionFind を付与できます。

```
db.grantRolesToUser("user3",["collectionRole"])
```

最後に、user1 と user2 が退職し、この両ユーザーのクラスターに対するアクセス権を取り消す必要があります。これを行うには、次のようにユーザーを削除します。

```
db.dropUser("user1")
db.dropUser("user2")
```

すべてのユーザーが適切なロールを持っていることを確認するには、次のコマンドを使用してすべてのユーザーを一覧表示します。

```
show users
```

このオペレーションによる出力は、次のようになります。

```
{
   "_id":"serviceadmin",
   "user":"serviceadmin",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"root"
      }
   ]
}
{
   "_id":"master-user",
   "user":"master-user",
   "db":"admin",
   "roles":[
      {
```

```
         "db":"admin",
         "role":"root"
      }
   ]
}
{
   "_id":"user3",
   "user":"user3",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"collectionRole"
      }
   ]
}
```

ユーザーとして Amazon DocumentDB に接続する

Amazon DocumentDB クラスターに接続するときは、特定のデータベースのコンテキストで接続します。デフォルトでは、接続文字列でデータベースを指定しない場合、test データベースのコンテキストでクラスターに自動的に接続されます。insert や find などのすべてのコレクションレベルのコマンドは、test データベース内のコレクションに対して発行されます。

コンテキスト内にあるデータベース、つまりコマンドの発行を確認するには、以下のように、mongo シェル内の db コマンドを実行します。

クエリ:

```
db
```

出力:

```
test
```

デフォルトの接続が test データベースのコンテキストに存在する場合でも、この接続に関連付けられているユーザーが test データベースに対するアクションの実行を許可されているとは限りません。前のシナリオ例で、sample-database-1 データベースに対する readWrite ロールを持つユーザー user3 として認証する場合、接続のデフォルトのコンテキストは test データベースです。ただし、test データベースのコレクションにドキュメントを挿入しようとすると、認可の失

敗エラーメッセージが表示されます。これは、このコマンドをデータベースに対して実行する権限がユーザーに許可されていないことが原因です。

クエリ:

```
db
```

出力:

```
test
```

クエリ:

```
db.col.insert({x:1})
```

出力:

```
WriteCommandError({ "ok" : 0, "code" : 13, "errmsg" : "Authorization failure" })
```

接続のコンテキストを sample-database-1 データベースに変更すると、ユーザーが書き込み権限を持つコレクションに書き込むことができます。

クエリ:

```
use sample-database-1
```

出力:

```
switched to db sample-database-1
```

クエリ:

```
db.col.insert({x:1})
```

出力:

```
WriteResult({ "nInserted" : 1})
```

特定のユーザーとしてクラスターに対して認証を行う場合は、接続文字列でデータベースを指定することもできます。これにより、ユーザーが admin データベースに対して認証された後で use コマンドを実行する必要がなくなります。

次の接続文字列では、admin データベースに対してユーザーを認証しますが、接続のコンテキストは sample-database-1 データベースになります。

```
mongo "mongodb://user3:abc123@sample-cluster.node.us-east-1.docdb.amazonaws.com:27017/
sample-database-2"
```

一般的なコマンド

このセクションでは、Amazon DocumentDB でロールベースのアクセスコントロールを使用する一般的なコマンドの例を示します。ユーザーとロールを作成および変更するには、admin データベースのコンテキストで操作する必要があります。use admin コマンドを使用して、admin データベースに切り替えることができます。

> **Note**
>
> ユーザーおよびロールに対する変更は、admin データベース内で暗黙で実行されます。すべてのデータベースを対象とするロール (readAnyDatabase など) を持つユーザーを作成するには、ユーザーの作成時に admin データベース (use admin) のコンテキストで操作をしているか、ユーザーの作成時にロールのデータベースを明示的に指定する必要があります (このセクションの例 2 を参照)。

例 1: データベース read 用の foo ロールを持つユーザーを作成します。

```
db.createUser({user: "readInFooBar", pwd: "abc123", roles: [{role: "read", db:
 "foo"}]})
```

このオペレーションによる出力は、次のようになります。

```
{
   "user":"readInFooBar",
   "roles":[
      {
         "role":"read",
```

```
         "db":"foo"
      }
   ]
}
```

例 2: すべてのデータベースへの読み取りアクセス権を持つユーザーを作成します。

```
db.createUser({user: "readAllDBs", pwd: "abc123", roles: [{role: "readAnyDatabase", db:
 "admin"}]})
```

このオペレーションによる出力は、次のようになります。

```
{
   "user":"readAllDBs",
   "roles":[
      {
         "role":"readAnyDatabase",
         "db":"admin"
      }
   ]
}
```

例 3: 新しいデータベースの既存のユーザーに read ロールを付与します。

```
db.grantRolesToUser("readInFooBar", [{role: "read", db: "bar"}])
```

例 4: ユーザーのロールを更新します。

```
db.updateUser("readInFooBar", {roles: [{role: "read", db: "foo"}, {role: "read", db:
 "baz"}]})
```

例 5: ユーザーのデータベースへのアクセスを取り消します。

```
db.revokeRolesFromUser("readInFooBar", [{role: "read", db: "baz"}])
```

例 6: 組み込みロールを説明します。

```
db.getRole("read", {showPrivileges:true})
```

このオペレーションによる出力は、次のようになります。

```
{
   "role":"read",
   "db":"sample-database-1",
   "isBuiltin":true,
   "roles":[

   ],
   "inheritedRoles":[

   ],
   "privileges":[
      {
         "resource":{
            "db":"sample-database-1",
            "collection":""
         },
         "actions":[
            "changeStream",
            "collStats",
            "dbStats",
            "find",
            "killCursors",
            "listCollections",
            "listIndexes"
         ]
      }
   ],
   "inheritedPrivileges":[
      {
         "resource":{
            "db":"sample-database-1",
            "collection":""
         },
         "actions":[
            "changeStream",
            "collStats",
            "dbStats",
            "find",
            "killCursors",
            "listCollections",
            "listIndexes"
         ]
```

```
      }
   }
```

例 7: クラスターからユーザーを削除します。

```
db.dropUser("readInFooBar")
```

このオペレーションによる出力は、次のようになります。

```
true
```

例 8: 特定のコレクションに対する読み取りおよび書き込みアクセス権を持つロールを作成する

```
db.createRole(
{
    role: "collectionRole",
    privileges: [
    {
        resource: {db: "sample-database-2", collection: "col2"}, actions: ["find",
 "update", "insert", "remove"]
    }],
    roles: []
}
)
```

このオペレーションによる出力は、次のようになります。

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
            "collection":"col2"
         },
         "actions":[
            "find",
            "update",
            "insert",
            "remove"
         ]
```

```
      }
   ],
   "roles":[

   ]
}
```

例 9: ユーザーを作成し、ユーザー定義ロールを割り当てる

```
db.createUser(
{
    user: "user3",
    pwd: "abc123",
    roles: []
})

db.grantRolesToUser("user3",["collectionRole"])
```

例 10: ユーザー定義ロールに追加権限を付与する

```
db.grantPrivilegesToRole(
  "collectionRole",
  [
    {
      resource: { db: "sample-database-1", collection: "col1" },
      actions: ["find", "update", "insert", "remove"]
    }
  ]
)
```

例 11: ユーザー定義ロールから権限を削除する

```
db.revokePrivilegesFromRole(
  "collectionRole",
  [
    {
      resource: { db: "sample-database-1", collection: "col2" },
      actions: ["find", "update", "insert", "remove"]
    }
  ]
)
```

例 12: 既存のユーザー定義ロールを更新する

```
db.updateRole(
  "collectionRole",
  {
    privileges: [
    {
        resource: {db: "sample-database-3", collection: "sample-collection-3"},
 actions: ["find", "update", "insert", "remove"]
    }],
    roles: []
  }
)
```

機能的な違い

Amazon DocumentDB では、ユーザーとロールの定義が admin データベースに保存され、ユーザーは admin データベースに対して認証されます。この機能は MongoDB Community Edition とは異なりますが、MongoDB Atlas と一貫しています。

Amazon DocumentDB は、変更ストリームもサポートしています。変更ストリームは、クラスターのコレクション内で発生する変更イベントの時系列シーケンスを提供します。listChangeStreams アクションは、クラスターレベル (つまり、すべてのデータベースを対象に) 適用されます。modifyChangeStreams アクションは、データベースレベルで適用されます。

制限

次のテーブルは Amazon DocumentDB におけるロールベースアクセスコントロールの制限を含む

説明	[制限]
クラスターあたりのユーザー数	1,000
ユーザーに関連付けられているロールの数	1,000
ユーザー定義ロールの数	100
権限に関連付けられているリソースの数	100

ロールベースのアクセス制御 (RBAC) を使用したデータベースへのアクセス

ロールベースのアクセスコントロールでは、ユーザーを作成して 1 つ以上のロールを付与し、ユーザーがデータベースやクラスターで実行できるオペレーションを指定できます。

Amazon DocumentDB で現在サポートされている組み込みロールは次のとおりです。

> **Note**
>
> Amazon DocumentDB 4.0 と 5.0 では、それぞれ `ListCollection` および `ListDatabase` コマンドでは、オプションで `authorizedCollections` および `authorizedDatabases` パラメータを使用して、ユーザーが権限を持っているコレクションとデータベースを一覧表示し、`listCollections` および `listDatabase` ロールを要求するアクセスをします。また、ユーザーは `KillCursor` ロールを要求すねことなく、自分のカーソルを強制終了できるようになりました。

Database user

ロール名	説明	アクション

`read`	指定したデータベースへの読み取りアクセス権をユーザーに付与します。	changeStreams
		`collStats`
		`dbStats`
		`find`
		`killCursors`
		`listIndexes`
		`listCollections`

`readWrite`	指定したデータベースへの読み取りおよび書き込みアクセス権をユーザーに付与します。	`read` アクセス許可のすべてのアクション。

ロール名	説明	アクション
		createCollection dropCollection createIndex dropIndex insert killCursors listIndexes listCollections remove update

Cluster user

ロール名	説明	アクション
readAnyDatabase	クラスター内のすべてのデータベースへの読み取りアクセス権をユーザーに付与します。	read アクセス許可のすべてのアクション。 listChangeStreams listDatabases
readWriteAnyDatabase	クラスター内のすべてのデータベースへの読取りおよび書込みアクセス権をユーザーに付与します。	readWrite アクセス許可のすべてのアクション。

ロール名	説明	アクション
		listChangeStreams listDatabases
userAdminAnyDatabase	指定したデータベースに対するユーザーのロールや特権を割り当てたり変更したりできる権限をユーザーに付与します。	changeCustomData changePassword createUser dropRole dropUser grantRole listDatabases revokeRole viewRole viewUser
dbAdminAnyDatabase	指定したデータベースに対してデータベース管理ロールを実行する権限をユーザーに付与します。	dbAdmin アクセス許可のすべてのアクション。 dropCollection listDatabases listChangeStreams modifyChangeStreams

Superuser

ロール名	説明	アクション
root	すべてのロール (readWriteAnyDatabase 、dbAdminAnyDatabase 、userAdminAnyDatabase 、clusterAdmin 、restore、backup) のリソースとオペレーションに対するアクセス権をユーザーに付与します。	readWriteAnyDatabase 、dbAdminAnyDatabase 、userAdminAnyDatabase 、clusterAdmin 、restore および backup のすべてのアクション。

Database administrator

ロール名	説明	アクション
dbAdmin	指定したデータベースに対して管理タスクを実行する権限をユーザーに付与します。	bypassDocumentValidation collMod collStats createCollection createIndex dropCollection dropDatabase dropIndex dbStats

ロール名	説明	アクション
		find killCursors listIndexes listCollections modifyChangeStreams
dbOwner	dbAdmin ロールと readWrite ロールを組み合わせて、指定したデータベースに対する管理タスクを実行する権限をユーザーに付与します。	dbAdmin と readWrite のすべてのアクション。

Cluster administrator

ロール名	説明	アクション
clusterAdmin	clusterManager 、clusterMonitor 、および hostManager の各ロールを組み合わせで、最大のクラスター管理アクセス権をユーザーに付与します。	clusterManager 、clusterMonitor 、および hostManager のすべてのアクション。 listChangeStreams dropDatabase modifyChangeStreams

ロール名	説明	アクション
clusterManager	指定したクラスターに対して管理とモニタリングのアクションを実行する権限をユーザーに付与します。	listChangeStreams listSessions modifyChangeStreams replSetGetConfig

ロール名	説明	アクション
clusterMonitor	モニタリングツールへの読み取り専用アクセス権をユーザーに付与します。	collStats dbStats find getParameter hostInfo indexStats killCursors listChangeStreams listCollections listDatabases listIndexes listSessions replSetGetConfig serverStatus top

ロール名	説明	アクション
hostManager	サーバーをモニタリングおよび管理する権限をユーザーに付与します。	auditConfigure killCursors killAnyCursor killAnySession killop

Backup administrator

ロール名	説明	アクション
backup	データのバックアップに必要なアクセス権をユーザーに付与します。	getParameter insert find listChangeStreams listCollections listDatabases listIndexes update
restore	データの復元に必要なアクセス権をユーザーに付与します。	bypassDocumentValidation changeCustomData

ロール名	説明	アクション
		changePassword collMod createCollection createIndex createUser dropCollection dropRole dropUser getParameter grantRole find insert listCollections modifyChangeStreams revokeRole remove viewRole viewUser update

Amazon DocumentDB でのログ記録とモニタリング

Amazon DocumentDB (MongoDB 互換) は、Amazon DocumentDB クラスターとインスタンスの健全性とパフォーマンスを判断するためにモニタリングできる、さまざまな Amazon CloudWatch メトリクスを提供します。Amazon DocumentDB コンソール、、Amazon CloudWatch コンソール、CloudWatch API など、さまざまなツールを使用して AWS CLI Amazon DocumentDB メトリクスを表示できます。モニタリングの詳細については、「Amazon DocumentDB のモニタリング」をご参照ください。

Amazon CloudWatch メトリクスに加えて、プロファイラーを使用して、クラスターで実行されたオペレーションの実行時間と詳細をログできます。プロファイラーは、クラスターで最も遅いオペレーションをモニタリングし、個々のクエリパフォーマンスとクラスター全体のパフォーマンスを向上させるのに役立ちます。有効にすると、オペレーションが Amazon CloudWatch Logs にログされ、CloudWatch インサイトを使用して Amazon DocumentDB プロファイリングデータを分析、モニタリング、アーカイブできます。詳細については、「Amazon DocumentDB オペレーションのプロファイリング」を参照してください。

Amazon DocumentDB は、Amazon DocumentDB (MongoDB 互換) のユーザー、ロール、または AWS サービスによって実行されたアクションを記録するサービスである AWS CloudTrail とも統合されています。CloudTrail は、Amazon DocumentDB からの呼び出しや Amazon DocumentDB AWS Management Console SDK へのコード呼び出しを含むAmazon DocumentDB のすべての AWS CLI API コールをイベントとしてキャプチャします。詳細については、「を使用した Amazon DocumentDB API コールのログ記録 AWS CloudTrail」を参照してください。

Amazon DocumentDB を使用すると、クラスター内で実行されたイベントを監査できます。ログに記録されるイベントの例としては、認証の成功と失敗、データベース内のコレクションの削除、インデックスの作成などがあります。デフォルト では、監査が Amazon DocumentDB 上で無効化されていて、この機能を選択する必要があります。詳細については、「Amazon DocumentDB イベントの監査」を参照してください。

Amazon DocumentDB TLS 証明書の更新

トピック

- アプリケーションと Amazon DocumentDB クラスターの更新
- サーバー証明書の自動ローテーション
- トラブルシューティング
- よくある質問

Amazon DocumentDB クラスターの認証機関 (CA) 証明書は、2024 年 8 月に更新されました。TLS (Transport Layer Security) が有効 (デフォルト設定) になっている クラスターを使用しており、クライアントアプリケーション証明書とサーバー証明書を更新していない場合は、アプリケーションと Amazon DocumentDB クラスターとの接続問題を軽減するために次の手順を実行する必要があります。

- ステップ 1: 新しい CA 証明書をダウンロードしてアプリケーションを更新する
- ステップ 2: サーバー証明書を更新する

CA とサーバーの証明書は、Amazon DocumentDB のための標準的なメンテナンスおよびセキュリティのベストプラクティスの一環として更新されています。クライアントアプリケーションは、新しい CA 証明書をトラストストアに追加する必要があり、この有効期限より前に新しい CA 証明書を使用するために既存の Amazon DocumentDB インスタンスを更新する必要があります。

アプリケーションと Amazon DocumentDB クラスターの更新

このセクションの手順に従って、アプリケーションの CA 証明書バンドル (ステップ 1) とクラスターのサーバー証明書 (ステップ 2) を更新します。本番環境に変更を適用する前に、開発環境またはステージング環境でこれらの手順をテストすることを強くお勧めします。

> **Note**
>
> Amazon DocumentDB クラスターがある各 AWS リージョン でステップ 1 と 2 を完了する必要があります。

ステップ 1: 新しい CA 証明書をダウンロードしてアプリケーションを更新する

新しい CA 証明書をダウンロードし、新しい CA 証明書を使用して Amazon DocumentDB への TLS 接続を作成するようにアプリケーションを更新します。https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem から新しい CA 証明書バンドルをダウンロードします。このオペレーションにより、`global-bundle.pem` という名前のファイルがダウンロードされます。

> **Note**
>
> 古い CA 証明書 (`rds-ca-2019-root.pem`) と新しい CA 証明書 (`rds-ca-rsa2048-g1`、`rds-ca-rsa4096-g1`、`rds-ca-ecc384-g1`) の両方を含むキーストアにアクセスしている場合は、キーストアで `global-bundle` が選択されるか確認してください。

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

次に、新しい証明書バンドルを使用するようにアプリケーションを更新します。新しい CA バンドルには、古い CA 証明書 (rds-ca-2019) と新しい CA 証明書 (rds-ca-rsa2048-g1, rds-ca-rsa4096-g1, rds-ca-ecc384-g1) の両方が含まれています。新しい CA バンドルに両方の CA 証明書を使用すると、2 つの手順でアプリケーションとクラスターを更新できます。

Java アプリケーションの場合、新しい CA 証明書を使用して新しい信頼ストアを作成する必要があります。手順については、「TLS が有効な場合の接続」トピックの Java タブを参照してください。

アプリケーションで最新の CA 証明書バンドルを使用していることを確認するには、「最新の CA バンドルを使用していることを確認するにはどうすればよいですか?」を参照してください。アプリケーションで最新の CA 証明書バンドルをすでに使用している場合は、ステップ 2 に進むことができます。

アプリケーションで CA バンドルを使用する例については、「Encrypting data in transit」および「TLS が有効な場合の接続」を参照してください。

> **Note**
>
> 現在、MongoDB Go ドライバー 1.2.1 は、`sslcertificateauthorityfile` で 1 つの CA サーバー証明書しか受け入れません。TLS が有効な場合は、Go を使用して Amazon DocumentDB に接続する方法について「TLS が有効な場合の接続」を参照してください。

ステップ 2: サーバー証明書を更新する

新しい CA バンドルを使用するようにアプリケーションを更新したら、次のステップとして Amazon DocumentDB クラスター内の各インスタンスを変更してサーバー証明書を更新します。新しいサーバー証明書を使用するようにインスタンスを変更するには、次の手順を参照してください。

Amazon DocumentDB には DB インスタンスの DB サーバー証明書に署名するために以下の CA が用意されています。

- rds-ca-ecc384-g1 — ECC 384 プライベートキーアルゴリズムと SHA384 署名アルゴリズムを備えた認証局を使用します。この CA はサーバー証明書の自動ローテーションをサポートします。これは現在、Amazon DocumentDB 4.0 および 5.0 でのみサポートされています。

- rds-ca-rsa2048-g1 — ほとんどの AWS リージョンで RSA 2048 プライベートキーアルゴリズムと SHA256 署名アルゴリズムで認証機関を使用します。この CA はサーバー証明書の自動ローテーションをサポートします。
- rds-ca-rsa2048-g1 — RSA 4096 プライベートキーアルゴリズムと SHA256 署名アルゴリズムを備えた認証局を使用します。この CA はサーバー証明書の自動ローテーションをサポートします。

> **Note**
>
> を使用している場合は AWS CLI、describe-certificates を使用して、上記の認証機関の有効性を確認できます。

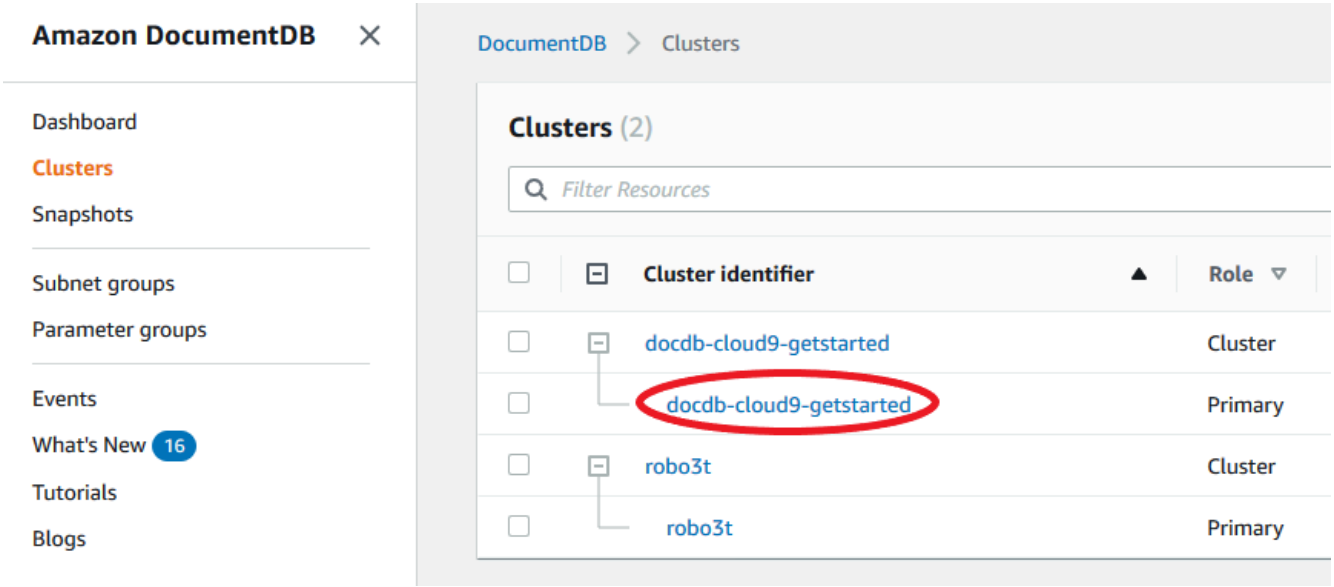
これらの CA 証明書は、地域およびグローバル証明書バンドルに含まれています。rds-ca-rsa2048-g1, rds-ca-rsa4096-g1, or rds-ca-ecc384-g1 のいずれかの CA をデータベースに使用すると、Amazon DocumentDB はそのデータベース上で DB サーバー証明書を管理します。Amazon DocumentDB は、DB サーバー証明書の有効期限が切れる前に証明書のローテーションを行います。

> **Note**
>
> お使いのクラスターが次のエンジンパッチバージョンで実行されている場合、Amazon DocumentDB は証明書ローテーションのために再起動を必要としません。
>
> - Amazon DocumentDB 3.6: 1.0.208662 以降
> - Amazon DocumentDB 4.0: 2.0.10179 以降
> - Amazon DocumentDB 5.0: 3.0.4780 以降
>
> Amazon DocumentDB エンジンパッチの最新バージョンは、次のコマンドを実行して確認できます: `db.runCommand({getEngineVersion: 1})`。
> サーバー証明書を更新する前に、ステップ 1 を完了しておく必要があります。

Using the AWS Management Console

次の手順を実行し、 AWS Management Consoleを使って既存の Amazon DocumentDB インスタンスのための古いサーバー証明書を特定するために、次のステップを完了します。

1. にサインインし AWS Management Console、「https://https://console.aws.amazon.com/docdb://www.com で Amazon DocumentDB コンソールを開きます。
2. 画面の右上隅にあるリージョンのリストで、クラスターが存在する AWS リージョン を選択します。
3. コンソールの左側のナビゲーションペインで、Clusters を選択します。
4. 古いサーバー証明書 (rds-ca-2019) に残っているインスタンスを特定する必要がある場合があります。これは [クラスター] 表の右端にある [証明機関] 列で行うことができます。
5. [クラスター] 表の左端に [クラスター識別子] の列があります。インスタンスは、以下のスクリーンショットのように、クラスターの下に表示されます。



6. 興味がある対象のインスタンスの左側にあるチェックボックスをチェックします。
7. アクション を選択してから、変更 を選択します。
8. [証明機関] で、このインスタンスの新しいサーバー証明書 (rds-ca-rsa2048-g1) を選択します。
9. 次のページで変更の概要を確認できます。接続の中断を避けるためにインスタンスを変更する前に、アプリケーションが最新の証明書 CA バンドルを使用していることを確認するよう通知する、追加のアラートがあることに注意してください。
10. 次のメンテナンス期間中に変更を適用するか、すぐに適用するかを選択できます。サーバー証明書をすぐに変更する場合は、[すぐに適用] オプションを使用します。
11. [インスタンスの変更] を選択して、更新を完了します。

Using the AWS CLI

次の手順を実行し、 AWS CLIを使って既存の Amazon DocumentDB インスタンスのための古いサーバー証明書を特定して更新します。

1. インスタンスをすぐに変更するには、クラスターの各インスタンスに対して次のコマンドを実行します。

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. クラスターの次回のメンテナンスウィンドウで新しい CA 証明書を使用するようにクラスター内のインスタンスを変更するには、クラスター内のインスタンスごとに次のコマンドを実行します。

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

サーバー証明書の自動ローテーション

Amazon DocumentDB は、サーバー証明書の自動ローテーションをサポートしています。サーバー証明書は、各クラスターインスタンスに発行されたリーフ証明書です。ルート CA 証明書とは対照的に、サーバー証明書の有効性は短く (12 か月) 、Amazon DocumentDB はユーザーのアクションなしで自動的にローテーションを処理します。Amazon DocumentDB はこの自動ローテーションに同じルート CA を使用するため、新しい CA バンドルをダウンロードする必要はありません。

> **Important**
>
> Amazon DocumentDB クラスターに接続するときは、各サーバー証明書を直接信頼するのではなく、ルート CA バンドルを信頼することをお勧めします。これにより、サーバー証明書がローテーションされた後の接続エラーを防ぐことができます。「TLS が有効な場合の接続」を参照してください。

Amazon DocumentDB は、サーバー証明書の有効期間が半分になると、任意のメンテナンスウィンドウでサーバー証明書のローテーションを試みます。新しいサーバー証明書は 12 か月間有効です。

describe-db-engine-versions コマンドを使用して SupportsCertificateRotationWithoutRestartフラグを検査し、エンジンバージョンが再起動せずに証明書をローテーションすることをサポートしているかどうかを特定します。

> **Note**
>
> クラスターが次のエンジンパッチバージョンで実行されている場合、Amazon DocumentDB は再起動せずにサーバー証明書のローテーションをサポートします。
>
> - Amazon DocumentDB 3.6: 1.0.208662 以降
> - Amazon DocumentDB 4.0: 2.0.10179 以降
> - Amazon DocumentDB 5.0: 3.0.4780 以降
>
> 現在の Amazon DocumentDB エンジンパッチバージョンを確認するには、 コマンド を実行しますdb.runCommand({getEngineVersion: 1})。
> 古いエンジンパッチバージョンを使用している場合、Amazon DocumentDB はサーバー証明書をローテーションし、任意のメンテナンスウィンドウでデータベース再起動イベントをスケジュールします。

トラブルシューティング

証明書の更新の一環としてクラスターへの接続に問題がある場合は、次の操作をお勧めします。

- クライアントが最新の証明書バンドルを使用していることを確認します。「最新の CA バンドルを使用していることを確認するにはどうすればよいですか?」を参照してください。
- インスタンスが最新の証明書を使用していることを確認します。「古い/新しいサーバー証明書を使用している Amazon DocumentDB インスタンスを確認するにはどうすればいいですか。」を参照してください。
- 最新の証明書 CA がアプリケーションで使用されていることを確認します。Java や Go などの一部のドライバーは、複数の証明書を証明書バンドルから信頼ストアにインポートするために、追加のコードを必要とします。TLS を使用した Amazon DocumentDB への接続のさらなる詳細については、プログラムによる Amazon DocumentDB への接続 を参照してください。
- サポートへのお問い合わせ。ご質問や問題がある場合は、サポート にお問い合わせください。

よくある質問

次に、TLS 証明書に関していくつかのよくある質問に対する回答を示します。

質問や問題がある場合はどうしたらいいですか?

ご質問や問題がある場合は、サポート にお問い合わせください。

TLS を使用して Amazon DocumentDB クラスターに接続しているかどうかを知るにはどうすればよいですか。

クラスターが TLS を使用しているかどうかを判断するには、クラスターのクラスターパラメータグループの `tls` パラメータを調べます。`tls` パラメータが `enabled` に設定されている場合は、TLS 証明書を使用してクラスターに接続しています。詳細については、「Amazon DocumentDB クラスターパラメータグループの管理」を参照してください。

CA 証明書とサーバー証明書を更新する理由

Amazon DocumentDB の CA 証明書とサーバー証明書は、Amazon DocumentDB の標準的なメンテナンスおよびセキュリティのベストプラクティスの一環として更新されてきました。

有効期限までにアクションしないと、どうなりますか?

TLS を使用して期限切れの CA 証明書で Amazon DocumentDB クラスターに接続している場合、TLS 経由で接続するアプリケーションは Amazon DocumentDB クラスターと通信できなくなります。

Amazon DocumentDB は、有効期限が切れる前にデータベース証明書を自動的にローテーションしません。有効期限の前後を問わず、新しい CA 証明書を使用するようにアプリケーションとクラスターを更新する必要がある。

古い/新しいサーバー証明書を使用している Amazon DocumentDB インスタンスを確認するにはどうすればいいですか。

古いサーバー証明書をまだ使用している Amazon DocumentDB インスタンスを特定するには、Amazon DocumentDB AWS Management Console または を使用できます AWS CLI。

の使用 AWS Management Console

古い証明書を使用しているクラスター内のインスタンスを特定するには

1. にサインインし AWS Management Console、「https://https://console.aws.amazon.com/docdb://www.com で Amazon DocumentDB コンソールを開きます。
2. 画面の右上隅にあるリージョンのリストで、インスタンスが存在する AWS リージョン を選択します。
3. コンソールの左側のナビゲーションペインで、Clusters を選択します。
4. [証明機関] 列 (表の右端近く) には、古いサーバー証明書 (rds-ca-2019) および新しいサーバー証明書 (rds-ca-rsa2048-g1) に残っているインスタンスが表示されます。

の使用 AWS CLI

古いサーバー証明書を使用しているクラスター内のインスタンスを特定するには、以下を指定して describe-db-clusters コマンドを使用します。

```
aws docdb describe-db-instances \
    --filters Name=engine,Values=docdb \
    --query 'DBInstances[*].
{CertificateVersion:CACertificateIdentifier,InstanceID:DBInstanceIdentifier}'
```

Amazon DocumentDB クラスター内の個々のインスタンスを変更してサーバー証明書を更新するにはどうすればよいですか。

特定のクラスター内のすべてのインスタンスのサーバー証明書を同時に更新することをお勧めします。クラスター内のインスタンスを変更するには、コンソールまたは AWS CLIを使用できます。

> **Note**
>
> サーバー証明書を更新する前に、ステップ 1 を完了しておく必要があります。

の使用 AWS Management Console

1. にサインインし AWS Management Console、「https://https://console.aws.amazon.com/docdb.com」で Amazon DocumentDB コンソールを開きます。

2. 画面の右上隅にあるリージョンのリストで、クラスターが存在する AWS リージョン を選択します。
3. コンソールの左側のナビゲーションペインで、Clusters を選択します。
4. [証明機関] 列 (表の右端近く) には、古いサーバー証明書 (rds-ca-2019) に残っているインスタンスが表示されます。
5. [クラスター] 表の [クラスター識別子] で、変更するインスタンスを選択します。
6. アクション を選択してから、変更 を選択します。
7. [証明機関] で、このインスタンスの新しいサーバー証明書 (rds-ca-rsa2048-g1) を選択します。
8. 次のページで変更の概要を確認できます。接続の中断を避けるためにインスタンスを変更する前に、アプリケーションが最新の証明書 CA バンドルを使用していることを確認するよう通知する、追加のアラートがあることに注意してください。
9. 次のメンテナンス期間中に変更を適用するか、すぐに適用するかを選択できます。
10. [インスタンスの変更] を選択して、更新を完了します。

の使用 AWS CLI

次の手順を実行し、 AWS CLIを使って既存の Amazon DocumentDB インスタンスのための古いサーバー証明書を特定して更新します。

1. インスタンスをすぐに変更するには、クラスターの各インスタンスに対して次のコマンドを実行します。

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. クラスターの次回のメンテナンスウィンドウで新しい CA 証明書を使用するようにクラスター内のインスタンスを変更するには、クラスター内のインスタンスごとに次のコマンドを実行します。

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

既存のクラスターに新しいインスタンスを追加するとどうなりますか?

作成される新しいインスタンスはすべて古いサーバー証明書を使用し、古い CA 証明書を使用する TLS 接続が必要です。2024 年 1 月 25 日以降に作成された新しい Amazon DocumentDB インスタンスでは、デフォルトで新しい証明書 rds-ca-rsa2048-g1 が使用されます。

クラスターにインスタンスの置き換えまたはフェイルオーバーがある場合はどうなりますか?

クラスターにインスタンスの置き換えがある場合、作成される新しいインスタンスは、そのインスタンスが以前使用していたものと同じサーバー証明書を引き続き使用します。すべてのインスタンスのサーバー証明書を同時に更新することをお勧めします。クラスターでフェイルオーバーが発生した場合、新しいプライマリのサーバー証明書が使用されます。

クラスターへの接続に TLS を使用していない場合でも、各インスタンスを更新する必要がありますか?

TLS を有効にすることを強くお勧めします。なお、TLS を有効にしない場合でも、今後 TLS を使用してクラスターに接続する予定がある場合は、Amazon DocumentDB インスタンス上で証明書をローテーションすることをお勧めします。Amazon DocumentDB クラスターへの接続に TLS を使用する予定がない場合、対応は必要ありません。

現在、クラスターへの接続に TLS を使用していませんが、将来的に計画している場合、どうすればよいですか。

2024 年 1 月より前にクラスターを作成した場合は、前のセクションの ステップ 1 と ステップ 2 に従って、アプリケーションで更新済みの CA バンドルを使用していること、および各 Amazon DocumentDB インスタンスで最新のサーバー証明書を使用していることを確認してください。2024 年 1 月 25 日以降にクラスターを作成した場合、クラスターにはすでに最新のサーバー証明書 (rds-ca-rsa2048-g1) があります。アプリケーションで最新の CA バンドルを使用していることを確認するには、「クラスターへの接続に TLS を使用していない場合でも、各インスタンスを更新する必要がありますか?」を参照してください。

2024 年 8 月以降に締め切りを延長することはできますか?

アプリケーションが TLS 経由で接続している場合、締め切りを延長することはできません。

最新の CA バンドルを使用していることを確認するにはどうすればよいですか?

最新のバンドルがあることを確認するには、次のコマンドを使用します。このコマンドを実行するには、Java がインストールされていて、Java ツールがシェルの PATH 変数に含まれている必要があります。詳細については、「 Java の使用」を参照してください。

macOS と Amazon Linux

```
keytool -printcert -v -file global-bundle.pem
```

Windows

```
keytool -printcert -v -file global-bundle.p7b
```

CA バンドルの名前に「RDS」が表示されるのはなぜですか?

証明書マネージメントのような特定の管理機能では、Amazon DocumentDB は Amazon RDS (Amazon Relational Database Service) と共有の運用テクノロジーを使用します。

新しい証明書の有効期限はいつですか?

新しいサーバー証明書は (一般的に) 以下のように期限切れになります。

- rds-ca-rsa2048-g1: 2061 年
- rds-ca-rsa4096-g1: 2121 年
- rds-ca-ecc384-g1: 2121 年 に有効期限 となります

証明書の有効期限が切れる前にアクションを実行しないと、どのようなエラーが表示されますか ?

エラーメッセージはお使いのドライバーによって異なります。一般に、「証明書の有効期限切れです」の文字列を含む証明書検証エラーが表示されます。

新しいサーバー証明書を適用した場合、古いサーバー証明書に戻すことはできますか。

インスタンスを古いサーバー証明書に戻す必要がある場合は、クラスターのすべてのインスタンスに対してこの操作を実行することをお勧めします。 AWS Management Console または を使用して、クラスター内の各インスタンスのサーバー証明書を元に戻すことができます AWS CLI。

の使用 AWS Management Console

1. にサインインし AWS Management Console、「https://https://console.aws.amazon.com/docdb.com」で Amazon DocumentDB コンソールを開きます。
2. 画面の右上隅にあるリージョンのリストで、クラスターが存在する AWS リージョン を選択します。
3. コンソールの左側のナビゲーションペインで、Clusters を選択します。
4. [クラスター] 表の [クラスター識別子] で、変更するインスタンスを選択します。[アクション] を選択してから、[変更] を選択します。
5. [証明機関] で、古いサーバー証明書を選択します (`rds-ca-2019`)。
6. [続行] を選択して、変更の概要を表示します。
7. この結果ページでは、変更を次のメンテナンスウィンドウで適用するようにスケジュールするか、変更をすぐに適用するかを選択できます。選択を行い、[Modify instance (インスタンスの変更)] を選択します。

> **Note**
>
> 変更の即時適用を選択した場合、保留中の変更キューにあるすべての変更も同様に適用されます。ダウンタイムを必要とする保留中の変更がある場合、このオプションを選択すると予想外のダウンタイムが発生することがあります。

の使用 AWS CLI

```
aws docdb modify-db-instance --db-instance-identifier <db_instance_name> ca-
certificate-identifier rds-ca-2019 <--apply-immediately | --no-apply-immediately>
```

`--no-apply-immediately` を選択した場合、変更はクラスターの次のメンテナンス期間中に適用されます。

スナップショットまたはポイントインタイム復元から復元した場合、新しいサーバー証明書が含まれていますか?

2024 年 8 月以降にスナップショットを復元するか、ポイントインタイム復元を実行すると、作成される新しいクラスターでは新しい CA 証明書が使用されます。

どれかの Mac OS から Amazon DocumentDB クラスターに直接接続する際に問題が発生した場合はどうすればいいですか。

Mac OS は、信頼できる証明書の要件を更新しています。信頼された証明書の有効期間は 397 日以下である必要があります (https://support.apple.com/en-us/HT211025 を参照)。

> **Note**
>
> この制限は Mac OS の新しいバージョンでも見られます。

Amazon DocumentDB インスタンス証明書の有効期間は 4 年以上であり、Mac OS の上限を超えています。Mac OS を実行しているコンピュータから Amazon DocumentDB クラスターに直接接続するには、TLS 接続の作成時に無効な証明書を許可する必要があります。この場合、無効な証明書とは、有効期間が 397 日より長いものを意味します。Amazon DocumentDB クラスターに接続する場合、無効な証明書を許可する前に、そのリスクを理解する必要があります。

を使用して Mac OS から Amazon DocumentDB クラスターに接続するには AWS CLI、`tlsAllowInvalidCertificates`パラメータを使用します。

```
mongo --tls --host <hostname> --username <username> --password <password> --port 27017
 --tlsAllowInvalidCertificates
```

Amazon DocumentDB TLS 証明書の更新 - GovCloud

トピック

- アプリケーションと Amazon DocumentDB クラスターの更新
- トラブルシューティング
- よくある質問

> **Note**
>
> この情報は、GovCloud (米国西部) と GovCloud (米国東部) リージョンのユーザーのみに適用されます。

Amazon DocumentDB (MongoDB 互換) クラスターの証明機関 (CA) 証明書は、2022 年 5 月 18 日に更新されました。TLS (Transport Layer Security) が有効 (デフォルト設定) になっている クラスターを使用しており、クライアントアプリケーション証明書とサーバー証明書を更新していない場合は、アプリケーションと Amazon DocumentDB クラスターとの接続問題を軽減するために次の手順を実行する必要があります。

- ステップ 1: 新しい CA 証明書をダウンロードしてアプリケーションを更新する
- ステップ 2: サーバー証明書を更新する

CA とサーバーの証明書は、Amazon DocumentDB のための標準的なメンテナンスおよびセキュリティのベストプラクティスの一環として更新されています。以前の CA 証明書が 2022 年 5 月 18 日に失効します。クライアントアプリケーションは、新しい CA 証明書をトラストストアに追加する必要があり、この有効期限より前に新しい CA 証明書を使用するために既存の Amazon DocumentDB インスタンスを更新する必要があります。

アプリケーションと Amazon DocumentDB クラスターの更新

このセクションの手順に従って、アプリケーションの CA 証明書バンドル (ステップ 1) とクラスターのサーバー証明書 (ステップ 2) を更新します。本番環境に変更を適用する前に、開発環境またはステージング環境でこれらの手順をテストすることを強くお勧めします。

> **Note**
>
> Amazon DocumentDB クラスターがある各 AWS リージョン で、ステップ 1 と 2 を完了する必要があります。

ステップ 1: 新しい CA 証明書をダウンロードしてアプリケーションを更新する

新しい CA 証明書を使用して Amazon DocumentDB への TLS 接続を作成できるように、新しい CA 証明書をダウンロードしてアプリケーションを更新します。

- GovCloud (米国西部) の場合は、https://truststore.pki.us-gov-west-1.rds.amazonaws.com/us-gov-west-1/us-gov-west-1-bundle.pem から新しい CA 証明書バンドルをダウンロードします。このオペレーションにより、`us-gov-west-1-bundle.pem` という名前のファイルがダウンロードされます。
- GovCloud (米国東部) の場合は、https://truststore.pki.us-gov-west-1.rds.amazonaws.com/us-gov-east-1/us-gov-east-1-bundle.pem から新しい CA 証明書バンドルをダウンロードします。このオペ

レーションにより、us-gov-east-1-bundle.pem という名前のファイルがダウンロードされます。

> **Note**
>
> 古い CA 証明書 (rds-ca-2017-root.pem) と新しい CA 証明書 (rds-ca-rsa2048-g1.pem、rds-ca-rsa4096-g1.pem、rds-ca-ecc384-g1.pem のいずれか) の両方を含むキーストアにアクセスしている場合は、そのキーストアで使用したい証明書が選択されるかを確認してください。各証明書の詳細については、以下のステップ 2 を参照してください。

```
wget https://truststore.pki.us-gov-west-1.rds.amazonaws.com/us-gov-west-1/us-gov-west-1-bundle.pem
```

```
wget https://truststore.pki.us-gov-west-1.rds.amazonaws.com/us-gov-east-1/us-gov-east-1-bundle.pem
```

次に、新しい証明書バンドルを使用するようにアプリケーションを更新します。新しい CA バンドルには、古い CA 証明書と新しい CA 証明書 (rds-ca-rsa2048-g1.pem、rds-ca-rsa4096-g1.pem、rds-ca-ecc384-g1.pem のいずれか) の両方が含まれます。新しい CA バンドルに両方の CA 証明書を使用すると、2 つの手順でアプリケーションとクラスターを更新できます。

2021 年 12 月 21 日以降に CA 証明書バンドルをダウンロードする場合は、新しい CA 証明書バンドルを使用する必要があります。アプリケーションで最新の CA 証明書バンドルを使用していることを確認するには、「最新の CA バンドルを使用していることを確認するにはどうすればよいですか?」を参照してください。アプリケーションで最新の CA 証明書バンドルをすでに使用している場合は、ステップ 2 に進むことができます。

アプリケーションで CA バンドルを使用する例については、「Encrypting data in transit」および「TLS が有効な場合の接続」を参照してください。

> **Note**
>
> 現在、MongoDB Go ドライバー 1.2.1 は、sslcertificateauthorityfile で 1 つの CA サーバー証明書しか受け入れません。TLS が有効な場合は、Go を使用して Amazon DocumentDB に接続する方法について「TLS が有効な場合の接続」を参照してください。

ステップ 2: サーバー証明書を更新する

新しい CA バンドルを使用するようにアプリケーションを更新したら、次のステップとして Amazon DocumentDB クラスター内の各インスタンスを変更してサーバー証明書を更新します。新しいサーバー証明書を使用するようにインスタンスを変更するには、次の手順を参照してください。

Amazon DocumentDB には DB インスタンスの DB サーバー証明書に署名するために以下の CA が用意されています。

- rds-ca-ecc384-g1 — ECC 384 プライベートキーアルゴリズムと SHA384 署名アルゴリズムを備えた認証局を使用します。この CA はサーバー証明書の自動ローテーションをサポートします。これは現在、Amazon DocumentDB 4.0 および 5.0 でのみサポートされています。
- rds-ca-rsa2048-g1 — ほとんどの AWS リージョンで、RSA 2048 プライベートキーアルゴリズムと SHA256 署名アルゴリズムを備えた認証局を使用します。この CA はサーバー証明書の自動ローテーションをサポートします。
- rds-ca-rsa2048-g1 — RSA 4096 プライベートキーアルゴリズムと SHA256 署名アルゴリズムを備えた認証局を使用します。この CA はサーバー証明書の自動ローテーションをサポートします。

> **Note**
>
> を使用している場合は AWS CLI、describe-certificates を使用して、上記の認証機関の有効性を確認できます。

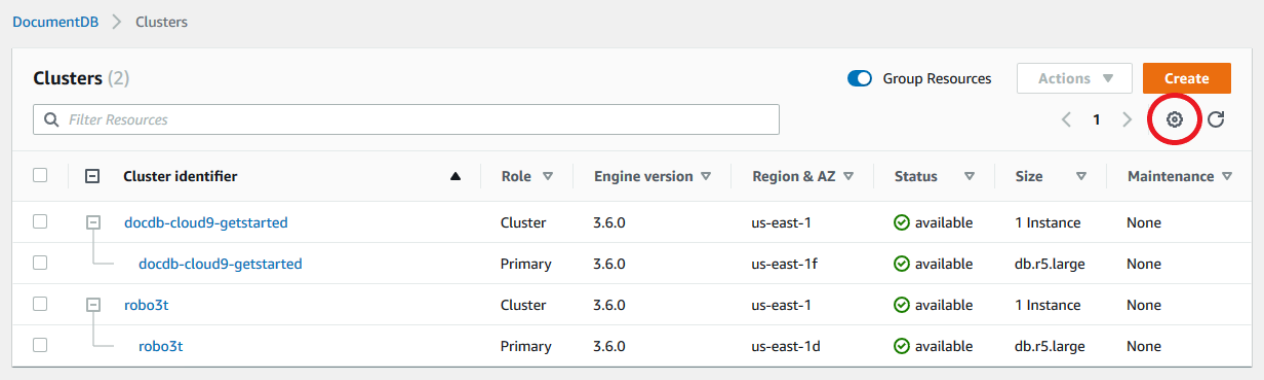
> **Note**
>
> Amazon DocumentDB 4.0 および 5.0 インスタンスの場合、再起動は必要ありません。Amazon DocumentDB 3.6 インスタンスを更新するには再起動が必要であり、そのためにサービスが中断される場合があります。サーバー証明書を更新する前に、ステップ 1 を完了しておく必要があります。

Using the AWS Management Console

次の手順を実行し、 AWS Management Consoleを使って既存の Amazon DocumentDB インスタンスのための古いサーバー証明書を特定するために、次のステップを完了します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. 画面の右上隅にあるリージョンのリストで、クラスター AWS リージョン が存在する を選択します。
3. コンソールの左側のナビゲーションペインで、Clusters を選択します。
4. 古いサーバー証明書 (`rds-ca-2017`) に残っているインスタンスを特定する必要がある場合があります。デフォルトでは非表示になっている列 Certificate authority でこれを行えます。[証明機関] 列を表示するには、次の手順を実行します。

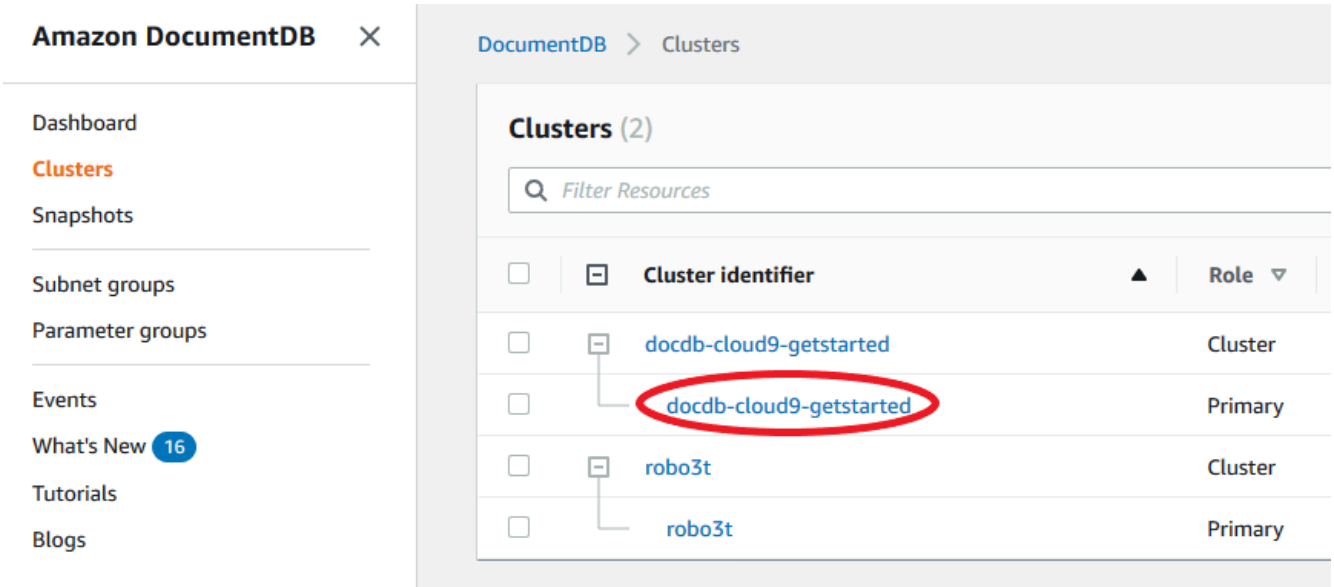
   a. [設定] アイコンを選択します。



   b. 表示されている列の一覧で、[証明機関] 列を選択します。

   c. [確定] を選択し、変更を保存します。

5. Clusters ナビゲーションボックスに戻り、Cluster Identifier 列が表示されます。インスタンスは、以下のスクリーンショットのように、クラスターの下に表示されます。



6. 興味がある対象のインスタンスの左側にあるチェックボックスをチェックします。
7. アクション を選択してから、変更 を選択します。
8. [証明機関] で、このインスタンス向けの新しいサーバー証明書 (rds-ca-rsa2048-g1、rds-ca-rsa4096-g1、rds-ca-ecc384-g1 のいずれか) を選択します。
9. 次のページで変更の概要を確認できます。接続の中断を避けるためにインスタンスを変更する前に、アプリケーションが最新の証明書 CA バンドルを使用していることを確認するよう通知する、追加のアラートがあることに注意してください。
10. 次のメンテナンス期間中に変更を適用するか、すぐに適用するかを選択できます。サーバー証明書をすぐに変更する場合は、[すぐに適用] オプションを使用します。
11. [インスタンスの変更] を選択して、更新を完了します。

Using the AWS CLI

次の手順を実行し、 AWS CLIを使って既存の Amazon DocumentDB インスタンスのための古いサーバー証明書を特定して更新します。

1. インスタンスをすぐに変更するには、クラスターの各インスタンスに対して次のコマンドを実行します。次のいずれかの証明書を使用します: rds-ca-rsa2048-g1、rds-ca-rsa4096-g1、rds-ca-ecc384-g1。

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa4096-g1 --apply-immediately
```

2. クラスターの次回のメンテナンスウィンドウで新しい CA 証明書を使用するようにクラスター内のインスタンスを変更するには、クラスター内のインスタンスごとに次のコマンドを実行します。次のいずれかの証明書を使用します: rds-ca-rsa2048-g1、rds-ca-rsa4096-g1、rds-ca-ecc384-g1。

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa4096-g1 --no-apply-immediately
```

トラブルシューティング

証明書の更新の一環としてクラスターへの接続に問題がある場合は、次の操作をお勧めします。

- インスタンスを再起動します。新しい証明書に更新するには、各インスタンスを再起動する必要があります。新しい証明書を 1 つ以上のインスタンスに適用しても再起動していない場合は、インスタンスを再起動して新しい証明書を適用します。詳細については、「Amazon DocumentDB インスタンスの再起動」を参照してください。
- クライアントが最新の証明書バンドルを使用していることを確認します。「最新の CA バンドルを使用していることを確認するにはどうすればよいですか?」を参照してください。
- インスタンスが最新の証明書を使用していることを確認します。「古い/新しいサーバー証明書を使用している Amazon DocumentDB インスタンスを確認するにはどうすればいいですか。」を参照してください。
- 最新の証明書 CA がアプリケーションで使用されていることを確認します。Java や Go などの一部のドライバーは、複数の証明書を証明書バンドルから信頼ストアにインポートするために、追加のコードを必要とします。TLS を使用した Amazon DocumentDB への接続のさらなる詳細については、プログラムによる Amazon DocumentDB への接続 を参照してください。
- サポートへのお問い合わせ。ご質問や問題がある場合は、サポート にお問い合わせください。

よくある質問

次に、TLS 証明書に関していくつかのよくある質問に対する回答を示します。

質問や問題がある場合はどうしたらいいですか?

ご質問や問題がある場合は、サポート にお問い合わせください。

TLS を使用して Amazon DocumentDB クラスターに接続しているかどうかを知るにはどうすればよいですか。

クラスターが TLS を使用しているかどうかを判断するには、クラスターのクラスターパラメータグループの `tls` パラメータを調べます。`tls` パラメータが `enabled` に設定されている場合は、TLS 証明書を使用してクラスターに接続しています。詳細については、「Amazon DocumentDB クラスターパラメータグループの管理」を参照してください。

CA 証明書とサーバー証明書を更新する理由

Amazon DocumentDB の CA 証明書とサーバー証明書は、Amazon DocumentDB の標準的なメンテナンスおよびセキュリティのベストプラクティスの一環として更新されました。現在の CA 証明書とサーバー証明書は 2022 年 5 月 18 日 (水) に有効期限が切れます。

有効期限までにアクションしないと、どうなりますか?

Amazon DocumentDB クラスターへの接続に TLS を使用しており、2022 年 5 月 18 日までに変更を行わなかった場合、TLS 経由で接続するアプリケーションは Amazon DocumentDB クラスターと通信できなくなります。

Amazon DocumentDB は、有効期限が切れる前にデータベース証明書を自動的にローテーションしません。有効期限の前後を問わず、新しい CA 証明書を使用するようにアプリケーションとクラスターを更新する必要がある。

古い/新しいサーバー証明書を使用している Amazon DocumentDB インスタンスを確認するにはどうすればいいですか。

古いサーバー証明書をまだ使用している Amazon DocumentDB インスタンスを特定するには、Amazon DocumentDB AWS Management Console または を使用できます AWS CLI。

の使用 AWS Management Console

古い証明書を使用しているクラスター内のインスタンスを特定するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. 画面の右上隅にあるリージョンのリストで、インスタンスが存在する AWS リージョン を選択します。
3. コンソールの左側のナビゲーションペインで、[インスタンス] を選択します。
4. [証明機関] 列 (デフォルトでは非表示) で、どのインスタンスがまだ古いサーバー証明書 (`rds-ca-2017`) を使用しており、どのインスタンスに新しいサーバー証明書 (`rds-ca-rsa2048-g1`、`rds-ca-rsa4096-g1`、`rds-ca-ecc384-g1` のいずれか) が適用されているかを確認できます。[証明機関] 列を表示するには、次の手順を実行します。
   a. [設定] アイコンを選択します。
   b. 表示されている列の一覧で、[証明機関] 列を選択します。
   c. [確定] を選択し、変更を保存します。

の使用 AWS CLI

古いサーバー証明書を使用しているクラスター内のインスタンスを特定するには、以下を指定して `describe-db-clusters` コマンドを使用します。

```
aws docdb describe-db-instances \
    --filters Name=engine,Values=docdb \
    --query 'DBInstances[*].
{CertificateVersion:CACertificateIdentifier,InstanceID:DBInstanceIdentifier}'
```

Amazon DocumentDB クラスター内の個々のインスタンスを変更してサーバー証明書を更新するにはどうすればよいですか。

特定のクラスター内のすべてのインスタンスのサーバー証明書を同時に更新することをお勧めします。クラスター内のインスタンスを変更するには、コンソールまたは AWS CLIを使用できます。

> **Note**
>
> インスタンスを更新するには再起動が必要であり、そのためにサービスが中断される場合があります。サーバー証明書を更新する前に、ステップ 1 を完了しておく必要があります。

の使用 AWS Management Console

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. 画面の右上隅にあるリージョンのリストで、クラスター AWS リージョン が存在する を選択します。
3. コンソールの左側のナビゲーションペインで、[インスタンス] を選択します。
4. [証明機関] 列 (デフォルトでは非表示) に、古いサーバー証明書に残っているインスタンスが表示されます (rds-ca-2017)。[証明機関] 列を表示するには、次の手順を実行します。
   a. [設定] アイコンを選択します。
   b. 表示されている列の一覧で、[証明機関] 列を選択します。
   c. [確定] を選択し、変更を保存します。
5. 変更するインスタンスを選択します。
6. アクション を選択してから、変更 を選択します。
7. [証明機関] で、このインスタンス向けの新しいサーバー証明書を一つ (rds-ca-rsa2048-g1、rds-ca-rsa4096-g1、rds-ca-ecc384-g1 のいずれか) 選択します。

8. 次のページで変更の概要を確認できます。接続の中断を避けるためにインスタンスを変更する前に、アプリケーションが最新の証明書 CA バンドルを使用していることを確認するよう通知する、追加のアラートがあることに注意してください。
9. 次のメンテナンス期間中に変更を適用するか、すぐに適用するかを選択できます。
10. [インスタンスの変更] を選択して、更新を完了します。

の使用 AWS CLI

次の手順を実行し、 AWS CLIを使って既存の Amazon DocumentDB インスタンスのための古いサーバー証明書を特定して更新します。

1. インスタンスをすぐに変更するには、クラスターの各インスタンスに対して次のコマンドを実行します。

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa4096-g1 --apply-immediately
```

2. クラスターの次回のメンテナンスウィンドウで新しい CA 証明書を使用するようにクラスター内のインスタンスを変更するには、クラスター内のインスタンスごとに次のコマンドを実行します。

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa4096-g1 --no-apply-immediately
```

既存のクラスターに新しいインスタンスを追加するとどうなりますか?

作成される新しいインスタンスはすべて古いサーバー証明書を使用し、古い CA 証明書を使用する TLS 接続が必要です。2022 年 3 月 21 日以降に新規作成される Amazon DocumentDB インスタンスは、デフォルトで新しい証明書を使用します。

クラスターにインスタンスの置き換えまたはフェイルオーバーがある場合はどうなりますか?

クラスターにインスタンスの置き換えがある場合、作成される新しいインスタンスは、そのインスタンスが以前使用していたものと同じサーバー証明書を引き続き使用します。すべてのインスタンスのサーバー証明書を同時に更新することをお勧めします。クラスターでフェイルオーバーが発生した場合、新しいプライマリのサーバー証明書が使用されます。

クラスターへの接続に TLS を使用していない場合でも、各インスタンスを更新する必要がありますか?

Amazon DocumentDB クラスターへの接続に TLS を使用していない場合、操作は必要ありません。

現在、クラスターへの接続に TLS を使用していませんが、将来的に計画している場合、どうすればよいですか。

2022 年 3 月 21 日より前にクラスターを作成した場合は、前のセクションの ステップ 1 と ステップ 2 に従って、アプリケーションで更新済みの CA バンドルを使用していること、および各 Amazon DocumentDB インスタンスで最新のサーバー証明書を使用していることを確認します。2022 年 3 月 21 日以降にクラスターを作成した場合、クラスターにはすでに最新のサーバー証明書があります。アプリケーションで最新の CA バンドルを使用していることを確認するには、「クラスターへの接続に TLS を使用していない場合でも、各インスタンスを更新する必要がありますか?」を参照してください。

2022 年 5 月 18 日以降に締め切りを延長することはできますか?

アプリケーションが TLS 経由で接続している場合、2022 年 3 月 18 日以降に締め切りを延長することはできません。

最新の CA バンドルを使用していることを確認するにはどうすればよいですか?

互換性の理由から、古い CA バンドルファイルと新しい CA バンドルファイルの両方に `us-gov-west-1-bundle.pem` という名前が付けられます。また、`openssl` や `keytool` などのツールを使用して CA バンドルを検査することもできます。

CA バンドルの名前に「RDS」が表示されるのはなぜですか?

証明書マネージメントのような特定の管理機能では、Amazon DocumentDB は Amazon RDS (Amazon Relational Database Service) と共有の運用テクノロジーを使用します。

新しい証明書の有効期限はいつですか?

新しいサーバー証明書は (一般的に) 以下のように期限切れになります。

- rds-ca-rsa2048-g1: 2061 年
- rds-ca-rsa4096-g1: 2121 年
- rds-ca-ecc384-g1: 2121 年 に有効期限 となります

証明書の有効期限が切れる前にアクションを実行しないと、どのようなエラーが表示されますか？

エラーメッセージはお使いのドライバーによって異なります。一般に、「証明書の有効期限切れです」の文字列を含む証明書検証エラーが表示されます。

新しいサーバー証明書を適用した場合、古いサーバー証明書に戻すことはできますか。

インスタンスを古いサーバー証明書に戻す必要がある場合は、クラスターのすべてのインスタンスに対してこの操作を実行することをお勧めします。 AWS Management Console または を使用して、クラスター内の各インスタンスのサーバー証明書を元に戻すことができます AWS CLI。

の使用 AWS Management Console

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. 画面の右上隅にあるリージョンのリストで、クラスター AWS リージョン が存在する を選択します。
3. コンソールの左側のナビゲーションペインで、[インスタンス] を選択します。
4. 変更するインスタンスを選択します。[アクション] を選択してから、[変更] を選択します。
5. [証明機関] で、古いサーバー証明書(`rds-ca-2017`)を選択することができます。
6. [続行] を選択して、変更の概要を表示します。
7. この結果ページでは、変更を次のメンテナンスウィンドウで適用するようにスケジュールするか、変更をすぐに適用するかを選択できます。選択を行い、[Modify instance (インスタンスの変更)] を選択します。

> **Note**
>
> 変更の即時適用を選択した場合、保留中の変更キューにあるすべての変更も同様に適用されます。ダウンタイムを必要とする保留中の変更がある場合、このオプションを選択すると予想外のダウンタイムが発生することがあります。

の使用 AWS CLI

```
aws docdb modify-db-instance --db-instance-identifier <db_instance_name> ca-
certificate-identifier rds-ca-2017 <--apply-immediately | --no-apply-immediately>
```

--no-apply-immediately を選択した場合、変更はクラスターの次のメンテナンス期間中に適用されます。

スナップショットまたはポイントインタイム復元から復元した場合、新しいサーバー証明書が含まれていますか?

2022 年 3 月 21 日以降にスナップショットを復元するか、ポイントインタイム復元を実行すると、作成される新しいクラスターでは新しい CA 証明書が使用されます。

Mac OS X Catalina から Amazon DocumentDB クラスターに直接接続する際に問題が発生した場合はどうすればいいですか。

Mac OS X Catalina は、信頼できる証明書の要件を更新しています。信頼された証明書の有効期間は 825 日以下である必要があります (https://support.apple.com/en-us/HT210176 を参照)。Amazon DocumentDB インスタンス証明書の有効期間は 4 年以上であり、Mac OS X の上限を超えています。Mac OS X Catalina を実行しているコンピュータから Amazon DocumentDB クラスターに直接接続するには、TLS 接続の作成時に無効な証明書を許可する必要があります。この場合、無効な証明書とは、有効期間が 825 日より長いものを意味します。Amazon DocumentDB クラスターに接続する場合、無効な証明書を許可する前に、そのリスクを理解する必要があります。

を使用して OS X Catalina から Amazon DocumentDB クラスターに接続するには AWS CLI、tlsAllowInvalidCertificatesパラメータを使用します。

```
mongo --tls --host <hostname> --username <username> --password <password> --port 27017
 --tlsAllowInvalidCertificates
```

Amazon DocumentDB のコンプライアンス検証

Amazon DocumentDB のセキュリティおよびコンプライアンスは、以下を含む複数の AWS コンプライアンスプログラムの一環として、サードパーティーの監査者により評価されます。

- 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)。詳細については、「コンプライアンスプログラムによる対象範囲内のAWS サービス」を参照してください。
- 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。
- ISO 9001、27001、27017、および 27018。詳細については、「ISO 認定」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 への準拠」を参照してください。

AWS は、コンプライアンスプログラムによる対象範囲内 AWS のサービスで、特定のコンプライアンスAWS プログラムの対象となるサービスの頻繁に更新されるリストを提供します。

サードパーティーの監査レポートは、 を使用してダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

AWS コンプライアンスプログラムの詳細については、AWS 「 コンプライアンスプログラム」を参照してください。

Amazon DocumentDB を使用する際のお客様のコンプライアンス責任は、組織のデータの機密性や組織のコンプライアンス目的、適用可能な法律、規制によって決定されます。Amazon DocumentDB の使用が HIPAA や PCI などの標準に準拠していることを前提としている場合、 は以下に役立つリソース AWS を提供します。

- AWS コンプライアンスリソース – お客様の業界や地域に適用される可能性のあるワークブックとガイドのコレクション。
- セキュリティとコンプライアンスのクイックスタートガイド – 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デプロイする手順を提供するデプロイガイド AWS。
- AWS Config – リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価するサービス。
- AWS Security Hub – セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立つ AWS 、 内のセキュリティ状態の包括的なビュー。
- 「Architecting for HIPAA Security and Compliance」ホワイトペーパー – 企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法を説明するホワイトペーパー。

Amazon DocumentDB の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。 は、低レイテンシー、高スループット、および高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティーゾーン AWS リージョンを提供します。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。

Amazon DocumentDB クラスターは、最低 2 つのアベイラビリティーゾーンに最低 2 つのサブネットがある Amazon VPC 内にのみ作成できます。少なくとも 2 つのアベイラビリティーゾーンにまたがってクラスターインスタンスを配布することで、万一 1 つのアベイラビリティーゾーンに障害が発生した場合でも、Amazon DocumentDB では、クラスター内でインスタンスを使用できます。Amazon DocumentDB クラスターのクラスターボリュームは、データ損失の可能性が少ない耐久性のあるストレージを提供するために、常に 3 つのアベイラビリティーゾーンにまたがっています。

AWS リージョン およびアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

Amazon DocumentDB は、 AWS グローバルインフラストラクチャに加えて、データの耐障害性とバックアップのニーズをサポートするのに役立ついくつかの機能を提供しています。

耐障害性と自己修復機能を備えたストレージ

ストレージボリュームの 10 GB ごとに、3 つのアベイラビリティーゾーン間で 6 つの方法でレプリケートされます。Amazon DocumentDB は、データベースの書き込み可用性に影響を与えずに最大 2 つのデータの損失を透過的に処理するフォールトトレラントストレージを使用し、読み取り可用性に影響を与えずに最大 3 つのコピーを透過的に処理します。Amazon DocumentDB ストレージも自己修復機能であり、データブロックとディスクは継続的にエラーをスキャンし、自動的に交換されます。

手動バックアップと復元

Amazon DocumentDB は、長期的な保持と回復のためにクラスターの完全バックアップを作成する機能を提供します。詳細については、「Amazon DocumentDB でのバックアップと復元」を参照してください。

ポイントインタイムリカバリ

ポイントインタイムリカバリを使用することで、Amazon DocumentDB クラスターが誤って書き込みされたり削除されたりしないようにできます。ポイントインタイムリカバリを有効化すれば、オンデマンドバックアップの作成、維持、スケジュールを心配する必要はありません。詳細については、「特定時点への復元」を参照してください。

Amazon DocumentDB 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 Amazon DocumentDB は グローバル AWS ネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「セキュリティの柱 AWS Well-Architected フレームワーク」の「インフラストラクチャの保護」を参照してください。

AWS が公開した API コールを使用して、ネットワーク経由で Amazon DocumentDB にアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードはJava 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。またはAWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

これらの API オペレーションは任意のネットワークの場所から呼び出すことができます。Amazon DocumentDB ポリシーを使用して、特定の Amazon Virtual Private Cloud (Amazon VPC) エンドポイントまたは特定の VPC からのアクセスを制御することができます。これにより、実質的にネットワーク内の特定の VPC からのみ、特定の Amazon DocumentDB リソースへの AWS ネットワークアクセスが分離されます。

> **Note**
>
> Amazon DocumentDB では、リソースベースのポリシーは使用できません。

Amazon DocumentDB API とインターフェイス VPC エンドポイント (AWS PrivateLink) について

> **Note**
>
> Amazon DocumentDB エラスティッククラスターは AWS PrivateLink VPC エンドポイントをサポートしていません。

インターフェイス VPC エンドポイントを作成することで、VPC と Amazon DocumentDB 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 を使用します AWS PrivateLink。

Amazon DocumentDB インスタンスベースのクラスターはインターフェイス VPC エンドポイント接続を必要としませんが、 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 Amazon DocumentDB 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 内の Amazon DocumentDB インスタンスは、データベースインスタンス やデータベースクラスターの起動、変更、または終了のための Amazon DocumentDB API エンドポイントとの通信に、パブリック IP アドレスを必要としません。また、Amazon DocumentDB インスタンスは、利用可能な Amazon DocumentDB API オペレーション全ての使用にパブリック IP アドレスを必要としません。VPC と Amazon DocumentDB 間のトラフィックは、すべて Amazon ネットワーク内で行われます。

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。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 Virtual Private Cloud (AWS PrivateLink) 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 のサービス を使用して にアクセスする」を参照してください。Amazon DocumentDB オペレーションの詳細については、「Amazon DocumentDB クラスター、インスタンス、およびリソース管理 API リファレンス」を参照してください。

トピック

- VPC エンドポイントに関する考慮事項
- 利用可能なリージョン
- Amazon DocumentDB API 用のインターフェイス VPC エンドポイントの作成

- Amazon DocumentDB API 向け VPC エンドポイントポリシーの作成

VPC エンドポイントに関する考慮事項

Amazon DocumentDB API エンドポイント向けにインターフェイス VPC エンドポイントを設定する前に、「Amazon Virtual Private Cloud (AWS PrivateLink) ユーザーガイド」の「インターフェイスエンドポイントの前提条件」を確認してください。

Amazon DocumentDB リソースの管理に関連するすべての Amazon DocumentDB API オペレーションは、 AWS PrivateLinkを使用して VPC から利用することができます。

VPC エンドポイントポリシーは Amazon DocumentDB 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 Amazon DocumentDB API オペレーションへのフルアクセスが提供されます。詳細については、「Amazon Virtual Private Cloud (AWS PrivateLink) ユーザーガイド」の「エンドポイントポリシーを使用して VPC エンドポイントへのアクセスを制御する」を参照してください。

利用可能なリージョン

Amazon DocumentDB API は現在、以下の VPC エンドポイントをサポートしています AWS リージョン。

- 米国東部(オハイオ)
- 米国東部 (バージニア北部)
- 米国西部 (オレゴン)
- アフリカ (ケープタウン)
- アジアパシフィック (香港)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (ハイデラバード)
- アジアパシフィック (大阪)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)

- カナダ (中部)
- 中国 (北京)
- 中国 (寧夏)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (パリ)
- 欧州 (スペイン)
- 欧州 (ミラノ)
- 中東 (UAE)
- 南米 (サンパウロ)
- AWS GovCloud (米国東部)
- AWS GovCloud (米国西部)

Amazon DocumentDB API 用のインターフェイス VPC エンドポイントの作成

Amazon DocumentDB API の VPC エンドポイントを作成できますAWS CLI。 AWS Command Line Interface 詳細については、Amazon Virtual Private Cloud (AWS PrivateLink) 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 のサービス を使用して にアクセスする」を参照してください。

サービス名 `com.amazonaws.region.rds` を使用して、Amazon DocumentDB API 向け VPC エンドポイントを作成します。

AWS リージョン 中国を除き、エンドポイントのプライベート DNS を有効にすると、rds.us-east-1.amazonaws.com AWS リージョンなどの のデフォルト DNS 名を使用して、VPC エンドポイントで Amazon DocumentDB への API リクエストを行うことができます。中国 (北京) および中国 (寧夏) では AWS リージョン、それぞれ rds-api.cn-north-1.amazonaws.com.cn と rds-api.cn-northwest-1.amazonaws.com.cn を使用して VPC エンドポイントで API リクエストを行うことができます。

詳細については、Amazon Virtual Private Cloud (AWS PrivateLink) 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 のサービス を使用して にアクセスする」を参照してください。

Amazon DocumentDB API 向け VPC エンドポイントポリシーの作成

Amazon DocumentDB API へのアクセスを制御するエンドポイントポリシーを VPC エンドポイントにアタッチできます。このポリシーでは、以下の情報を指定します。

- 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。
- アクションを実行できるリソース。

詳細については、「Amazon Virtual Private Cloud (AWS PrivateLink) ユーザーガイド」の「エンドポイントポリシーを使用して VPC エンドポイントへのアクセスを制御する」を参照してください。

例: Amazon DocumentDB API アクション向け VPC エンドポイントポリシー

Amazon DocumentDB API 向けエンドポイントポリシーの例を次に示します。このポリシーをエンドポイントにアタッチすると、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 Amazon DocumentDB API アクションへのアクセスを許可します。

```
{
"Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "docdb:CreateDBInstance",
            "docdb:ModifyDBInstance",
            "docdb:CreateDBSnapshot"
         ],
         "Resource":"*"
      }
   ]
}
```

例: 指定された 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して 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" ] }
     }
   ]
}
```

Amazon DocumentDB のセキュリティベストプラクティス

セキュリティのベストプラクティスとして、Amazon DocumentDB API オペレーション、特に Amazon DocumentDB リソースを作成、変更、削除するオペレーションへのアクセスを制御するには、 AWS Identity and Access Management (IAM) アカウントを使用する必要があります。 Amazon DocumentDB そのようなリソースには、クラスター、セキュリティグループ、およびパラメータグループなどがあります。また、IAMを使用して、クラスターのバックアップや復元など、一般的な管理操作を実行するアクションも制御します。IAM ロールを作成する際には、最小特権の原則を採択します。

- ロールベースのアクセスコントロールを使用して、最小特権を適用します。
- Amazon DocumentDB のリソースを管理する各ユーザーにそれぞれ IAM アカウントを割り当てます。Amazon DocumentDB リソースの管理に AWS アカウント ルートユーザーを使用しないでください。お客様を含めて全員に IAM ユーザーを作成します。
- それぞれの職務の実行に最低限必要になる一連のアクセス許可を各 ユーザーに付与します。
- IAM グループを使用して、複数のユーザーのアクセス許可を効果的に管理します。IAM の詳細については、IAM ユーザーガイドを参照してください。IAM のベストプラクティスの詳細については、「IAM のベストプラクティス」を参照してください。
- IAM 認証情報のローテーションを定期的に行います。
- Amazon DocumentDB の AWS シークレットを自動的にローテーションするように Secrets Manager を設定します。詳細については、AWS 「 Secrets Manager ユーザーガイド」の

「Secrets Manager シークレットのローテーション」およびAmazon DocumentDB のシークレットのローテーション」を参照してください。 AWS

- Transport Layer Security (TLS) と保管時の暗号化を使用してデータを暗号化します。

Amazon DocumentDB イベントの監査

Amazon DocumentDB (MongoDB の互換性あり) を使用すると、クラスター内で実行されたイベントを監査できます。ログに記録されるイベントの例としては、認証の成功と失敗、データベース内のコレクションの削除、インデックスの作成などがあります。Amazon DocumentDB ではデフォルトで監査が無効になっているため、この機能を選択して使用する必要があります。

監査が有効な場合、Amazon DocumentDB はデータ定義言語 (DDL)、データ操作言語 (DML)、認証、認可、およびユーザー管理イベントを Amazon CloudWatch Logs に記録します。監査が有効な場合、Amazon DocumentDB はクラスターの監査レコード (JSON ドキュメント) を Amazon CloudWatch Logs にエクスポートします。Amazon CloudWatch Logs を使用すると、Amazon DocumentDB 監査イベントを分析、モニタリング、およびアーカイブすることができます。

Amazon DocumentDB では監査を有効にしても追加料金は発生しませんが、CloudWatch Logs の使用には標準料金がかかります。CloudWatch Logs の料金の詳細については、Amazon CloudWatch の料金 をご覧ください。

Amazon DocumentDB 監査機能は、モニタリング対象のサービスリソースの使用状況とは明確に異なります AWS CloudTrail。CloudTrail は、 AWS Command Line Interface （ AWS CLI) またはクラスター、インスタンス、パラメータグループ、スナップショットなどのリソース AWS Management Console で実行されるオペレーションを記録します。CloudTrailを使用した リソースの監査はデフォルトで有効化され、無効にすることはできません。Amazon DocumentDB 監査機能は、オプトイン機能です。これによって、オブジェクト (データベース、コレクション、インデックスやユーザーなど) のクラスター内で発生したオペレーションが記録されます。

トピック

- サポートされるイベント
- 監査の有効化
- 監査の無効化
- 監査イベントへのアクセス
- DML 監査イベントのフィルタリング

サポートされるイベント

Amazon DocumentDB 監査は、以下のカテゴリのイベントをサポートしています。

- データ定義言語 (DDL)- データベースの管理操作、接続、ユーザー管理、認可が含まれます。
- データ操作言語読み取りイベント (DML 読み取り) - `find()`、 およびさまざまな集計演算子、算術演算子、ブール演算子、およびその他の読み取りクエリ演算子が対象になります。
- データ操作言語書き込みイベント (DML 書き込み) - `insert()`, `update()`, `delete()`, および `bulkWrite()` 演算子が対象になります。

イベントタイプは次のとおりです。

イベントタイプ	カテゴリ	説明
`authCheck`	Authorization	結果コード 0: 成功
		結果コード 13: 不正な操作によるオペレーションの実行。
`authenticate`	接続	新規接続に対する認証 (成功または失敗)。
`auditConfigure`	DDL	フィルター設定の監査。
`createDatabase`	DDL	新しいデータベースの作成。
`createCollection`	DDL	データベース内での新しいコレクションの作成。
`createIndex`	DDL	コレクション内での新しいインデックスの作成。

イベントタイプ	カテゴリ	説明
dropCollection	DDL	データベース内のコレクションの削除。
dropDatabase	DDL	データベースの削除。
dropIndex	DDL	コレクション内のインデックスの削除。
modifyChangeStreams	DDL	変更ストリームが作成されました。
renameCollection	DDL	データベース内のコレクションの名前変更。
createRole	ロールの管理	ロールの作成。
dropAllRolesFromDatabase	ロールの管理	データベース内のすべてのロールの削除。
dropRole	ロールの管理	ロールの削除。
grantPrivilegesToRole	ロールの管理	ロールへの権限の付与。
grantRolesToRole	ロールの管理	ユーザー定義ロールへのロールの付与。
revokePrivilegesFromRole	ロールの管理	ロールからの権限の取り消し。
revokeRolesFromRole	ロールの管理	ユーザー定義ロールからロールの取り消し。
updateRole	ロールの管理	ロールの更新。

イベントタイプ	カテゴリ	説明
createUser	ユーザー管理	新しいユーザーの作成。
dropAllUsersFromDatabase	ユーザー管理	データベース内のすべてのユーザーの削除。
dropUser	ユーザー管理	既存のユーザーの削除。
grantRolesToUser	ユーザー管理	ユーザーにロールを付与する。
revokeRolesFromUser	ユーザー管理	ユーザーからロールを取り消します。
updateUser	ユーザー管理	既存のユーザーの更新。
insert	DML 書き込み	1 つまたは複数のドキュメントをコレクションに挿入します。
delete	DML 書き込み	コレクションから 1 つまたは複数のドキュメントを削除します。
update	DML 書き込み	既存の 1 つまたは複数のコレクション内のドキュメントを変更します。
bulkWrite	DML 書き込み	実行順序を制御を使って複数の書き込み操作を実行します。

イベントタイプ	カテゴリ	説明
setAuditConfig	DML 書き込み	DML 監査向け新しいフィルターを設定します。
count	DML 読み取り	コレクションまたはビューの find() クエリに一致するドキュメントの数を返します。
countDocuments	DML 読み取り	コレクションまたはビューのクエリに一致するドキュメントの数を返します。
find	DML 読み取り	コレクションまたはビュー内のドキュメントを選択し、選択したドキュメントにカーソルを返します。
getAuditConfig	DML 読み取り	DML 監査向けの現在のフィルターを取得します。
findAndModify	DML 読み取りと DML 書き込み	単一のドキュメントを変更して返します。
findOneAndDelete	DML 読み取りと DML 書き込み	フィルターと並べ替え条件に基づいて 1 つのドキュメントを削除し、削除したドキュメントを返します。

イベントタイプ	カテゴリ	説明
`findOneAndReplace`	DML 読み取りと DML 書き込み	指定したフィルターに基づいて 1 つのドキュメントを置き換えます。
`findOneAndUpdate`	DML 読み取りと DML 書き込み	フィルターと並べ替え条件に基づいて 1 つのドキュメントを更新します。
`aggregate`	DML 読み取りと DML 書き込み	集約パイプラインの API をサポートします。
`distinct`	DML 読み取り	1 つのコレクションまたはビューから指定されたフィールドの個別値を検索し、その結果を配列で返します。

> **Note**
>
> DML イベントドキュメントパラメータフィールドの値には 1 KB のサイズ制限があります。サイズが 1 KB を超えると、Amazon DocumentDB 値は切り捨てられます。

> **Note**
>
> 現時点では、TTL 削除イベントは監査されません。

監査の有効化

クラスターでの監査の有効化は、2 ステップのプロセスです。両方のステップが完了していることを確認してください。完了していない場合、監査ログは CloudWatch Logs に送信されません。

ステップ 1. audit_logs クラスターパラメータを有効にする

監査を有効にするには、パラメータグループの `audit_logs` パラメータを変更する必要があります。`audit_logs` は記録するイベントのカンマ区切りのリストです。イベントはすべて小文字で指定する必要があります。リスト要素間に空白があってはいけません。

パラメータには以下の値を指定できます。

値	説明	
`ddl`	設定すると、createDatabase、dropDatabase、createCollection、dropCollection、createIndex、dropIndex、authCheck、authenticate、createUser、dropUser、grantRolesToUser、revokeRolesFromUser、updateUser、dropAllUsersFromDatabase などの DDL イベントの監査が有効になります	
`dml_read`	これを設定すると、検索、ソート数、個別、グループ、射影、巻き戻し、geoNear、geoIntersects、geoWithin、その他の MongoDB 読み取	

値	説明	
	りクエリ演算子などの DML 読み取りイベントの監査が可能になります。	
dml_write	設定すると、insert()、update()、delete()、bulkWrite() などの DML 書き込みイベントの監査が有効になります。	
all	設定すると、読み取りクエリ、書き込みクエリ、データベースアクション、管理者アクションなどのデータベースイベントの監査が有効になります。	
none	設定すると、監査が無効になります	

値	説明	
`enabled` (レガシー)	これは「ddl」と同等のレガシーパラメータ設定です。設定すると、createDatabase、dropDatabase、createCollection、dropCollection、createIndex、dropIndex、authCheck、authenticate、createUser、dropUser、grantRolesToUser、revokeRolesFromUser、updateUser、dropAllUsersFromDatabase などの DDL イベントの監査が有効になります。レガシー設定であるため、この設定の使用はお勧めしません。	
`disabled` (レガシー)	これは「なし」と同等のレガシーパラメータ設定です。レガシー設定であるため、この設定の使用はお勧めしません。	

> **Note**
>
> audit_logs クラスターパラメーターのデフォルト値は none (レガシー「disabled」) です。

上記の値を組み合わせて使用することもできます。

値	説明
`ddl, dml_read`	設定すると、DDL イベントと DML 読み取りイベントの監査が有効になります。
`ddl, dml_write`	設定すると、DDL イベントと DML 書き込みイベントの監査が有効になります。
`dml_read, dml_write`	設定すると、すべての DML イベントの監査が有効になります。

> **Note**
>
> 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。

詳細については次を参照してください:

- Amazon DocumentDB クラスターパラメータグループを作成する

  カスタムパラメータグループを作成したら、`audit_logs` パラメータ値を `all` に変更します。
- Amazon DocumentDB クラスターパラメータグループを変更する

ステップ 2. Amazon CloudWatch Logs のエクスポートを有効にする

`audit_logs` クラスターパラメータの値が `enabled` か `ddl`、`dml_read`、`dml_write` のいずれかの場合、Amazon DocumentDB がログを Amazon CloudWatch にエクスポートできるようにする必要があります。上述の手順のいずれかを省略すると、監査ログは CloudWatch に送信されません。

クラスターを作成するときにポイントインタイムリストアまたはスナップショットの復元を実行する場合、以下のステップで CloudWatch Logs を有効にすることができます。

Using the AWS Management Console

コンソールを使用して Amazon DocumentDB へのログを CloudWatch にエクスポートすることを有効にするには、次のトピックを参照してください。

- クラスターを作成する場合 - を使用したクラスターとプライマリインスタンスの作成 AWS Management Console の「クラスターを作成する: 追加の設定 (ステップ 5、[ログのエクスポート])」を参照してください。
- 既存のクラスターを変更する場合 - Amazon DocumentDB クラスターの変更
- クラスタースナップショットリストアを実行する場合 - クラスタースナップショットからの復元
- ポイントインタイムリストアを実行する場合 - 特定時点への復元

Using the AWS CLI

新しいクラスターの作成時に監査ログを有効にするには

次のコードでは、クラスター `sample-cluster` を作成し、CloudWatch 監査ログを有効にします。

Example

Linux、macOS、Unix の場合:

```
aws docdb create-db-cluster \
    --db-cluster-identifier sample-cluster \
    --port 27017 \
    --engine docdb \
    --master-username master-username \
    --master-user-password password \
```

```
    --db-subnet-group-name default \
    --enable-cloudwatch-logs-exports audit
```

Windows の場合:

```
aws docdb create-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --port 27017 ^
    --engine docdb ^
    --master-username master-username ^
    --master-user-password password ^
    --db-subnet-group-name default ^
    --enable-cloudwatch-logs-exports audit
```

既存のクラスターを変更するときに監査ログを有効にするには

次のコードでは、クラスター sample-cluster を変更し、CloudWatch 監査ログを有効にします。

Example

Linux、macOS、Unix の場合:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Windows の場合:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

これらのオペレーションによる出力は、次のようになります (JSON 形式)。

```
{
    "DBCluster": {
        "HostedZoneId": "ZNKXH85TT8WVW",
        "StorageEncrypted": false,
        "DBClusterParameterGroup": "default.docdb4.0",
```

```
        "MasterUsername": "<user-name>",
        "BackupRetentionPeriod": 1,
        "Port": 27017,
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-77186e0d"
            }
        ],
        "DBClusterArn": "arn:aws:rds:us-east-1:900083794985:cluster:sample-cluster",
        "Status": "creating",
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
        "MultiAZ": false,
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1c",
            "us-east-1f"
        ],
        "DBSubnetGroup": "default",
        "DBClusterMembers": [],
        "ReaderEndpoint": "sample-cluster.cluster-ro-corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
        "EnabledCloudwatchLogsExports": [
            "audit"
        ],
        "PreferredMaintenanceWindow": "wed:03:08-wed:03:38",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2019-02-13T16:35:04.756Z",
        "DbClusterResourceId": "cluster-YOS52CUXGDTNKDQ7DH72I4LED4",
        "Endpoint": "sample-cluster.cluster-corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
        "PreferredBackupWindow": "07:16-07:46",
        "DBClusterIdentifier": "sample-cluster"
    }
}
```

監査の無効化

CloudWatch Logs のエクスポートを無効にし、audit_logs パラメータを無効にすることで、監査を無効にすることができます。

CloudWatch Logs のエクスポートの無効化

監査ログのエクスポートは、 AWS Management Console または を使用して無効にできます AWS CLI。

Using the AWS Management Console

次の手順では、 を使用してCloudWatch AWS Management Console へのログの Amazon DocumentDB エクスポートを無効にします。

監査ログを無効にするには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。ログのエクスポートを無効にするクラスターの名前の左側にあるボタンを選択します。
3. [アクション] を選択してから、[変更] を選択します。
4. [ログのエクスポート] セクションまで下にスクロールし、[無効] を選択します。
5. [続行] をクリックしてください。
6. 変更を確認してから、この変更をクラスターにいつ適用するかを選択します。
   - Apply during the next scheduled maintenance window (次に予定されているメンテナンス期間中に適用)
   - すぐに適用
7. [クラスタークラスターの変更] を選択します。

Using the AWS CLI

次のコードでは、クラスター `sample-cluster` を変更し、CloudWatch 監査ログを無効にします。

Example

Linux、macOS、Unix の場合:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
```

```
    --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

Windows の場合:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "DBCluster": {
        "DBClusterParameterGroup": "default.docdb4.0",
        "HostedZoneId": "ZNKXH85TT8WVW",
        "MasterUsername": "<user-name>",
        "Status": "available",
        "Engine": "docdb",
        "Port": 27017,
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1c",
            "us-east-1f"
        ],
        "EarliestRestorableTime": "2019-02-13T16:35:50.387Z",
        "DBSubnetGroup": "default",
        "LatestRestorableTime": "2019-02-13T16:35:50.387Z",
        "DBClusterArn": "arn:aws:rds:us-east-1:900083794985:cluster:sample-
cluster2",
        "Endpoint": "sample-cluster2.cluster-corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster2.cluster-ro-corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
        "BackupRetentionPeriod": 1,
        "EngineVersion": "4.0.0",
        "MultiAZ": false,
        "ClusterCreateTime": "2019-02-13T16:35:04.756Z",
        "DBClusterIdentifier": "sample-cluster2",
        "AssociatedRoles": [],
        "PreferredBackupWindow": "07:16-07:46",
        "DbClusterResourceId": "cluster-YOS52CUXGDTNKDQ7DH72I4LED4",
        "StorageEncrypted": false,
        "PreferredMaintenanceWindow": "wed:03:08-wed:03:38",
        "DBClusterMembers": [],
```

```
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-77186e0d"
            }
        ]
    }
}
```

audit_logs パラメータを無効にする

クラスターの `audit_logs` パラメータを無効にするには、`audit_logs` パラメータ値が `disabled` であるパラメータグループを使用するようにクラスターを変更します。または、クラスターの `audit_logs` パラメータグループのパラメータ値を `disabled` に変更することもできます。

詳細については、以下の各トピックを参照してください。

- Amazon DocumentDB クラスターの変更
- Amazon DocumentDB クラスターパラメータグループを変更する

監査イベントへのアクセス

Amazon CloudWatch の監査イベントにアクセスするには、以下のステップを実行してください。

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. Amazon DocumentDB クラスターと同じリージョンであることを確認します。
3. ナビゲーションペインで [ログ] を選択します。
4. クラスターの監査ログを確認するため、リストから **/aws/docdb/*yourClusterName*/audit** を見つけ出して選択します。

   各インスタンスの監査イベントは、各インスタンス名の下に表示されます。

DML 監査イベントのフィルタリング

DML 監査フィルタリングの開始方法

DML 監査イベントは、Amazon CloudWatch への書き込み前にフィルタリングできます。この機能を使用するには、監査ログと DML ログ記録を有効にする必要があります。Amazon DocumentDB

は、atype、command、user、namespace、auditAuthorizationSuccess でのフィルタリングをサポートしています。

> **Note**
>
> DDL イベントはフィルタリングされません。

db.adminCommand({ command }) オペレーションの setAuditConfig、filter、auditAuthorizationSuccess パラメータを使用して監査フィルターを指定することで、いつでも監査フィルタリングを有効にできます。

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter:
         {
            //filter conditions
         },
      auditAuthorizationSuccess: true | false
   }
)
```

次のコマンドを実行して、監査フィルター設定を取得することもできます。

```
db.admin.runCommand( { getAuditConfig: 1})
```

セキュリティ要件

DML 監査フィルターを設定または一覧表示する際に admindb に対して上記のコマンドを実行できるのは、特権アクション auditConfigure を持つデータベースユーザー/ロールのみです。[clusterAdmin、hostManager、root] の組み込みロールのいずれかを使用するか、あるいは auditConfigure 権限を持つカスタムロールを作成できます。以下は、auditConfigure 権限で既存のロールを使用する例と、カスタムロールを使用する例です。

ロールが組み込まれているユーザー:

```
use admin
db.createUser(
```

```
  {
    user: "myClusterAdmin",
    pwd: "password123",
    roles: [ { role: "clusterAdmin", db: "admin" } ]
  }
)
```

カスタムロールを持つユーザー:

```
use admin
db.createRole(
   {
     role: "myRole",
     privileges: [
       { resource: { cluster: true }, actions: [ "auditConfigure" ] }
     ],
     roles: []
   }
)
db.createUser(
  {
    user: "myUser",
    pwd: "myPassword",
    roles: [ { role: "myRole", db: "admin" } ]
  }
)
```

ユースケースのフィルタリング

例: コマンドによるイベントのフィルタリング

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
        "$and": [
         {
            "param.command":
               {
                  $in: [ "find","count", "insert", "delete", "update",
 "findandmodify" ]
               }
         }
```

```
         ]
      },
      auditAuthorizationSuccess: true
   }
)
```

例: ユーザー名によるイベントのフィルタリング

この例では、ユーザー「myUser」のみがログに記録されます。

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.user":
               {
                  $in: [ "myUser" ]
               }
         }
         ]},
      auditAuthorizationSuccess: true})
```

例: **atype** によるフィルタリング

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {atype: "authCheck"},
      auditAuthorizationSuccess: true
   })
```

> **Note**
>
> すべての DML ログは、atype として authCheck を持っています。DDL のみが別の atype を持っています。filter に authCheck 以外の値を指定した場合、CloudWatch での DML ログの生成は行われません。

例: 演算子によって結合された複数のフィルターを使用したフィルタリング

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.command":
               {
                  $in: [ "find","count", "insert", "delete", "update",
 "findandmodify" ]
               }
         }
         ],
       "$nor": [
         {
            "param.command":
               {
                  $in: ["count", "insert", "delete", "update", "findandmodify" ]
               }
         }]
       },
      auditAuthorizationSuccess: true})
```

> **Note**
>
> 最上位レベルでは、$and、$or、$nor のみがサポートされています。他の演算子はサポートされていないため、エラーが発生します。

例: **auditAuthorizationSuccess** によるイベント別のフィルタリング

このフィルターでは、認証に合格したすべてのコマンドはログに記録されません。

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: false
   }
)
```

例: **$in** および **$nin** 条件によるフィルタリング

$in と $nin の両方で使用する場合、条件間に暗黙的な「and」があるため、コマンドはログに記録されません。この例では、正規表現で find コマンドがブロックされるため、ログには何も記録されません。

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            atype: "authCheck",
            "param.command":
               {
                  $in: [ "find", "insert", "delete", "update", "findandmodify" ],
                  $nin: ["count", "insert", "delete", "update", "findandmodify" ],
                  $not: /^^find.*/
               }
         },
         ],
       "$or": [
         {
            "param.command":
               {
                  $nin: ["count", "insert", "delete", "update", "findandmodify" ]
               }
         }]
       },
      auditAuthorizationSuccess: true})
```

例: **namespace** によるフィルタリング

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.ns":
               {
                  $in: [ "test.foo" ]
               }
         }
         ]},
```

```
      auditAuthorizationSuccess: true})
```

例: デフォルトフィルターへのリセット

デフォルト値にリセットすると、すべての DML 監査イベントがログに記録されます。フィルターをデフォルト値にリセットするには、次のコマンドを実行します。

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: true
   }
)
```

Amazon DocumentDB でのバックアップと復元

Amazon DocumentDB (MongoDB 互換) は、データを Amazon Simple Storage Service (Amazon S3) に 1 ~ 35 日間、継続的にバックアップするため、バックアップ保持期間の任意の時点にすばやく復元できます。Amazon DocumentDB は、この継続的なバックアッププロセスの一環として、データの自動スナップショットも取得します。

> **Note**
>
> これらはサービス管理された Amazon S3 バケットであり、バックアップファイルにアクセスすることはできません。自分でバックアップを行いたい場合は、[データのダンプ、復元、インポート、およびエクスポート] の手順に従ってください。

クラスターのデータの手動スナップショットを作成して、バックアップ保持期間を超えてバックアップデータを保持することもできます。バックアッププロセスは、クラスターのパフォーマンスに影響を与えません。

このセクションでは、Amazon DocumentDB でのバックアップ機能のユースケースについて説明し、Amazon DocumentDB クラスターのためのバックアップ管理法を示します。

トピック

- バックアップと復元: 概念
- バックアップストレージの使用状況を確認する
- データのダンプ、復元、インポート、およびエクスポート
- クラスタースナップショットに関する考慮事項
- 自動スナップショットと手動スナップショットの比較
- 手動クラスタースナップショットの作成
- Amazon DocumentDB クラスタースナップショットのコピー
- Amazon DocumentDB クラスタースナップショットの共有
- クラスタースナップショットからの復元
- 特定時点への復元
- クラスタースナップショットの削除

バックアップと復元: 概念

名詞	説明	API (動詞)

バックアップの保存期間	特定時点への復元を実行できる、1～35 日間の期間。	`create-db-cluster`
		`modify-db-cluster`
		`restore-db-cluster-to-point-in-time`

Amazon DocumentDB のストレージボリューム	3 つのアベイラビリティーゾーンに渡って 6 つの方法でデータをレプリケートする、可用性と耐久性に優れたストレージボリューム。Amazon DocumentDB クラスターは、クラスター内のインスタンス数に関係なく、高い耐久性があります。	`create-db-cluster` `delete-db-cluster`
バックアップウィンドウ	自動スナップショット	`create-db-cluster`

名詞	説明	API (動詞)
	が実行される 1 日の時間帯。	describe-db-cluster modify-db-cluster
自動スナップショット	クラスターの完全なバックアップである毎日のスナップショット。これは Amazon DocumentDB の継続的なバックアッププロセスによって自動的に作成されます。	restore-db-cluster-from-snapshot describe-db-cluster-snapshot-attributes describe-db-cluster-snapshots
手動スナップショット	バックアップ期間を超えてクラスターの完全バックアップを保持するために、手動で作成するスナップショット。	create-db-cluster-snapshot copy-db-cluster-snapshot delete-db-cluster-snapshot describe-db-cluster-snapshot-attributes describe-db-cluster-snapshots modify-db-cluster-snapshot-attribute

バックアップストレージの使用状況を確認する

Amazon DocumentDB バックアップストレージは、バックアップ保持期間内の継続的なバックアップと、保持期間外の手動スナップショットで構成されます。バックアップストレージの使用状況を管理するには、バックアップの保存間隔を短縮するか、不要になった古い手動スナップショットを削除するか、またはその両方を行います。Amazon DocumentDB バックアップの一般情報については、Amazon DocumentDB でのバックアップと復元 を参照してください。Amazon DocumentDB バックアップストレージの料金情報については、Amazon DocumentDB の料金 を参照してください。

コストを管理するには、保存期間を超えて保持されている継続的バックアップと手動スナップショットに使用されているストレージの量をモニタリングできます。次に、バックアップの保存間隔を短縮し、不要になった手動スナップショットを削除できます。

次に示すように、 Amazon DocumentDB バックアップに使用されているストレージの量を確認およびモニタリングするために、Amazon CloudWatch メトリクスの `TotalBackupStorageBilled` 、`SnapshotStorageUsed` 、および `BackupRetentionPeriodStorageUsed` を使用できます。

- `BackupRetentionPeriodStorageUsed` は現時点で継続的なバックアップの保存に使用されているバックアップストレージの量を表します。このメトリクス値は、クラスターボリュームのサイズと、保持期間中に行った変更数によって変わります。ただし、請求のため、このメトリクスが保持期間中に累積的なクラスターボリュームのサイズを超えることはありません。たとえば、クラスターサイズが 100 GiB で、保持期間が 2 日間である場合、`BackRetentionPeriodStorageUsed` の最大値は 200 GiB (100 GiB + 100 GiB) となります。

- `SnapshotStorageUsed` は、バックアップ保持期間を超えて手動スナップショットの保存に使用されるバックアップストレージの量を表します。保持期間内に作成された手動スナップショットは、バックアップストレージに対してカウントされません。同様に、自動スナップショットはバックアップストレージに対して計算されません。各スナップショットのサイズは、そのスナップショットを作成した時点のクラスターボリュームのサイズです。`SnapshotStorageUsed` 値は、保持するスナップショットの数と、各スナップショットのサイズによって異なります。たとえば、保存期間外のスナップショットが 1 つあり、このスナップショットを作成した時点のクラスターのボリュームサイズが 100 GiB であったとします。この場合の `SnapshotStorageUsed` の量は 100 GiB です。

- TotalBackupStorageBilled は BackupRetentionPeriodStorageUsed と SnapshotStorageUsed の合計から、1 日のクラスターボリュームのサイズと等しい無料バックアップストレージの量を引いた値になります。例えば、クラスターサイズが 100 GiB、保持期間が 1 日であり、保持期間外に 1 つのスナップショットがある場合、TotalBackupStorageBilled は 100 GiB (100 GiB + 100 GiB - 100 GiB) となります。

- これらのメトリクスは、Amazon DocumentDB クラスターごとに個別に計算されます。

CloudWatch コンソールを通して CloudWatch メトリクスを使用することで、Amazon DocumentDB クラスターをモニタリングし、レポートを作成することができます。CloudWatch のメトリクスを使用する方法については、Amazon DocumentDB のモニタリング を参照してください。

データのダンプ、復元、インポート、およびエクスポート

mongodump、mongorestore、mongoexport、および mongoimport の各ユーティリティを使用して、Amazon DocumentDB クラスターの内外にデータを移動できます。このセクションでは、これらの各ツールの目的と、パフォーマンスの向上に役立つ設定について説明します。

トピック

- mongodump
- mongorestore
- mongoexport
- mongoimport
- チュートリアル

mongodump

mongodump ユーティリティは、MongoDB データベースのバイナリ (BSON) バックアップを作成します。mongodump のツールは、バイナリ形式でデータを保存することでサイズを縮小できるため、Amazon DocumentDB クラスターへの復元時にソース MongoDB デプロイからデータをダンプする方法として最適です。

コマンドの実行に使用しているインスタンスまたはマシンで使用可能なリソースに応じて、 --numParallelCollectionsオプションを使用してデフォルト 1 からダンプされた並列コレクショ

ンの数を増やすmongodumpことで、 を高速化できます。目安として、Amazon DocumentDB クラスターのプライマリインスタンスで [vCPU] ごとに 1 つのワーカーから開始します。

> **Note**
>
> Amazon DocumentDB には、バージョン 100.6.1 までの MongoDB データベースツールをお勧めします。MongoDB データベースツールのダウンロードには、こちらからアクセスできます。

使用例

Amazon DocumentDB クラスター (`sample-cluster`) での mongodump のユーティリティの使用例を次に示します。

```
mongodump --ssl \
    --host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --collection=sample-collection \
    --db=sample-database \
    --out=sample-output-file \
    --numParallelCollections 4  \
    --username=sample-user \
    --password=abc0123 \
    --sslCAFile global-bundle.pem
```

mongorestore

mongorestore ユーティリティを使用すると、mongodump ユーティリティで作成されたデータベースのバイナリ (BSON) バックアップを復元できます。復元のパフォーマンスを向上させるには、`--numInsertionWorkersPerCollection` オプションを使用して復元時に各コレクションのワーカー数 (デフォルトは 1) を増やします。目安として、Amazon DocumentDB クラスターのプライマリインスタンスで [vCPU] ごとに 1 つのワーカーから開始します。

使用例

Amazon DocumentDB クラスター (`sample-cluster`) での mongorestore のユーティリティの使用例を次に示します。

```
mongorestore --ssl \
```

```
    --host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --username=sample-user \
    --password=abc0123 \
    --sslCAFile global-bundle.pem <fileToBeRestored>
```

mongoexport

mongoexport のツールは、Amazon DocumentDB 中のデータを JSON、CSV、または TSV ファイル形式にエクスポートします。mongoexport ツールは、人間またはマシンが読み取り可能なデータをエクスポートする方法として最適です。

> **Note**
>
> mongoexport は並列エクスポートを直接サポートしていません。ただし、コレクションごとに複数の mongoexport ジョブを同時に実行することで、パフォーマンスを向上させることができます。

使用例

Amazon DocumentDB クラスター (sample-cluster) での mongoexport ツールの使用例を次に示します。

```
mongoexport --ssl \
    --host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --collection=sample-collection \
    --db=sample-database \
    --out=sample-output-file \
    --username=sample-user \
    --password=abc0123 \
    --sslCAFile global-bundle.pem
```

mongoimport

mongoimport のツールは、JSON、CSV、または TSV ファイルの内容を Amazon DocumentDB クラスターにインポートします。--numInsertionWorkers パラメータを使用すると、インポート (デフォルトは 1) を並列化および高速化できます。

使用例

Amazon DocumentDB クラスター (sample-cluster) でのmongoimport ツールの使用例を次に示します。

```
mongoimport --ssl \
    --host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --collection=sample-collection \
    --db=sample-database \
    --file=<yourFile> \
    --numInsertionWorkers 4 \
    --username=sample-user \
    --password=abc0123 \
    --sslCAFile global-bundle.pem
```

チュートリアル

次のチュートリアルでは mongodump、mongorestore、mongoexport、および mongoimport の各ユーティリティを使用して Amazon DocumentDB クラスターの内外にデータを移動する方法について説明します。

1. 前提条件 - 開始する前に、Amazon DocumentDB クラスターがプロビジョニングされていること、およびクラスターと同じ VPC 内の Amazon EC2 インスタンスにアクセスできることを確認してください。詳細については、「Amazon EC2 を使用した接続」を参照してください。

   mongo ユーティリティツールを使用するには、次のように、EC2 インスタンスに mongodb-org-tools パッケージがインストールされている必要があります。

   ```
   sudo yum install mongodb-org-tools-4.0.18
   ```

   Amazon DocumentDB はデフォルトで TLS (Transport Layer Security) 暗号化を使用するため、次のように mongo シェルを使用するために Amazon RDS 認証機関 (CA) ファイルもダウンロードする必要があります。

   ```
   wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
   ```

2. サンプルデータをダウンロードする - このチュートリアルでは、レストランに関する情報が含まれているサンプルデータをダウンロードします。

```
wget https://raw.githubusercontent.com/ozlerhakan/mongodb-json-files/master/
datasets/restaurant.json
```

3. Amazon DocumentDB にサンプルデータをインポートする - データは論理 JSON 形式であるため、mongoimport のユーティリティを使用してデータを Amazon DocumentDB クラスターにインポートします。

```
mongoimport --ssl \
    --host="tutorialCluster.amazonaws.com:27017" \
    --collection=restaurants \
    --db=business \
    --file=restaurant.json \
    --numInsertionWorkers 4 \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile global-bundle.pem
```

4. **mongodump** でデータをダンプする - Amazon DocumentDB クラスターにデータをインポートしたので、mongodump ユーティリティを使用してこのデータのバイナリダンプを取得できます。

```
mongodump --ssl \
    --host="tutorialCluster.us-east-1.docdb.amazonaws.com:27017"\
    --collection=restaurants \
    --db=business \
    --out=restaurantDump.bson \
    --numParallelCollections 4 \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile global-bundle.pem
```

5. **restaurants** コレクションを削除する - restaurants のコレクションを business のデータベースに復元する前に、次のように、そのデータベース内にある既存のコレクションを最初に削除する必要があります。

```
use business
```

```
db.restaurants.drop()
```

6. **mongorestore** でデータを復元する - ステップ 3 からのデータのバイナリダンプを使用して、mongorestore のユーティリティを使用してデータを Amazon DocumentDB クラスターに復元できます。

```
mongorestore --ssl \
    --host="tutorialCluster.us-east-1.docdb.amazonaws.com:27017" \
    --numParallelCollections 4 \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile global-bundle.pem restaurantDump.bson
```

7. **mongoexport** を使用してデータをエクスポートする - チュートリアルを完了するには、クラスターからデータを JSON ファイルの形式でエクスポートします。ステップ 1 でインポートしたファイルと同じ結果になります。

```
mongoexport --ssl \
    --host="tutorialCluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --collection=restaurants \
    --db=business \
    --out=restaurant2.json \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile global-bundle.pem
```

8. 検証 - 次のコマンドを使用して、ステップ 5 の出力がステップ 1 と同じ結果をもたらすことを検証できます。

```
wc -l restaurant.json
```

このコマンドからの出力:

```
2548 restaurant.json
```

```
wc -l restaurant2.json
```

このコマンドからの出力:

```
2548 restaurant2.json
```

クラススナップショットに関する考慮事項

Amazon DocumentDB は、クラスターのバックアップウィンドウ中にクラスターの自動スナップショットを毎日作成します。Amazon DocumentDB は、指定したバックアップ保持期間に従って、クラスターの自動スナップショットを保存します。必要に応じて、バックアップ保持期間内の任意の時点でクラスターを復旧できます。同じクラスターに対して同じリージョンでコピー操作が実行されている間は、自動スナップショットは発生しません。

トピック

- バックアップストレージ
- バックアップウィンドウ
- バックアップの保存期間
- クラススナップショットの暗号化をコピーする

自動のクラススナップショットに加えて、手動でクラスターのスナップショットを作成できます。自動スナップショットと手動スナップショットの両方をコピーできます。詳細については、「手動クラススナップショットの作成」および「Amazon DocumentDB クラススナップショットのコピー」を参照してください。

> **Note**
>
> 自動スナップショットを作成するには、クラスターのスタータスが 使用可能 である必要があります。
> Amazon DocumentDB 自動クラススナップショットを共有することはできません。回避策として、自動スナップショットをコピーして手動スナップショットを作成し、そのコピーを共有できます。スナップショットのコピーの詳細については、「Amazon DocumentDB クラススナップショットのコピー」を参照してください。スナップショットからクラスターを復元する方法の詳細については、「クラススナップショットからの復元」を参照してください。

バックアップストレージ

それぞれの Amazon DocumentDB バックアップストレージ AWS リージョン は、バックアップ保持期間に必要なバックアップストレージで構成されます。これには、そのリージョンの自動および手動のクラススナップショットが含まれます。デフォルトのバックアップ保持期間は 1 日です。

バックアップストレージの料金に関するさらなる情報については、Amazon DocumentDB の料金 を参照してください。

クラスターを削除すると、自動スナップショットはすべて削除され、復旧することはできません。ただし、クラスターを削除しても、手動スナップショットは削除されません。クラスターを削除する前に Amazon DocumentDB が最終スナップショット (手動スナップショット) を作成するように選択した場合は、最終スナップショットを使用してクラスターを復元できます。

スナップショットとストレージの詳細については、「バックアップストレージの使用状況を確認する」を参照してください。

バックアップウィンドウ

自動スナップショットは、任意のバックアップウィンドウ中に毎日行われます。バックアップウィンドウに割り当てられた時間より長い時間がスナップショットに必要な場合、バックアップウィンドウが終了した後もバックアップが完了するまでバックアッププロセスが継続します。クラスターの週 1 回のメンテナンスウィンドウとバックアップウィンドウは重複できません。

クラスターの作成時に任意のバックアップウィンドウを指定しない場合、Amazon DocumentDB がデフォルトの 30 分のバックアップウィンドウを割り当てます。このウィンドウは、クラスターのリージョンに関連付けられた 8 時間の時間帯からランダムに選択されます。クラスターを変更すると、バックアップウィンドウを変更できます。詳細については、「Amazon DocumentDB クラスターの変更」を参照してください。

リージョン名	リージョン	UTC 時間帯
米国東部 (オハイオ)	us-east-2	03:00 ~ 11:00
米国東部 (バージニア北部)	us-east-1	03:00 ~ 11:00
米国西部 (オレゴン)	us-west-2	06:00 ~ 14:00
アフリカ (ケープタウン)	af-south-1	03:00 ~ 11:00
アジアパシフィック (香港)	ap-east-1	06:00 ~ 14:00
アジアパシフィック (ハイデラバード)	ap-south-2	06:30 ~ 14:30

リージョン名	リージョン	UTC 時間帯
アジアパシフィック (ムンバイ)	ap-south-1	06:00 ~ 14:00
アジアパシフィック (ソウル)	ap-northeast-2	13:00 ~ 21:00
アジアパシフィック (シンガポール)	ap-southeast-1	14:00 ~ 22:00
アジアパシフィック (シドニー)	ap-southeast-2	12:00 ~ 20:00
アジアパシフィック (東京)	ap-northeast-1	13:00 ~ 21:00
カナダ (中部)	ca-central-1	03:00 ~ 11:00
中国 (北京)	cn-north-1	06:00 ~ 14:00
中国 (寧夏)	cn-northwest-1	06:00 ~ 14:00
欧州 (フランクフルト)	eu-central-1	21:00 ~ 05:00
欧州 (アイルランド)	eu-west-1	22:00 ~ 06:00
欧州 (ロンドン)	eu-west-2	22:00 ~ 06:00
欧州 (ミラノ)	eu-south-1	02:00 ~ 10:00
欧州 (パリ)	eu-west-3	23:59 ~ 07:29
欧州 (スペイン)	eu-south-2	02:00 ~ 10:00
欧州 (ストックホルム)	eu-north-1	4:00 ~ 12:00
中東 (アラブ首長国連邦)	me-central-1	05:00 ~ 13:00
南米 (サンパウロ)	sa-east-1	00:00 ~ 08:00
イスラエル (テルアビブ)	il-central-1	04:00 ~ 12:00
AWS GovCloud (米国東部)	us-gov-east-1	17:00 ~ 01:00

リージョン名	リージョン	UTC 時間帯
AWS GovCloud (米国西部)	us-gov-west-1	06:00 ~ 14:00

バックアップの保存期間

バックアップ保持期間は、自動的に削除される前に自動バックアップが保持される日数です。Amazon DocumentDB では、1 ~ 35 日間のバックアップ保持期間をサポートしています。

クラスターを作成するとき、バックアップ保持期間を設定できます。バックアップ保持期間を明示的に設定しない場合、デフォルトのバックアップ保持期間である 1 日がクラスターに割り当てられます。クラスターを作成したら、 AWS Management Console または を使用してクラスターを変更することで、バックアップ保持期間を変更できます AWS CLI。詳細については、「Amazon DocumentDB クラスターの変更」を参照してください。

クラスタースナップショットの暗号化をコピーする

クラスターとスナップショットの暗号化は、 KMS 暗号化キーに基づいています。 KMS キー ID は、その KMS 暗号化キーの Amazon リソースネーム (ARN)、 KMS キー識別子、または KMS キーエイリアスのいずれかとなります。

以下のガイドラインと制限が適用されます。

- 暗号化は、スナップショットの作成時にクラスターから推測されます。クラスターが暗号化されている場合、そのクラスターのスナップショットは同じ KMS キーで暗号化されます。クラスターが暗号化されていない場合、スナップショットは暗号化されません。
- 暗号化されたクラスタースナップショットを Amazon Web Services アカウントからコピーする場合、`KmsKeyId` の値を指定して新しい KMS 暗号化キーでコピーを暗号化できます。`KmsKeyId` の値を指定しないと、クラスタースナップショットのコピーはコピー元のクラスタースナップショットと同じ KMS キーで暗号化されます。
- 別の Amazon Web Services アカウントと共有されている暗号化されたクラスタースナップショットをコピーする場合は、`KmsKeyId` の値を指定する必要があります。
- 暗号化されたクラスタースナップショットを別の Amazon Web Services リージョンにコピーするには、`KmsKeyId` を、コピー先のリージョンでクラスタースナップショットのコピーの暗号化に使用する KMS キー ID に設定します。KMS 暗号化キーは、それが作成された Amazon Web Services リージョンに固有のものであるため、ある Amazon Web Services リージョンの暗号化キーを別の Amazon Web Services リージョンで使用することはできません。

- 暗号化されていないクラスタースナップショットをコピーして KmsKeyId パラメータに値を指定すると、エラーが返されます。

自動スナップショットと手動スナップショットの比較

Amazon DocumentDB (MongoDB 互換) の自動スナップショットおよび手動スナップショットの主な機能を以下に示します。

Amazon DocumentDB 自動スナップショットには以下の主な機能があります。

- 自動スナップショットの名前付け - 自動スナップショットの名前はパターン rds:<cluster-name>-yyyy-mm-dd-hh-mm に従い、yyyy-mm-dd-hh-mm はスナップショットが作成された日時を表します。
- スケジュールに基づいて自動的に作成 - クラスターを作成または変更するときに、バックアップ保持期間 を 1 ~ 35 日の整数値に設定できます。デフォルトでは、新しいクラスターのバックアップ保持期間は 1 日です。バックアップ保持期間により、自動的に削除される前に自動スナップショットが保持される日数が定義されます。Amazon DocumentDB クラスター上で自動バックアップを無効にすることはできません。

  バックアップ保持期間を設定することに加えて、バックアップウィンドウも設定します。これは、自動スナップショットが作成される 1 日の時間帯です。
- 自動スナップショットの削除 - 自動スナップショットは、自動スナップショットのクラスターを削除すると削除されます。自動スナップショットを手動で削除することはできません。
- 増分 - バックアップ保持期間中にデータベースの更新が記録されるため、変更の増分記録が作成されます。
- 自動スナップショットから復元 - AWS Management Console または AWS CLIを使用して自動スナップショットから復元することができます。を使用してスナップショットから復元する場合は AWS CLI、クラスターが利用可能になった後にインスタンスを個別に追加する必要があります。
- 共有 - Amazon DocumentDB自動クラスタースナップショットを共有することはできません。回避策として、自動スナップショットをコピーして手動スナップショットを作成し、そのコピーを共有できます。スナップショットのコピーの詳細については、「Amazon DocumentDB クラスタースナップショットのコピー」を参照してください。スナップショットからクラスターを復元する方法の詳細については、「クラスタースナップショットからの復元」を参照してください。
- バックアップ保持期間内の任意の時点から復元可能 - データベースの更新は増分で記録されるため、バックアップ保持期間内の任意の時点にクラスターを復元できます。

を使用して自動スナップショットまたはpoint-in-time復元から復元する場合は AWS CLI、クラスターが利用可能になった後にインスタンスを個別に追加する必要があります。

Amazon DocumentDB 手動スナップショットには以下の主な機能があります。

- オンデマンドで作成 — Amazon DocumentDB 手動スナップショットは、Amazon DocumentDB マネジメントコンソールまたは を使用してオンデマンドで作成されます AWS CLI。
- 手動スナップショットの削除 - Amazon DocumentDB コンソールまたは AWS CLIを使用して明示的に削除するときのみ、手動スナップショットは削除されます。クラスターを削除しても、手動スナップショットは削除されません。
- 完全バックアップ - 手動スナップショットが作成されると、クラスターのデータの完全バックアップが作成および保存されます。
- 手動スナップショットの名前付け - 手動スナップショット名を指定します。Amazon DocumentDB では、`datetime` のスタンプを名前に追加しないため、名前に含めたい場合は、その情報を追加する必要があります。
- 手動スナップショットから復元 - コンソールまたは AWS CLIを使用して、手動スナップショットから復元することができます。を使用してスナップショットから復元する場合は AWS CLI、クラスターが使用可能になった後にインスタンスを個別に追加する必要があります。
- Service Quotasタ — あたり最大 100 個の手動スナップショットに制限されています AWS リージョン。
- 共有 - 手動クラスタースナップショットは共有できます。これは、許可された AWS アカウントによってコピーできます。暗号化されている手動スナップショットまたは暗号化されていない手動スナップショットを共有できます。スナップショットのコピーの詳細については、「Amazon DocumentDB クラスタースナップショットのコピー」を参照してください。
- 手動スナップショットが作成された時点への復元 - 手動スナップショットから復元するときは、手動スナップショットが作成された時点に復元します。

  を使用してスナップショットから復元する場合は AWS CLI、クラスターが使用可能になった後にインスタンスを個別に追加する必要があります。

手動クラスタースナップショットの作成

手動スナップショットは、 AWS Management Console または を使用して作成できます AWS CLI。スナップショットの作成にかかる時間は、データベースのサイズによって異なります。スナップショットを作成する場合、以下を実行する必要があります。

1. バックアップするクラスターを特定します。
2. スナップショットに名前を付けます。これにより、後でスナップショットから復元できます。

Using the AWS Management Console

を使用して手動スナップショットを作成するには AWS Management Console、次のいずれかの方法に従います。

1. 方法 1:

   1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
   2. ナビゲーションペインで、[Snapshots] を選択してください。

      > **Tip**
      > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

   3. [スナップショット] ページで、[作成] を選択します。
   4. [クラスタースナップショットの作成] ページで、次の操作を行います。
      a. クラスター識別子 - クラスターのドロップダウンリストから、スナップショットを作成するクラスターを選択します。
      b. スナップショット識別子 - スナップショットの名前を入力します。

         スナップショットの名前付けの制約:

         - 長さは、[1 ~ 255] 個の英字、数字、またはハイフンです。
         - 1 字目は英字である必要があります。

- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
- AWS のアカウントごと、リージョンごとに、すべてのクラスター (Amazon RDS、Amazon Neptune、および　Amazon DocumentDB に渡って) で一意にする必要があります。

c. [作成] を選択します。

2. 方法 2:

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. [クラスター] ページで、スナップショットを作成するクラスターの左側にあるボタンを選択します。
4. [アクション] メニューで、[スナップショットの取得] を選択します。
5. [クラスタースナップショットの作成] ページで、次の操作を行います。

a. スナップショット識別子 - スナップショットの名前を入力します。

スナップショットの名前付けの制約:

- 長さは、[1 ~ 63] 個の英字、数字、またはハイフンです。
- 1 字目は英字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
- AWS のアカウントごと、リージョンごとに、すべてのクラスター (Amazon RDS、Amazon Neptune、および　Amazon DocumentDB に渡って) で一意にする必要があります。

b. [作成] を選択します。

Using the AWS CLI

を使用してクラスタースナップショットを作成するには AWS CLI、次のパラメータを指定して `create-db-cluster-snapshot`オペレーションを使用します。

パラメータ

- **`--db-cluster-identifier`** — 必須。スナップショットを作成するクラスターの名前。このクラスターは存在していて、使用可能である必要があります。
- **`--db-cluster-snapshot-identifier`** — 必須。作成する手動スナップショットの名前。

次の例では、`sample-cluster` という名前のクラスターの、`sample-cluster-snapshot` という名前のスナップショットを作成します。

Linux、macOS、Unix の場合:

```
aws docdb create-db-cluster-snapshot \
    --db-cluster-identifier sample-cluster \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Windows の場合:

```
aws docdb create-db-cluster-snapshot ^
    --db-cluster-identifier sample-cluster ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBClusterSnapshot": {
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1b",
            "us-east-1c"
        ],
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterIdentifier": "sample-cluster",
        "SnapshotCreateTime": "2020-04-24T04:59:08.475Z",
        "Engine": "docdb",
        "Status": "creating",
        "Port": 0,
```

```
        "VpcId": "vpc-abc0123",
        "ClusterCreateTime": "2020-01-10T22:13:38.261Z",
        "MasterUsername": "master-user",
        "EngineVersion": "4.0.0",
        "SnapshotType": "manual",
        "PercentProgress": 0,
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:<accountID>:cluster-
snapshot:sample-cluster-snapshot"
    }
}
```

Amazon DocumentDB クラスタースナップショットのコピー

Amazon DocumentDB では、同じ 内の手動スナップショットと自動スナップショット、 AWS リージョン または同じアカウント AWS リージョン 内の別のスナップショットをコピーできます。同じ AWS アカウント で他の が所有するスナップショットを共有することもできます AWS リージョン。ただし、1 つのステップ AWS アカウント でクラスタースナップショットを AWS リージョン と 間でコピーすることはできません。これらのアクションは個別に実行される必要があります。

コピーの代わりに、手動スナップショットを他の と共有することもできます AWS アカウント。詳細については、「Amazon DocumentDB クラスタースナップショットの共有」を参照してください。

> **Note**
>
> Amazon DocumentDB は、保持するバックアップデータおよびスナップショットデータの量と保持期間に基づいて請求を行います。Amazon DocumentDB バックアップおよびスナップショットに伴うストレージのさらなる詳細については、バックアップストレージの使用状況を確認する を参照してください。Amazon DocumentDB ストレージの料金情報については、Amazon DocumentDB の料金 を参照してください。

トピック

- 共有スナップショットのコピー
- 間でのスナップショットのコピー AWS リージョン
- 制限

- 暗号化の処理
- パラメータグループに関する考慮事項
- クラススナップショットのコピー

共有スナップショットのコピー

他のユーザーから共有されたスナップショットをコピーできます AWS アカウント。別の から共有された暗号化されたスナップショットをコピーする場合は AWS アカウント、スナップショットの AWS KMS 暗号化に使用された暗号化キーにアクセスできる必要があります。

共有スナップショットは、スナップショットが暗号化されているかどうかにかかわらず AWS リージョン、同じ 内でのみコピーできます。詳細については、「暗号化の処理」を参照してください。

間でのスナップショットのコピー AWS リージョン

ソーススナップショットの AWS リージョン とは異なる にスナップショットをコピーすると AWS リージョン、各コピーは完全なスナップショットになります。フルスナップショットコピーには、Amazon DocumentDB クラスターを復元するために必要なデータやメタデータのすべてが含まれます。

AWS リージョン 関連するデータとコピーするデータの量によっては、クロスリージョンスナップショットのコピーが完了するまでに数時間かかる場合があります。場合によっては、特定のコピー元 AWS リージョンからの多数のクロスリージョンスナップショットコピーのリクエストが発生することがあります。このような場合、Amazon DocumentDB は、進行中のコピーが完了するまで、そのソースからの新しいクロスリージョンコピーリクエスト AWS リージョン をキューに入れることがあります。コピーリクエストがキューに入っている間は、そのリクエストに関する進捗情報は表示されません。コピーがスタートしたときに、進捗情報は表示されます。

制限

スナップショットをコピーする際の制約は以下のとおりです。

- ターゲットスナップショットが使用可能になる前にソーススナップショットを削除すると、スナップショットはコピーされない場合があります。ターゲットスナップショットのステータスが `AVAILABLE` になったことを確認してから、ソーススナップショットを削除してください。
- アカウントあたり 1 つのコピー先リージョンに対して最大 5 つのスナップショットコピーリクエストを実行できます。

- コピー元とコピー先のリージョンおよびデータのコピー量に応じて、リージョン間のスナップショットのコピーは完了するまでに長時間かかることがあります。詳細については、「間でのスナップショットのコピー AWS リージョン」を参照してください。

暗号化の処理

AWS KMS 暗号化キーを使用して暗号化されたスナップショットをコピーできます。暗号化されたスナップショットをコピーする場合は、スナップショットのコピーも暗号化する必要があります。同じ 内で暗号化されたスナップショットをコピーする場合 AWS リージョン、元のスナップショットと同じ AWS KMS 暗号化キーを使用してコピーを暗号化することも、別の AWS KMS 暗号化キーを指定することもできます。暗号化されたスナップショットをリージョン間でコピーする場合、キーはリージョン固有であるため、ソーススナップショットと同じ AWS KMS 暗号化 AWS KMS キーをコピーに使用することはできません。代わりに、送信先 n AWS リージョンで有効な AWS KMS キーを指定する必要があります。

ソーススナップショットはコピープロセス全体で暗号化されたままになります。詳細については、「Amazon DocumentDB におけるデータ保護」を参照してください。

> **Note**
>
> Amazon DocumentDB クラスタースナップショットの場合、暗号化されていないクラスタースナップショットをコピー時に暗号化することはできません。

パラメータグループに関する考慮事項

リージョン間でスナップショットをコピーすると、コピーにはコピー元の Amazon DocumentDB クラスターで使用されているパラメータグループは含まれません。スナップショットを復元して新しいクラスターを作成すると、そのクラスター AWS リージョン は作成された のデフォルトのパラメータグループを取得します。新しいクラスターにコピー元と同じパラメータを使用するには、以下の操作を行う必要があります。

1. 送信先で AWS リージョン、元のクラスターと同じ設定の Amazon DocumentDB クラスターパラメータグループを作成します。新しい に既に存在する場合は AWS リージョン、そのものを使用できます。
2. コピー先のスナップショットを復元したら AWS リージョン、新しい Amazon DocumentDB クラスターを変更し、前のステップの新規または既存のパラメータグループを追加します。詳細については、「Amazon DocumentDB クラスターの変更」を参照してください。

クラススナップショットのコピー

AWS CLI次のように、 AWS Management Console または を使用して Amazon DocumentDB クラスターをコピーできます。

Using the AWS Management Console

を使用してクラススナップショットのコピーを作成するには AWS Management Console、次の手順を実行します。この手順は、同じ 内またはリージョン間で、暗号化されたクラススナップショット AWS リージョン または暗号化されていないクラススナップショットをコピーするために機能します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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、コピーしたいスナップショットの左にあるボタンを選択します。

   > **Tip**
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. [アクション] メニューから [コピー] を選択します。
4. 結果としての クラススナップショットをコピーする のページで、設定 セクションを完了します。

   a. コピー先リージョン - オプション。クラススナップショットを別の にコピーするには AWS リージョン、送信先リージョン AWS リージョン のスナップショットを選択します。

   b. 新しいスナップショット識別子 - 新しいスナップショットの名前を入力します。

      ターゲットのスナップショットの名前付けの制約:

      - 既存のスナップショットの名前にすることはできません。
      - 長さは、[1 ~ 63] 個の英字、数字、またはハイフンです。
      - 1 字目は英字である必要があります。

- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
- Amazon RDS、Neptune、Amazon DocumentDB のすべてのクラスターで AWS アカウント、リージョンごとに一意である必要があります。

c. タグをコピー - ソーススナップショットに付いているタグをスナップショットコピーにコピーするには、タグをコピー を選択します。

5. [保管時の暗号化] セクションに入力します。

a. 保管時の暗号化 — スナップショットが暗号されていない場合、暗号化されていないスナップショットから暗号化されたコピーは作成できないため、これらのオプションは使用できません。スナップショットが暗号化されている場合は、保管時の暗号化中 AWS KMS key に使用される を変更できます。

スナップショットのコピーの暗号化に関する詳細については、「クラススナップショットの暗号化をコピーする」を参照してください。

保管時の暗号化の詳細については、「Amazon DocumentDB データの保管時の暗号化」を参照してください。

b. AWS KMS キー — ドロップダウンリストから、次のいずれかを選択します。

- (デフォルト) aws/rds — アカウント番号と AWS KMS キー ID は、このオプションの後に表示されます。
- <some-key-name> - キーを作成した場合は、表示され、選択することができます。
- [キー ARN を入力] - [ARN] ボックスに AWS KMS キーの Amazon リソースネーム (ARN) を入力します。ARN の形式は `arn:aws:kms:<region>:<accountID>:key/<key-id>` です。

6. 選択したスナップショットのコピーを作成するには、[スナップショットのコピー] を選択します。または、スナップショットのコピーを作成しない場合は、キャンセルする を選択できます。

Using the AWS CLI

AWS CLIを使って暗号化されていないクラススナップショットのコピーを作成するには、以下のパラメータで `copy-db-cluster-snapshot` のオペレーションを使用します。スナップショットを別の にコピーする場合は AWS リージョン、スナップショットのコピー AWS リージョン 先の で コマンドを実行します。

- **--source-db-cluster-snapshot-identifier** — 必須。コピーを作成するクラスタースナップショットの識別子。クラスターのスナップショットが存在し、利用可能な状態である必要があります。スナップショットを別の にコピーする場合 AWS リージョン、この識別子はソースの ARN 形式である必要があります AWS リージョン。このパラメータは大文字と小文字が区別されません。
- **--target-db-cluster-snapshot-identifier** — 必須。ソースクラスタースナップショットから作成する新しいクラスタースナップショットの識別子。このパラメータは大文字と小文字が区別されません。

  ターゲットのスナップショットの名前付けの制約:

  - 既存のスナップショットの名前にすることはできません。
  - 長さは、[1 ~ 63] 個の英字、数字、またはハイフンです。
  - 1 字目は英字である必要があります。
  - ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
  - Amazon RDS、Neptune、Amazon DocumentDB のすべてのクラスターで AWS アカウント、リージョンごとに一意である必要があります。
- **--source-region** — スナップショットを別の にコピーする場合は AWS リージョン、 AWS リージョン 暗号化されたクラスタースナップショットのコピー元の を指定します。

  スナップショットを別の AWS リージョン にコピーし、`--source-region` を指定しない場合、代わりに `pre-signed-url` のオプションを指定する必要があります。`pre-signed-url` 値は、 AWS リージョン クラスタースナップショットのコピー元のソースで呼び出すCopyDBClusterSnapshotアクションの署名バージョン 4 の署名付きリクエストを含む URL である必要があります。`pre-signed-url` の詳細については、CopyDBClusterSnapshot を参照してください。
- **--kms-key-id** - クラスタースナップショットのコピーを暗号化するのに使用するキーのための KMS キー識別子。

  暗号化されたクラスタースナップショットを別の にコピーする場合は AWS リージョン、このパラメータが必要です。送信先の KMS キーを指定する必要があります AWS リージョン。

  暗号化されたクラスタースナップショットを同じ にコピーする場合 AWS リージョン、 AWS KMS キーパラメータはオプションです。クラスタースナップショットのコピーは、ソースクラスタースナップショットと同じ AWS KMS キーで暗号化されます。コピーの AWS KMS 暗号化に使用する新しい暗号化キーを指定する場合は、このパラメータを使用して指定できます。
- **--copy-tags** - オプション。コピーするタグと値。

コピーオペレーションを進行中にキャンセルするには、DB クラススナップショット が コピー中 ステータスの間に --target-db-cluster-snapshot-identifier または TargetDBClusterSnapshotIdentifier により識別されるターゲットクラススナップショットを削除できます。

Example

例1: 暗号化されていないスナップショットを同一リージョンにコピーする場合

次の AWS CLI 例では、ソーススナップショット AWS リージョン と同じ sample-cluster-snapshot-copyに sample-cluster-snapshotという名前の のコピーを作成します。コピーが作成されると、元のスナップショットのすべてのタグはコピーされたスナップショットにコピーされます。

Linux、macOS、Unix の場合:

```
aws docdb copy-db-cluster-snapshot \
    --source-db-cluster-snapshot-identifier sample-cluster-snapshot \
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy \
    --copy-tags
```

Windows の場合:

```
aws docdb copy-db-cluster-snapshot ^
    --source-db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy ^
    --copy-tags
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBClusterSnapshot": {
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1b",
            "us-east-1c"
        ],
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot-copy",
        "DBClusterIdentifier": "sample-cluster",
        "SnapshotCreateTime": "2020-03-27T08:40:24.805Z",
        "Engine": "docdb",
```

```
        "Status": "copying",
        "Port": 0,
        "VpcId": "vpc-abcd0123",
        "ClusterCreateTime": "2020-01-10T22:13:38.261Z",
        "MasterUsername": "master-user",
        "EngineVersion": "4.0.0",
        "SnapshotType": "manual",
        "PercentProgress": 0,
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:111122223333:key/sample-key-id",
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-
snapshot:sample-cluster-snapshot-copy",
        "SourceDBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-
snapshot:sample-cluster-snapshot"
    }
}
```

Example

例 2: 暗号化されていないスナップショットを にコピーする AWS リージョン

次の AWS CLI 例ではsample-cluster-snapshot、ARN を持つ のコピーを作成しますarn:aws:rds:us-east-1:123456789012:cluster-snapshot:sample-cluster-snapshot。このコピーは という名前sample-cluster-snapshot-copyで、 コマンドが実行される AWS リージョン にあります。

Linux、macOS、Unix の場合:

```
aws docdb copy-db-cluster-snapshot \
    --source-db-cluster-snapshot-identifier arn:aws:rds:us-
east-1:123456789012:cluster-snapshot:sample-cluster-snapshot \
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy
```

Windows の場合:

```
aws docdb copy-db-cluster-snapshot ^
    --source-db-cluster-snapshot-identifier arn:aws:rds:us-
east-1:123456789012:cluster-snapshot:sample-cluster-snapshot ^
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBClusterSnapshot": {
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1b",
            "us-east-1c"
        ],
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot-copy",
        "DBClusterIdentifier": "sample-cluster",
        "SnapshotCreateTime": "2020-04-29T16:45:51.239Z",
        "Engine": "docdb",
        "AllocatedStorage": 0,
        "Status": "copying",
        "Port": 0,
        "VpcId": "vpc-abc0123",
        "ClusterCreateTime": "2020-04-28T16:43:00.294Z",
        "MasterUsername": "master-user",
        "EngineVersion": "4.0.0",
        "LicenseModel": "docdb",
        "SnapshotType": "manual",
        "PercentProgress": 0,
        "StorageEncrypted": false,
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-
snapshot:sample-cluster-snapshot-copy",
        "SourceDBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-
snapshot:sample-cluster-snapshot",
    }
}
```

Example

例 3: 暗号化されたスナップショットを にコピーする AWS リージョン

次の AWS CLI 例では、us-west-2 リージョンsample-cluster-snapshotから us-east-1 リージョンに のコピーを作成します。このコマンドは、us-east-1 リージョンで呼び出されます。

Linux、macOS、Unix の場合:

```
aws docdb copy-db-cluster-snapshot \
    --source-db-cluster-snapshot-identifier arn:aws:rds:us-
west-2:123456789012:cluster-snapshot:sample-cluster-snapshot \
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy \
    --source-region us-west-2 \
```

```
    --kms-key-id sample-us-east-1-key
```

Windows の場合:

```
aws docdb copy-db-cluster-snapshot ^
    --source-db-cluster-snapshot-identifier arn:aws:rds:us-
west-2:123456789012:cluster-snapshot:sample-cluster-snapshot ^
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy ^
    --source-region us-west-2 ^
    --kms-key-id sample-us-east-1-key
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBClusterSnapshot": {
        "AvailabilityZones": [],
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot-copy",
        "DBClusterIdentifier": "ayhu-xrsc-test-ap-southeast-1-small-cluster-kms",
        "SnapshotCreateTime": "2020-04-29T16:45:53.159Z",
        "Engine": "docdb",
        "AllocatedStorage": 0,
        "Status": "copying",
        "Port": 0,
        "ClusterCreateTime": "2020-04-28T16:43:07.129Z",
        "MasterUsername": "chimera",
        "EngineVersion": "4.0.0",
        "LicenseModel": "docdb",
        "SnapshotType": "manual",
        "PercentProgress": 0,
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:111122223333:key/sample-key-id",
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-
snapshot:sample-cluster-snapshot-copy",
        "SourceDBClusterSnapshotArn": "arn:aws:rds:us-west-2:111122223333:cluster-
snapshot:sample-cluster-snapshot",
    }
}
```

> **Note**
>
> スナップショットのコピーの暗号化に関する詳細については、「クラススナップショットの暗号化をコピーする」を参照してください。

保管時の暗号化の詳細については、「Amazon DocumentDB データの保管時の暗号化」を参照してください。

Amazon DocumentDB クラススナップショットの共有

Amazon DocumentDB を使用すると、次の方法で手動クラススナップショットを共有できます。

- 手動クラススナップショットを共有すると、暗号化されているかどうかにかかわらず、承認された AWS アカウントはスナップショットをコピーできます。
- 手動クラススナップショットを共有すると、暗号化されているかどうかにかかわらず、承認された AWS アカウントは、クラスターのコピーを取得して復元するのではなく、スナップショットからクラスターを直接復元できます。

> **Note**
>
> 自動クラススナップショットを共有するには、自動スナップショットをコピーして手動クラススナップショットを作成し、そのコピーを共有します。このプロセスは、 AWS バックアップによって生成されたリソースにも適用されます。

手動スナップショットは、最大 20 の他の と共有できます AWS アカウント。暗号化されていない手動スナップショットをパブリックとして共有することもできます。これにより、このスナップショットをすべての アカウントが使用できるようになります。スナップショットをパブリックとして共有する場合には、パブリックスナップショットにプライベート情報が含まれないように注意してください。

手動スナップショットを他の と共有し AWS アカウント、 AWS CLI または Amazon DocumentDB API を使用して共有スナップショットからクラスターを復元する場合は、共有スナップショットの Amazon リソースネーム (ARN) をスナップショット識別子として指定する必要があります。

暗号化されたスナップショットの共有

以下の制限は、暗号化されたスナップショットの共有に適用されます。

- 暗号化されたスナップショットをパブリックとして共有することはできません。

- スナップショットを共有したアカウントのデフォルトの AWS KMS 暗号化キーを使用して暗号化されたスナップショットを共有することはできません。

暗号化されたスナップショットを共有するには、以下のステップに従います。

1. スナップショットの暗号化に使用された AWS Key Management Service（ AWS KMS) 暗号化キーを、スナップショットにアクセスできるアカウントと共有します。

   AWS KMS 暗号化キーを別の AWS アカウントと共有するには、他のアカウントを AWS KMS キーポリシーに追加します。キーポリシーの更新の詳細については、 AWS Key Management Service デベロッパーガイドの「KMS AWS でのキーポリシーの使用」を参照してください。キーポリシーの作成例については、このトピックで後述する暗号化されているスナップショットのコピーを許可する IAM ポリシーの作成を参照してください。
2. 次に示すように AWS CLI、 を使用して、暗号化されたスナップショットを他のアカウントと共有します。

AWS KMS 暗号化キーへのアクセスを許可する

アカウントから共有された暗号化されたスナップショットを別の がコピー AWS アカウント するには、スナップショットを共有するアカウントが、スナップショットを暗号化した AWS KMS キーにアクセスできる必要があります。別のアカウントに AWS KMS キーへのアクセスを許可するには、キーポリシーのキーポリシーを、 AWS KMS キー AWS KMS ポリシーのプリンシパルとして共有しているアカウントの ARN で更新します。次に、`kms:CreateGrant` アクションを許可します。

AWS KMS 暗号化キーへのアクセス権をアカウントに付与した後、暗号化されたスナップショットをコピーするには、そのアカウントが AWS Identity and Access Management (IAM) ユーザーをまだ作成していない場合は作成する必要があります。さらに、そのアカウントは、ユーザーが AWS KMS キーを使用して暗号化されたスナップショットをコピーできるようにする IAM ポリシーをその IAM ユーザーにアタッチする必要があります。アカウントは IAM ユーザーである必要があり、 AWS KMS セキュリティ上の制限によりルート AWS アカウント ID にすることはできません。

次のキーポリシーの例では、ユーザー 123451234512 が AWS KMS 暗号化キーの所有者です。ユーザー 123456789012 がキーの共有先のアカウントです。この更新されたキーポリシーは、アカウントに AWS KMS キーへのアクセスを許可します。これを行うには、ユーザー 123456789012 のルート AWS アカウント ID の ARN をポリシーのプリンシパルとして含め、 `kms:CreateGrant`アクションを許可します。

JSON

```
{
    "Id": "key-policy-1",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {"AWS": [
                "arn:aws:iam::123451234512:user/KeyUser",
                "arn:aws:iam::123456789012:root"
            ]},
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"},
            {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {"AWS": [
                "arn:aws:iam::123451234512:user/KeyUser",
                "arn:aws:iam::123456789012:root"
            ]},
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
        }
    ]
}
```

暗号化されているスナップショットのコピーを許可する IAM ポリシーの作成

外部 AWS アカウント が AWS KMS キーにアクセスできる場合、そのアカウントの所有者は、アカウント用に作成された IAM ユーザーが、その AWS KMS キーで暗号化された暗号化されたスナップショットをコピーすることを許可するポリシーを作成できます。

次の例は、 AWS アカウント 123456789012 の IAM ユーザーにアタッチできるポリシーを示しています。このポリシーにより、IAM ユーザーは、us-west-2 リージョンc989c1dd-a3f2-4a5d-8d96-e793d082ab26の AWS KMS キーで暗号化されたアカウント 123451234512 から共有スナップショットをコピーできます。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey",
                "kms:CreateGrant",
                "kms:RetireGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:123451234512:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"]
        },
        {
            "Sid": "AllowAttachmentOfPersistentResources",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:123451234512:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"],
            "Condition": {
```

```
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

キーポリシーの更新の詳細については、AWS Key Management Service デベロッパーガイド 中の AWS KMSのキーポリシーを使用する を参照してください。

スナップショットの共有

AWS Management Console または を使用して、Amazon DocumentDB 手動クラスタースナップショット (または自動スナップショットのコピー) を共有できます AWS CLI。

Using the AWS Management Console

を使用してスナップショットを共有するには AWS Management Console、次の手順を実行します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[Snapshots] を選択します。
3. 共有する手動スナップショットを選択します。
4. Actions ドロップダウンメニューで、Share を選択します。
5. DB スナップショットを可視化するには、次のいずれかのオプションを選択します。
   - ソースが暗号化されていない場合は、パブリック を選択して、すべての AWS アカウントが手動スナップショットからクラスターを復元できるようにします。または、プライベート を選択して、手動スナップショットからクラスターを復元することを指定した AWS アカウントのみに許可します。

     > **Warning**
     >
     > DB スナップショットの可視性を Public に設定すると、すべての AWS アカウントが手動スナップショットからクラスターを復元し、データにアクセスできます。個

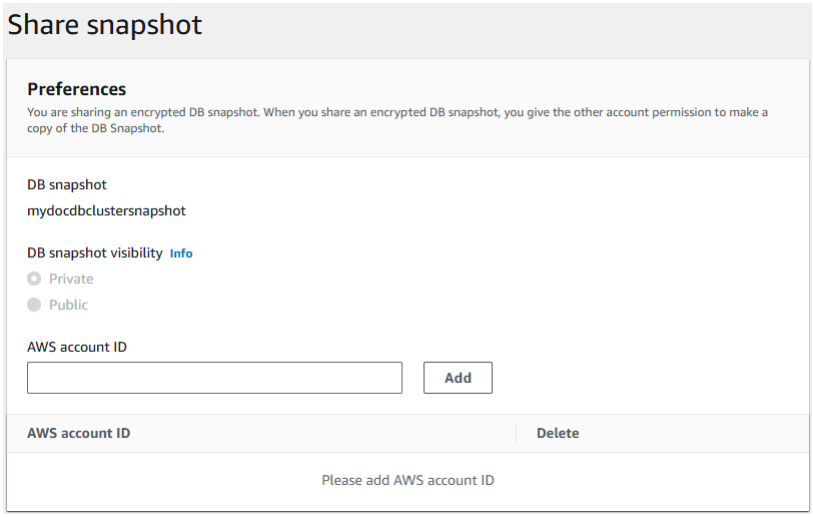
人情報を含む手動クラスタースナップショットをパブリックとして共有しないでください。

- 出典 DB クラスターが暗号化されている場合、暗号化されているスナップショットはパブリックとして共有できないため、[DB snapshot visibility] が [Private] に設定されます。

> **Note**
>
> デフォルトを使用して暗号化されたスナップショットは共有 AWS KMS key できません。

6. AWS アカウント ID に、手動スナップショットからクラスターを復元することを許可するアカウントのアカウント AWS 識別子を入力し、追加を選択します。繰り返して、最大 20 AWS 個のアカウント識別子を追加します AWS 。

   許可された AWS アカウントのリストにアカウント識別子を追加するときにエラーが発生した場合は、間違った AWS アカウント識別子の右側にある「削除」を選択して、リストから削除できます。



7. 手動スナップショットの復元を許可するすべての AWS アカウントの識別子を追加したら、保存を選択して変更を保存します。

Using the AWS CLI

を使用してスナップショットを共有するには AWS CLI、Amazon DocumentDB `modify-db-snapshot-attribute`オペレーションを使用します。`--values-to-add` パラメータを使用し

て、手動スナップショットの復元 AWS アカウント が許可されている の IDs のリストを追加します。

次の例では、 123451234512という名前のスナップショットを復元するために、 と 123456789012 の 2 つの AWS アカウント 識別子を許可しますmanual-snapshot1。また、all 属性値を削除し、スナップショットをプライベートとしてマークします。

Linux、macOS、Unix の場合:

```
aws docdb modify-db-cluster-snapshot-attribute \
    --db-cluster-snapshot-identifier sample-cluster-snapshot \
    --attribute-name restore \
    --values-to-add '["123451234512","123456789012"]'
```

Windows の場合:

```
aws docdb modify-db-cluster-snapshot-attribute ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --attribute-name restore ^
    --values-to-add '["123451234512","123456789012"]'
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512",
                    "123456789012"
                ]
            }
        ]
    }
}
```

リストから AWS アカウント 識別子を削除するには、 --values-to-removeパラメータを使用します。次の例では、 AWS アカウント ID 123456789012 がスナップショットを復元できないようにします。

Linux、macOS、Unix の場合:

```
aws docdb modify-db-cluster-snapshot-attribute \
    --db-cluster-snapshot-identifier sample-cluster-snapshot \
    --attribute-name restore \
    --values-to-remove '["123456789012"]'
```

Windows の場合:

```
aws docdb modify-db-cluster-snapshot-attribute ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --attribute-name restore ^
    --values-to-remove '["123456789012"]'
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512"
                ]
            }
        ]
    }
}
```

クラススナップショットからの復元

Amazon DocumentDB (MongoDB 互換) は、ストレージボリュームのクラススナップショットを作成します。クラススナップショットから復元することで、新しいクラスターを作成できます。クラスターを復元するときは、復元元のクラススナップショットの名前と、復元によって作成される新しいクラスターの名前を指定します。スナップショットから既存のクラスターに復元することはできません。復元時に新しいクラスターが作成されるためです。

クラススナップショットからクラスターを復元する場合:

- このアクションではクラスターのみを復元し、そのクラスターのインスタンスは復元しません。復元されたクラスターのインスタンスを作成するには、`create-db-instance` アクションを呼び出し、復元されたクラスターの識別子を `--db-cluster-identifier` に指定する必要があります。インスタンスを作成できるのは、クラスターが使用可能になった後のみです。
- 暗号化されたスナップショットを暗号化されていないクラスターに復元することはできません。ただし、 AWS KMS キーを指定することで、暗号化されていないスナップショットを暗号化されたクラスターに復元できます。
- 暗号化されたスナップショットからクラスターを復元するには、 AWS KMS キーにアクセスできる必要があります。

> **Note**
>
> 3.6 クラスターを 4.0 クラスターに復元することはできませんが、あるクラスタバージョンから別のクラスタバージョンに移行できます。詳細については、「Amazon DocumentDB への移行」を参照してください。

Using the AWS Management Console

次の手順は、Amazon DocumentDB 管理コンソールを使用して Amazon DocumentDB クラスタースナップショットから Amazon DocumentDB クラスターを復元する方法を示しています。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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、クラスターの復元に使用するスナップショットの左側にあるボタンを選択します。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. [アクション] メニューで、[復元] を選択します。
4. [スナップショットの復元] ページで、[設定] セクションに入力します。

a. クラスター識別子 - 新しいクラスターの名前。Amazon DocumentDB が指定した名前をそのまま使用するか、希望する名前を入力します。[Amazon DocumentDBsupplied] の名前は、docdb- に UTC タイムスタンプを付加した形式 (例: docdb-*yyyy-mm-dd-hh-mm-ss*)です。

b. インスタンスクラス - 新しいクラスターのインスタンスクラス。デフォルトのインスタンスクラスをそのまま使用することも、ドロップダウンリストからインスタンスクラスを選択することもできます。

c. インスタンス数 - このクラスターで作成するインスタンスの数。デフォルトの 3 つのインスタンス (プライマリ読み取り/書き込みレプリカ 1 つ、読み取り専用レプリカ 2 つ) をそのまま使用するか、ドロップダウンリストからインスタンス数を選択できます。

5. [クラスターストレージ設定]では、ストレージオプションを選択します。

> **Note**
>
> [Amazon DocumentDB I/O 最適化]ストレージ設定は、Amazon DocumentDB 5.0 エンジンバージョンでのみ使用できます。

6. クラスター設定に問題がなければ、[クラスターの復元] を選択し、クラスターが復元されるのを待ちます。

7. デフォルトでない Amazon VPC やセキュリティグループの指定など一部の設定を変更する場合は、ページの左下にある [詳細設定の表示] を選択し、以下のステップに進みます。

a. [ネットワーク設定] セクションに入力します。

- 仮想プライベートクラウド (VPC) - 現在の VPC を受け入れるか、ドロップダウンリストから VPC を選択します。
- サブネットグループ - default サブネットグループを使用するか、ドロップダウンリストからいずれかのサブネットグループを選択します。
- VPC セキュリティグループ - default (VPC) セキュリティグループを使用するか、リストからいずれのセキュリティグループを選択します。

b. [クラスターオプション] セクションに入力します。

- データベースポート - デフォルトのポート 27017 を使用するか、上下の矢印を使用してアプリケーション接続に使用するポートを設定します。

c. [暗号化] セクションに入力します。

- 保管時の暗号化 - スナップショットが暗号化されている場合、ここでのオプションは使用できません。暗号化されていない場合は、以下のいずれかを選択できます。
  - すべてのクラスターのデータを暗号化するには、[保管時の暗号化の有効化] を選択します。このオプションを選択した場合は、KMS キーを指定する必要があります。
  - クラスターのデータを暗号化しない場合は、[保管時の暗号化の無効化] を選択します。このオプションを選択した場合、暗号化セクションは完了です。
- AWS KMS キー — ドロップダウンリストから次のいずれかを選択します。
  - (デフォルト) aws/rds — アカウント番号と AWS KMS キー ID は、このオプションの後に表示されます。
  - カスタマーマネージドキー — このオプションは、 AWS Identity and Access Management (IAM) コンソールで IAM 暗号化キーを作成した場合にのみ使用できます。クラスターを暗号化するためのキーを選択できます。
  - キー ARN の入力 — ARN ボックスに、 AWS KMS キーの Amazon リソースネーム (ARN) を入力します。ARN の形式は `arn:aws:kms:<region>:<accountID>:key/<key-id>` です。

d. [ログのエクスポート] セクションに入力します。

- CloudWatch に公開するログの種類を選択する - 以下のいずれかを選択します。
  - 有効 — クラスターで DDL ログ記録を Amazon CloudWatch Logs にエクスポートできるようにします。
  - 無効 — クラスターで DDL ログ記録を Amazon CloudWatch Logs にエクスポートできないようにします。デフォルトは [無効] です。
- IAM ロール - リストから [RDS サービスにリンクされたロール] を選択します。

e. [タグ] セクションに入力します。

- タグの追加 - キー ボックスに、クラスターのためのタグの名前を入力します。値ボックスに、必要に応じてタグ値を入力します。タグは、Amazon DocumentDB リソースへのアクセスを管理し、リソースに適用できるアクションを制御するために AWS Identity and Access Management (IAM) ポリシーとともに使用されます。

f. [削除保護] セクションに入力します。

- 削除保護を有効にする - クラスターが誤って削除されないように保護します。このオプションが有効になっていると、クラスターを削除できません。

8. [クラスターの復元] を選択します。

Using the AWS CLI

を使用してスナップショットからクラスターを復元するには AWS CLI、次のパラメータを指定して `restore-db-cluster-from-snapshot`オペレーションを使用します。詳細については、「RestoreDBClusterFromSnapshot」を参照してください。

- **`--db-cluster-identifier`** — 必須。オペレーションによって作成されるクラスターの名前。この名前のクラスターは、このオペレーションの前に存在することはできません。

  クラスターの名前付けの制約:

  - 長さは [1 ～ 63] 個の英字、数字、またはハイフンです。
  - 1 字目は英字である必要があります。
  - ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
  - Amazon RDS、Neptune、Amazon DocumentDB のすべてのクラスターで AWS アカウント、リージョンごとに一意である必要があります。
- **`--snapshot-identifier`** — 必須。復元前に使用されたスナップショットの名前。この名前のスナップショットが存在し、利用可能な状態である必要があります。
- **`--engine`** — 必須。`docdb` を指定してください。
- **`--storage-type standard | iopt1`** - オプション。デフォルト: `standard`。
- **`--kms-key-id`** - オプション。暗号化されたスナップショットを復元するとき、または暗号化されていないスナップショットから復元するときにクラスターを暗号化するときに使用する AWS KMS キー識別子の ARN。キー ID を指定すると AWS KMS 、スナップショットが暗号化されたかどうかにかかわらず、復元されたクラスターが AWS KMS キーで暗号化されます。

  `--kms-key-id` の形式は `arn:aws:kms:<region>:<accountID>:key/<key-id>` です。`--kms-key-id` パラメータの値を指定しない場合は、以下のようになります。

  - のスナップショット`--snapshot-identifier`が暗号化されている場合、復元されたクラスターは、スナップショットの暗号化に使用されたのと同じ AWS KMS キーを使用して暗号化されます。
  - `--snapshot-identifier` のスナップショットが暗号化されていない場合、復元されたクラスターは暗号化されません。

Linux、macOS、Unix の場合:

```
aws docdb restore-db-cluster-from-snapshot \
    --db-cluster-identifier sample-cluster-restore \
```

```
    --snapshot-identifier sample-cluster-snapshot \
    --engine docdb \
    --kms-key-id arn:aws:kms:us-east-1:123456789012:key/SAMPLE-KMS-KEY-ID
```

Windows の場合:

```
aws docdb restore-db-cluster-from-snapshot ^
    --db-cluster-identifier sample-cluster-restore ^
    --snapshot-identifier sample-cluster-snapshot ^
    --engine docdb ^
    --kms-key-id arn:aws:kms:us-east-1:123456789012:key/SAMPLE-KMS-KEY-ID
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBCluster": {
        "AvailabilityZones": [
            "us-east-1c",
            "us-east-1b",
            "us-east-1a"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "sample-cluster-restore",
        "DBClusterParameterGroup": "default.docdb4.0",
        "DBSubnetGroup": "default",
        "Status": "creating",
        "Endpoint": "sample-cluster-restore.cluster-node.us-
east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster-restore.cluster-node.us-
east-1.docdb.amazonaws.com",
        "MultiAZ": false,
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
        "Port": 27017,
        "MasterUsername": "<master-user>",
        "PreferredBackupWindow": "02:00-02:30",
        "PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
        "DBClusterMembers": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcdefgh",
                "Status": "active"
            }
```

```
        ],
        "HostedZoneId": "ABCDEFGHIJKLM",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/<sample-key-id>",
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-cluster-
restore",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2020-04-01T01:43:40.871Z",
        "DeletionProtection": true
    }
}
```

クラスターのステータスが [使用可能] になったら、そのクラスターに 1 つ以上のインスタンスを作成します。

Linux、macOS、Unix の場合:

```
aws docdb create-db-instance \
    --db-cluster-identifier sample-cluster-restore  \
    --db-instance-identifier sample-cluster-restore-instance \
    --availability-zone us-east-1b \
    --promotion-tier 2 \
    --db-instance-class db.r5.large \
    --engine docdb
```

Windows の場合:

```
aws docdb create-db-instance ^
    --db-cluster-identifier sample-cluster-restore  ^
    --db-instance-identifier sample-cluster-restore-instance ^
    --availability-zone us-east-1b ^
    --promotion-tier 2 ^
    --db-instance-class db.r5.large ^
    --engine docdb
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "sample-cluster-restore-instance",
        "DBInstanceClass": "db.r5.large",
```

```
        "Engine": "docdb",
        "DBInstanceStatus": "creating",
        "PreferredBackupWindow": "02:00-02:30",
        "BackupRetentionPeriod": 1,
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcdefgh",
                "Status": "active"
            }
        ],
        "AvailabilityZone": "us-west-2b",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-6242c31a",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-abcdefgh",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    ...
                }
            ]
        },
        "PreferredMaintenanceWindow": "fri:09:43-fri:10:13",
        "PendingModifiedValues": {},
        "EngineVersion": "4.0.0",
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "DBClusterIdentifier": "sample-cluster-restore",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/<sample-key-id>",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "CACertificateIdentifier": "rds-ca-2019",
        "PromotionTier": 2,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-cluster-
restore-instance"
    }
```

```
}
```

特定時点への復元

AWS Management Console または AWS Command Line Interface () を使用して、クラスターをクラスターのバックアップ保持期間内の任意の時点に復元できますAWS CLI。

> **Note**
>
> 3.6 クラスターの 4.0 クラスターへの特定の時点への復元は実行できませんが、あるクラスターバージョンから別のクラスタバージョンに移行できます。詳細については、「Amazon DocumentDB への移行」を参照してください。

クラスターを特定時点に復元するときには、以下の点を考慮します。

- 新しいクラスターは、デフォルトのパラメータグループを使用して作成されることを除いて、ソースクラスターと同じ設定で作成されます。新しいクラスターのパラメータグループをソースクラスターのパラメータグループに設定するには、ステータスが [使用可能] になってからクラスターを変更します。クラスターの変更の詳細については、「Amazon DocumentDB クラスターの変更」を参照してください。

Using the AWS Management Console

AWS Management Consoleを使用して以下を完了すると、クラスターをバックアップ保持期間内の特定の時点に復元できます。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。クラスターのリストで、復元するクラスターの左側にあるボタンを選択します。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン

(≡)
を選択します。

3. [アクション] メニューで、[特定時点への復元] を選択します。
4. [復元時刻] セクションに情報を入力して、復元する日時を指定します。

   a. 復元日 - [最新復元時刻] から [最後の復元時刻] の範囲の日付を選択または入力します。

   b. 復元時刻 - [最新復元時刻] から [最後の復元時刻] の範囲の日付で、時分秒を選択または入力します。
5. [設定] セクションに情報を入力します。

   a. クラスター識別子 - デフォルトの識別子を使用するか、任意の識別子を入力します。

      クラスターの名前付けの制約:

      - 長さは [1 ～ 63] 個の英字、数字、またはハイフンです。
      - 1 字目は英字である必要があります。
      - ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
      - Amazon RDS、Neptune、Amazon DocumentDB のすべてのクラスターで AWS アカウント、リージョンごとに一意である必要があります。

   b. インスタンスクラス - ドロップダウンリストで、クラスターのインスタンスに使用するインスタンスクラスを選択します。

   c. インスタンス数 - ドロップダウンリストで、クラスター復元時に作成するインスタンスの数を選択します。
6. [クラスターストレージ設定]では、ストレージオプションを選択します。

   Note

   [Amazon DocumentDB I/O 最適化]ストレージ設定は、Amazon DocumentDB 5.0 エンジンバージョンでのみ使用できます。
7. オプション。ネットワーク設定とクラスターオプションを設定し、ログのエクスポートを有効にするには、[詳細設定の表示] を選択し、次のセクションに入力します。それ以外の場合は、次のステップに進みます。

   - ネットワーク設定

1. 仮想プライベートクラウド（VPC）- ドロップダウンリストから、このクラスターで使用したい VPC を選択します。
2. サブネットグループ - ドロップダウンリストから、このクラスターのサブネットグループを選択します。
3. VPC セキュリティグループ - ドロップダウンリストから、このクラスターの VPC セキュリティグループを選択します。

- クラスターオプション
  1. ポート - デフォルトのポート (27017) を使用するか、上向き矢印と下向き矢印を使用して、このクラスターと通信するポートを設定します。

- ログのエクスポート
  1. 監査ログ - このオプションを選択し、Amazon CloudWatch Logs への監査ログを有効にします。このオプションを選択した場合は、クラスターのカスタムパラメータグループで `audit_logs` を有効にする必要があります。詳細については、「Amazon DocumentDB イベントの監査」を参照してください。
  2. プロファイラーログ - オペレーションのプロファイラーログの Amazon CloudWatch Logs へのエクスポートを有効にするには、このオプションを選択します。このオプションを選択した場合は、クラスターのカスタムパラメータグループの次のパラメータも変更する必要があります。
     - `profiler` — `enabled` に設定します。
     - `profiler_threshold_ms` - `[0-INT_MAX]` の値に設定して、プロファイリングオペレーションのしきい値を設定します。
     - `profiler_sampling_rate` - プロファイルする低速オペレーションの割合を設定するには、`[0.0-1.0]` の値に設定します。

     詳細については、「Amazon DocumentDB オペレーションのプロファイリング」を参照してください。
  3. プロファイラーログ - プロファイラーログを Amazon CloudWatch にエクスポートします
  4. IAM ロール - ドロップダウンリストから RDS サービスにリンクされたロール を選択します。

- [タグ]

1. タグの追加 - キー ボックスに、クラスターのためのタグの名前を入力します。値ボックスに、必要に応じてタグ値を入力します。タグは、 AWS Identity and Access Management （ IAM ）のポリシーで使用され、Amazon DocumentDB リソースへのアクセスを管理したり、 リソースに適用できるアクションを制御したりできます。

- 削除保護

1. 削除保護を有効にする - クラスターが誤って削除されないように保護します。このオプションが有効になっていると、クラスターを削除できません。

8. クラスターを復元するには、[クラスターの作成] を選択します。または、[キャンセルする] を選択してオペレーションをキャンセルすることもできます。

Using the AWS CLI

スナップショットのバックアップ保持期間を使用して特定時点にクラスターを復元するには、以下のパラメータを指定して `restore-db-cluster-to-point-in-time` オペレーションを使用します。

- **`--db-cluster-identifier`** - 必須。作成する新しいクラスターの名前。このクラスターは、オペレーションの前に存在することはできません。パラメータ値は以下の制約を満たす必要があります。

  クラスターの名前付けの制約:

  - 長さは [1 ～ 63] 個の英字、数字、またはハイフンです。
  - 1 字目は英字である必要があります。
  - ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
  - Amazon RDS、Neptune、Amazon DocumentDB のすべてのクラスターで AWS アカウント、リージョンごとに一意である必要があります。

- **`--restore-to-time`** - クラスターを復元する UTC 日時。例えば、`2018-06-07T23:45:00Z`。

  時間の制約:

  - クラスターの復元可能な最新時刻より前である必要があります。
  - `--use-latest-restorable-time` パラメータを指定されていない場合は、指定する必要があります。
  - `--use-latest-restorable-time` パラメータが `true` である場合は、指定することはできません。

  - --restore-type パラメータ値が copy-on-write である場合は、指定することはできません。
- **--source-db-cluster-identifier** - 復元するソースクラスターの名前。このクラスターは存在していて、使用可能である必要があります。
- **--use-latest-restorable-time** または **--no-use-latest-restorable-time** - 最新の復元可能なバックアップ時間に復元するかどうか。--restore-to-time パラメータを使用する場合は、指定することができません。
- **--storage-type standard | iopt1** - オプション。デフォルト: standard。

AWS CLI オペレーションはクラスターのみをrestore-db-cluster-to-point-in-time復元し、そのクラスターのインスタンスは復元しません。復元されたクラスターのインスタンスを作成するには、create-db-instance オペレーションを呼び出し、復元されたクラスターの識別子を --db-cluster-identifier に指定する必要があります。インスタンスは、restore-db-cluster-to-point-in-time オペレーションを完了し、復元したクラスターが 使用可能になってから作成できます。

Example

次の例では、スナップショット sample-cluster-snapshot から復元可能な最新時刻に sample-cluster-restored を作成します。

Linux、macOS、Unix の場合:

```
aws docdb restore-db-cluster-to-point-in-time \
    --db-cluster-identifier sample-cluster-restored \
    --source-db-cluster-identifier sample-cluster-snapshot \
    --use-latest-restorable-time
```

Windows の場合:

```
aws docdb restore-db-cluster-to-point-in-time ^
    --db-cluster-identifier sample-cluster-restored ^
    --source-db-cluster-identifier sample-cluster-snapshot ^
    --use-latest-restorable-time
```

Example

次の例では、スナップショット sample-cluster-snapshot から、2018 年 12 月 11 日 03:15 (UTC) への sample-cluster-restored を作成します。これは、 sample-cluster のバックアップ保持期間範囲内です。

Linux、macOS、Unix の場合:

```
aws docdb restore-db-cluster-to-point-in-time \
    --db-cluster-identifier sample-cluster-restore \
    --source-db-cluster-identifier sample-cluster \
    --restore-to-time 2020-05-12T03:15:00Z
```

Windows の場合:

```
aws docdb restore-db-cluster-to-point-in-time ^
    --db-cluster-identifier sample-cluster-restore ^
    --source-db-cluster-identifier sample-cluster ^
    --restore-to-time 2020-05-12T03:15:00Z
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBCluster": {
        "AvailabilityZones": [
            "us-east-1c",
            "us-west-2b",
            "us-west-2a"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "sample-cluster-restored",
        "DBClusterParameterGroup": "sample-parameter-group",
        "DBSubnetGroup": "default",
        "Status": "creating",
        "Endpoint": "sample-cluster-restored.node.us-east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster-restored.node.us-
east-1.docdb.amazonaws.com",
        "MultiAZ": false,
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
        "Port": 27017,
        "MasterUsername": "master-user",
        "PreferredBackupWindow": "02:00-02:30",
```

```
        "PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
        "DBClusterMembers": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abc0123",
                "Status": "active"
            }
        ],
        "HostedZoneId": "ABCDEFGHIJKLM",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID^>:key/sample-key",
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-cluster-
restored",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2020-04-24T20:14:36.713Z",
        "DeletionProtection": false
    }
}
```

クラススナップショットの削除

手動スナップショットはフルバックアップであり、 AWS Management Console または を使用して手動で削除した場合にのみ削除されます AWS CLI。自動スナップショットを手動で削除することはできません。自動スナップショットが削除されるのは、スナップショットの保持期間が終了したときか、スナップショットのクラスターを削除したときのみです。

Using the AWS Management Console

を使用して手動クラススナップショットを削除するには AWS Management Console、次の手順を実行します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[Snapshots] を選択してください。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン

(≡)
を選択します。

3. スナップショットのリストで、削除するスナップショットの左側にあるボタンを選択します。スナップショットのタイプは [手動] であることが必要です。

   1. スナップショットのタイプが、[タイプ] 列の下で `manual` または `automatic` としてリストされているかどうかチェックすることによって、手動 であることを確認できます。

4. [アクション] メニューから、[削除] を選択します。[削除] オプションが使用できない場合、自動スナップショットを選択した可能性があります。
5. 削除の確認画面でスナップショットを削除するには、[削除] を選択します。スナップショットを保持するには、[キャンセルする] を選択します。

Using the AWS CLI

Amazon DocumentDB の手動クラスタースナップショットは、 AWS CLIを使用して手動で削除することができる完全なバックアップです。自動スナップショットを手動で削除することはできません。

を使用して手動クラスタースナップショットを削除するには AWS CLI、以下のパラメータを指定して `delete-db-cluster-snapshot`オペレーションを使用します。

パラメータ

- **`--db-cluster-snapshot-identifier`** — 必須。削除する手動スナップショットの名前。

以下の例では、クラスタースナップショット `sample-cluster-snapshot` を削除します。

Linux、macOS、Unix の場合:

```
aws docdb delete-db-cluster-snapshot \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Windows の場合:

```
aws docdb delete-db-cluster-snapshot ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

このオペレーションからの出力には、削除したクラスタースナップショットの詳細が一覧表示されます。

Amazon DocumentDB リソースの管理

以下のセクションでは、Amazon DocumentDB (MongoDB 互換) の実装を管理するためのさまざまなコンポーネントと関連タスクについて説明します。

トピック

- Amazon DocumentDB 運用タスクの概要
- Amazon DocumentDB グローバルクラスターの概要
- Amazon DocumentDB クラスターの管理
- Amazon DocumentDB インスタンスの管理
- Amazon DocumentDB サブネットグループの管理
- Amazon DocumentDB 高可用性とレプリケーション
- Amazon DocumentDB インデックスの管理
- コレクションレベルのドキュメント圧縮の管理
- Amazon DocumentDB イベントの管理
- リージョンとアベイラビリティーゾーンの選択
- Amazon DocumentDB クラスターパラメータグループの管理
- Amazon DocumentDB エンドポイントについて
- Amazon DocumentDB Amazon リソースネーム (ARN) の理解
- Amazon DocumentDB リソースへのタグ付け
- Amazon DocumentDB のメンテナンス
- サービスにリンクされたロールを理解する
- Amazon DocumentDB を用いた変更ストリームの使用
- 変更ストリーム AWS Lambda での の使用

Amazon DocumentDB 運用タスクの概要

このセクションでは、Amazon DocumentDB クラスターでの運用タスクと、 AWS CLIを使用してこれらのタスクを完了する方法について説明します。

トピック

- Amazon DocumentDB クラスターにレプリカを追加する
- クラスターとインスタンスの説明
- クラスタースナップショットの作成
- スナップショットからの復元
- クラスターからインスタンスを削除する
- クラスターの削除

Amazon DocumentDB クラスターにレプリカを追加する

Amazon DocumentDB クラスターのプライマリインスタンスを作成した後、1 つまたは複数の レプリカ を追加することができます。レプリカは、2 つの目的に対応する読み取り専用インスタンスです。

- スケーラビリティ - 同時アクセスを必要とするクライアントが多数ある場合は、レプリカを追加して読み取りをスケーリングできます。
- 高可用性 - プライマリインスタンスに障害が発生した場合、Amazon DocumentDB は自動的にレプリカインスタンスにフェイルオーバーし、それを新しいプライマリとして指定します。レプリカに障害が発生しても、障害が発生したノードが回復するまで、クラスター内の他のインスタンスがリクエストを処理することができます。

各 Amazon DocumentDB クラスターは、最大 15 個のレプリカをサポートできます。

> **Note**
>
> 耐障害性を最大にするため、レプリカを異なるアベイラビリティーゾーンにデプロイする必要があります。これにより、アベイラビリティーゾーン全体が使用できなくなった場合でも Amazon DocumentDB クラスターが機能し続けることができます。

次の AWS CLI 例は、新しいレプリカを追加する方法を示しています。`--availability-zone` パラメータは、指定されたアベイラビリティーゾーンにレプリカを配置します。

```
aws docdb create-db-instance \
    --db-instance-identifier sample-instance \
    --db-cluster-identifier sample-cluster \
    --engine docdb \
```

```
    --db-instance-class db.r5.large \
    --availability-zone us-east-1a
```

クラスターとインスタンスの説明

次の の AWS CLI 例では、リージョン内のすべての Amazon DocumentDB クラスターを一覧表示します。クラスターやインスタンスのライフサイクル管理などの特定の管理機能において、Amazon DocumentDB は Amazon RDS と共有される運用テクノロジーを使用します。`filterName=engine,Values=docdb` フィルターパラメータは Amazon DocumentDB クラスターのみを返します。

クラスターの説明と変更の詳細については、「Amazon DocumentDB クラスターのライフサイクル」を参照してください。

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBClusters": [
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-1",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        },
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
```

```
            "DBClusterIdentifier": "sample-cluster-2",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        },
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-3",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        }
    ]
}
```

次の の AWS CLI 例では、Amazon DocumentDB クラスター内のインスタンスを一覧表示します。クラスターの説明と変更の詳細については、「Amazon DocumentDB インスタンスのライフサイクル」を参照してください。

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterMembers]'
```

出力は次のようになります。この出力には、2 つのインスタンスがあります。プライマリインスタンスは sample-instance-1 ("IsClusterWriter": true) です。レプリカインスタンス sample-instance2 ("IsClusterWriter: false") もあります。

```
[
    [
        [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
```

```
            },
            {
                "DBInstanceIdentifier": "sample-cluster-2",
                "IsClusterWriter": false,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ]
    ]
]
```

クラススナップショットの作成

クラススナップショット は、Amazon DocumentDB クラスター内のデータの完全なバックアップです。スナップショットが作成されるときに、Amazon DocumentDB はクラスターボリュームから直接データを読み取ります。このため、その時点で実行中のインスタンスがクラスターにない場合でも、スナップショットを作成できます。スナップショットを作成するためにかかる時間は、クラスターボリュームサイズによって異なります。

Amazon DocumentDB は自動バックアップをサポートしており、バックアップウィンドウ (日中の 30 分間) に毎日実行されます。次の AWS CLI 例は、クラスターのバックアップウィンドウを表示する方法を示しています。

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].PreferredBackupWindow'
```

出力には、バックアップウィンドウ (UTC) が表示されています。

```
[
    "00:18-00:48"
]
```

Amazon DocumentDB クラスターを作成するときに、バックアップウィンドウを定義することができます。次の例に示すように、バックアップウィンドウを変更することもできます。バックアップウィンドウを定義しない場合、Amazon DocumentDB は自動的にクラスターに 1 つを割り当てます。

```
aws docdb modify-db-cluster \
```

```
    --db-cluster-identifier sample-cluster \
    --preferred-backup-window "02:00-02:30"
```

自動バックアップに加えて、いつでもクラスタースナップショットを手動で作成できます。この操作を行うときは、後で復元できるように、バックアップするクラスターと、スナップショットの一意の名前を指定します。

次の AWS CLI 例は、データのスナップショットを作成する方法を示しています。

```
aws docdb create-db-cluster-snapshot \
    --db-cluster-identifier sample-cluster \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

スナップショットからの復元

クラスタースナップショットを新しい Amazon DocumentDB クラスターに復元することができます。これを行うには、スナップショットの名前と新しいクラスターの名前を指定します。スナップショットから既存のクラスターに復元することはできません。代わりに、Amazon DocumentDB は復元時に新しいクラスターを作成してスナップショットデータを入力します。

次の例は、クラスター `sample-cluster` のすべてのスナップショットを示しています。

```
aws docdb describe-db-cluster-snapshots \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusterSnapshots[*].[DBClusterSnapshotIdentifier,SnapshotType,Status]'
```

出力は次のようになります。手動スナップショットは手動で作成したものであり、自動スナップショットは、クラスターのバックアップウィンドウ内で Amazon DocumentDB によって作成されたものです。

```
[
        "sample-cluster-snapshot",
        "manual",
        "available"
    ],
    [
        "rds:sample-cluster",
        "automated",
        "available"
    ]
```

```
]
```

次の例は、スナップショットから Amazon DocumentDB クラスターを復元する方法を示しています。

```
aws docdb restore-db-cluster-from-snapshot \
    --engine docdb \
    --db-cluster-identifier new-sample-cluster \
    --snapshot-identifier sample-cluster-snapshot
```

新しいクラスターには関連付けられたインスタンスがありません。このクラスターを操作するには、インスタンスを追加する必要があります。

```
aws docdb create-db-instance \
    --db-instance-identifier new-sample-instance \
    --db-instance-class db.r5.large \
    --engine docdb \
    --db-cluster-identifier new-sample-cluster
```

次の AWS CLI オペレーションを使用して、クラスターとインスタンスの作成の進行状況をモニタリングできます。クラスターとインスタンスのステータスが利用可能になったら、新しいクラスターのエンドポイントに接続し、データにアクセスできます。

```
aws docdb describe-db-clusters \
    --db-cluster-identifier new-sample-cluster  \
    --query 'DBClusters[*].[Status,Endpoint]'
```

```
aws docdb describe-db-instances \
    --db-instance-identifier new-sample-instance \
    --query 'DBInstances[*].[DBInstanceStatus]'
```

クラスターからインスタンスを削除する

Amazon DocumentDB はクラスターボリューム内のすべてのデータを保存します。クラスターからすべてのインスタンスを削除しても、そのクラスターボリュームのデータは保持されます。データに再アクセスする必要がある場合は、いつでもクラスターにインスタンスを追加し、中断した個所から再開できます。

次の例は、Amazon DocumentDB クラスターからインスタンスを削除する方法を示しています。

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-instance
```

クラスターの削除

Amazon DocumentDB クラスターを削除する前に、最初にそのすべてのインスタンスを削除する必要があります。次の の AWS CLI 例では、クラスター内のインスタンスに関する情報を返します。このオペレーションでインスタンス識別子が返される場合は、各インスタンスを削除する必要があります。詳細については、「クラスターからインスタンスを削除する」を参照してください。

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterMembers[*].DBInstanceIdentifier'
```

その他のインスタンスが残っていない場合は、クラスターを削除できます。そのときに、次のいずれかのオプションを選択する必要があります。

- 最終スナップショットの作成 - すべてのクラスターデータをスナップショットで取得し、後でそのデータで新しいインスタンスを再作成できるようにします。次の例は、その方法を示しています。

  ```
  aws docdb delete-db-cluster \
      --db-cluster-identifier sample-cluster \
      --final-db-snapshot-identifier sample-cluster-snapshot
  ```

- 最終スナップショットをスキップする - すべてのクラスターデータを永久に破棄します。この操作は取り消すことができません。次の例は、その方法を示しています。

  ```
  aws docdb delete-db-cluster \
      --db-cluster-identifier sample-cluster \
      --skip-final-snapshot
  ```

Amazon DocumentDB グローバルクラスターの概要

グローバルクラスターとは

グローバルクラスターは、1 つのプライマリリージョンと最大 10 の読み取り専用セカンダリリージョンで構成されます。書き込みオペレーションを、プライマリリージョンのプライマリクラスター

に直接行うと、Amazon DocumentDB 専用のインフラストラクチャを使用してセカンダリリージョンにデータを自動的にレプリケートします。レイテンシーは通常 1 秒未満です。

グローバルクラスターはどのように役立つのか

- リージョン全体の停止からのリカバリー - リージョン全体の停止が発生した場合、セカンダリクラスターの 1 つを数分以内にプライマリクラスターに昇格できます。通常、目標復旧時間 (RTO) は 1 分未満です。通常、目標復旧時点 (RPO) は秒単位で測定されますが、これは障害発生時のネットワーク経由の遅延によって異なります。
- ローカルレイテンシーによるグローバルな読み取り - 世界中にオフィスを持つ企業は、グローバルクラスターを使用することで、自社の主な情報源をプライマリリージョンで最新状態に保つことができます。他のリージョンにあるオフィスは、自社のリージョンにある情報にローカルのレイテンシーでアクセスすることができます。
- スケーラブルなセカンダリクラスター - 読み取り専用のインスタンス をセカンダリリージョンにさらに追加することでセカンダリクラスターは、スケールできます。セカンダリクラスターは読み取り専用です。したがって、読み取り専用のレプリカインスタンスを、1 つの クラスターにつき、通常の 15 件ではなく最大 16 件サポートします。
- プライマリからセカンダリクラスターへの迅速なレプリケーション - グローバルクラスターによるレプリケーションは、プライマリデータベースクラスターのパフォーマンスにほとんど影響しません。DB インスタンスのリソースは、全面的にアプリケーションの読み取りおよび書き込みワークロードに当てられます。

グローバルクラスターの現在の制限は何ですか。

- グローバルクラスターは Amazon DocumentDB v3.6 ではサポートされていません。
- グローバルクラスターは、t3、t4g と r4 インスタンスタイプではサポートされていません。
- グローバルクラスターは、南米 (サンパウロ) 、欧州 (ミラノ) 、中国 (北京) 、中国 (寧夏) の各リージョンでは使用できません。
- リージョンが異なるエンジンバージョンにある場合、スイッチオーバーとグローバルフェイルオーバーはサポートされていません。エンジンのバージョンが一致しない場合、手動フェイルオーバーがサポートされます。
- 書き込みオペレーションはプライマリクラスターのみが実行します。書き込みオペレーションを実行するクライアントは、プライマリクラスターの クラスターエンドポイントに接続します。
- クラスターには、最大 10 個のセカンダリリージョンと 1 つのプライマリリージョンを設定できます。

- セカンダリクラスターは停止できません。プライマリクラスターにセカンダリクラスターが関連付けられている場合、プライマリクラスターを停止できません。セカンダリクラスターを持たないリージョナルクラスターのみを停止できます。
- セカンダリクラスターにアタッチされたレプリカは、特定の場合に再起動することが可能です。プライマリリージョンのインスタンスが再起動またはフェイルオーバーすると、そのセカンダリリージョンにあるレプリカも再起動します。このクラスターは、その後すべてのレプリカがプライマリデータベースのクラスターのライターインスタンスと同期するまでは使用できません。この動作は想定されるものです。プライマリクラスターに変更を加えるときは、必ず事前に、グローバルクラスターへの影響を把握してください。
- セカンダリクラスターでは変更ストリームを使用できません。

トピック

- クイックスタートガイド: グローバルクラスター
- Amazon DocumentDB グローバルクラスターの管理
- Amazon DocumentDB グローバルクラスターに接続する
- Amazon DocumentDB グローバルクラスターのモニタリング
- ディザスタリカバリと Amazon DocumentDB グローバルクラスターについて

クイックスタートガイド: グローバルクラスター

トピック

- 設定
- Amazon DocumentDB グローバルクラスターの作成
- Amazon DocumentDB グローバルクラスター AWS リージョン への の追加
- Amazon DocumentDB グローバルクラスターのスナップショットの使用

設定

Amazon DocumentDB グローバルクラスターは、少なくとも 2 つにまたがります AWS リージョン。プライマリリージョンは、1 つのプライマリ (ライター) インスタンスと最大 15 個のレプリカインスタンスを持つクラスターをサポートします。セカンダリリージョンは、最大 16 個のレプリカインスタンスで構成される読み取り専用クラスターを実行します。グローバルクラスターには、最大 5 つのセカンダリリージョンを含めることができます。この表は、グローバルクラスターで許容されているクラスター、インスタンス、レプリカの最大数をリスト化したものです。

説明	プライマリ AWS リージョン	セカンダリ AWS リージョン
クラスター	1	5 (最大)
ライターインスタンス	1	0
読み取り専用インスタンス (Amazon DocumentDB レプリカ)、クラスターごと	15 (最大数)	16 (総数)
読み取り専用インスタンス (最大許容数、指定されたセカンダリリージョンの実数)	15 - s	s = セカンダリの総数 AWS リージョン

クラスターには、以下の固有の要件があります。

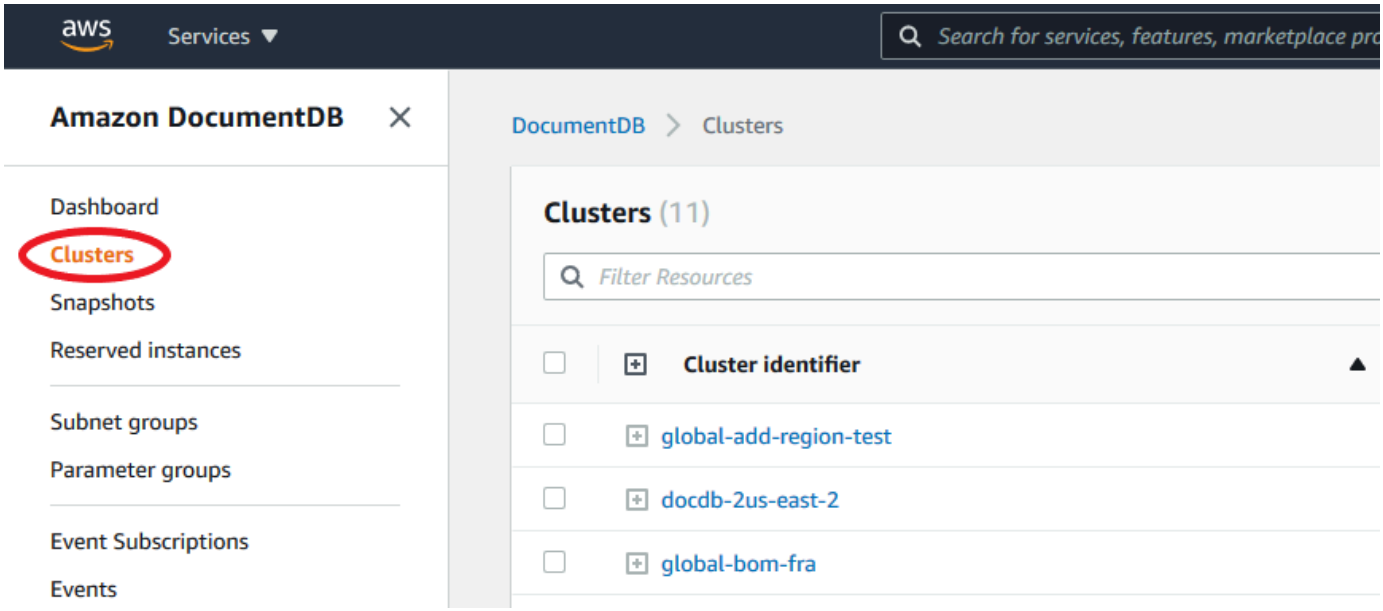
- データベースインスタンスクラスの要件 — 使用できるのは `db.r5` と`db.r6g` インスタンスクラスのみです。
- AWS リージョン 要件 — プライマリクラスターは 1 つのリージョンにあり、少なくとも 1 つのセカンダリクラスターは同じアカウントの別のリージョンにある必要があります。最大 5 つのセカンダリ (読み取り専用) クラスターを作成できます。それぞれが異なるリージョンに存在する必要があります。つまり、同じリージョンに 2 つのクラスターを含めることはできません。
- 命名要件 — 各クラスターに選択する名前は、すべてのリージョンで一意である必要があります。異なるリージョンにあるクラスターでも、同じ名前を使用することはできません。

Amazon DocumentDB グローバルクラスターの作成

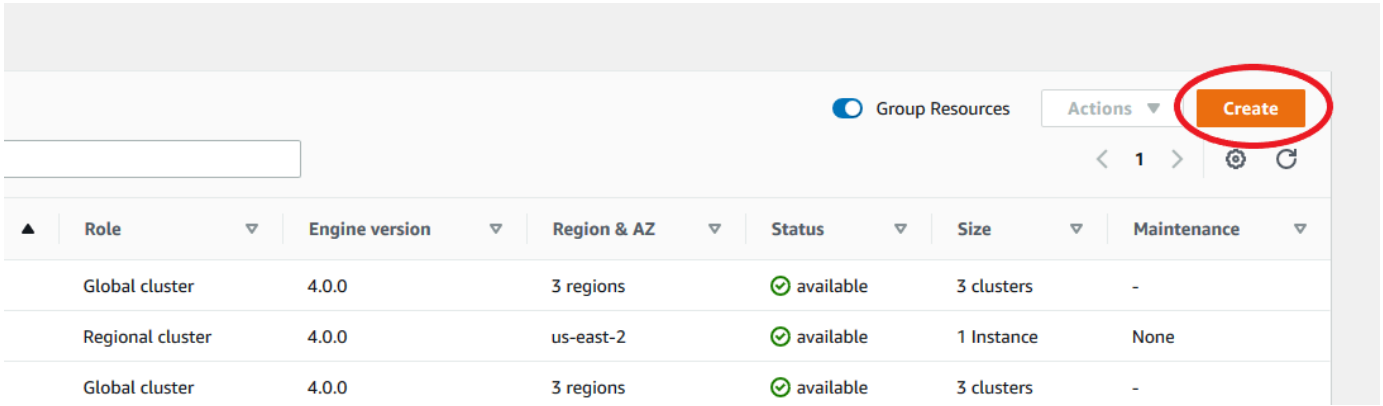
最初のグローバルクラスターを構築する準備はできていますか。このセクションでは、 AWS Management Console または を使用して、 AWS CLI 新しいデータベースクラスターとインスタンスを使用して新しいグローバルクラスターを作成する方法について説明します。

の使用 AWS Management Console

1. で AWS Management Console、Amazon DocumentDB に移動します。
2. Amazon DocumentDB コンソールにアクセスしたら、[クラスター] を選択します。

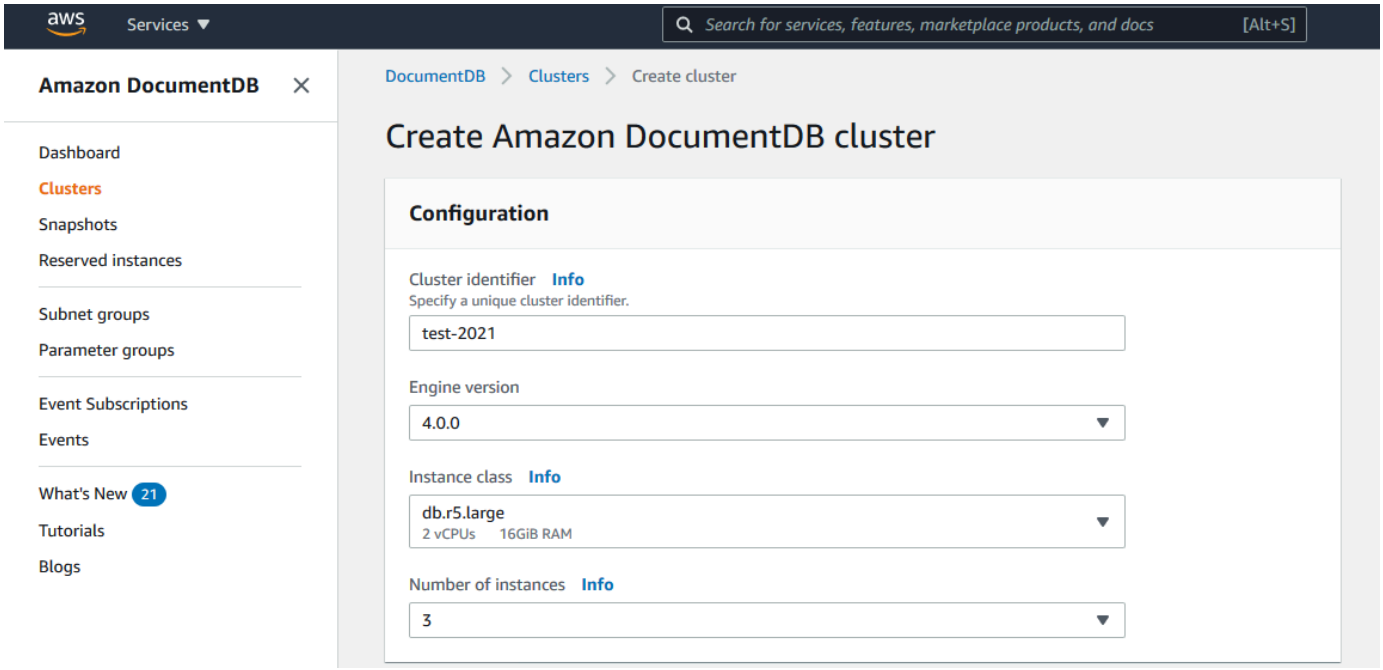


3. [Create] (作成) を選択します。

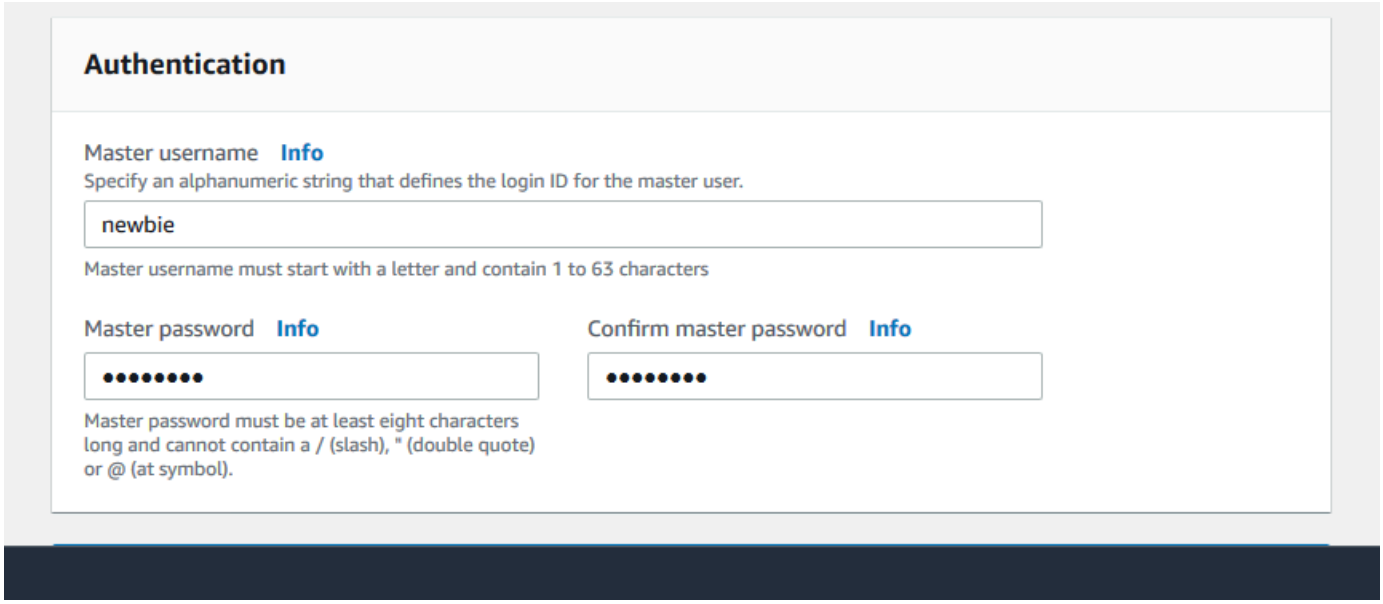


4. Amazon DocumentDB クラスター のフォームの [設定] セクションをそれぞれに応じて入力します。

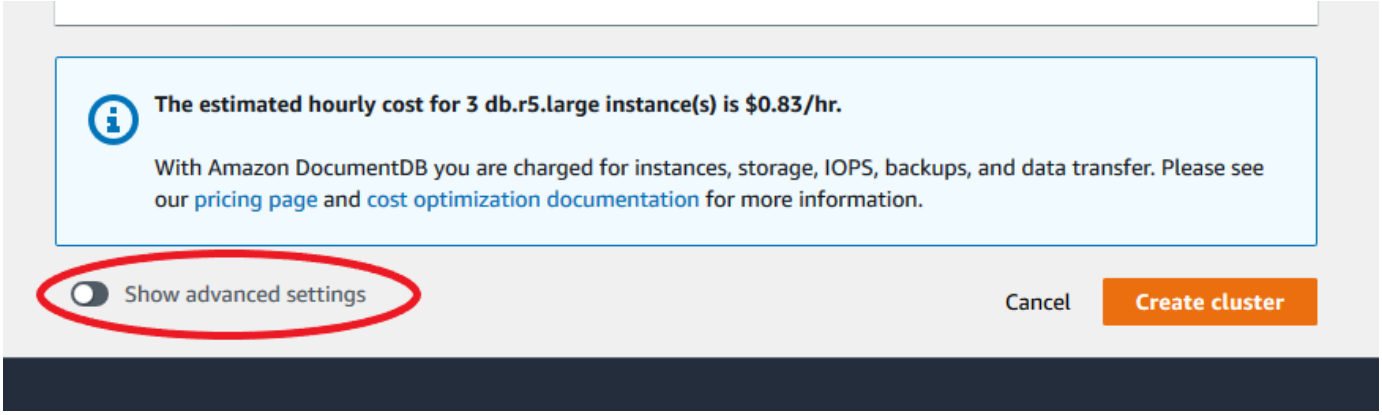
   - クラスター識別子: このインスタンスの一意の識別子を入力するか、クラスター識別子に基づき Amazon DocumentDB でインスタンスを指定できるようにします。
   - エンジンバージョン: [4.0.0] を選択します。
   - インスタンスクラス: [db.r5.large] を選択します。
   - インスタンス数: [3] を選択します。



5. [認証] セクションで、マスターユーザーネームとマスターパスワードを入力します。

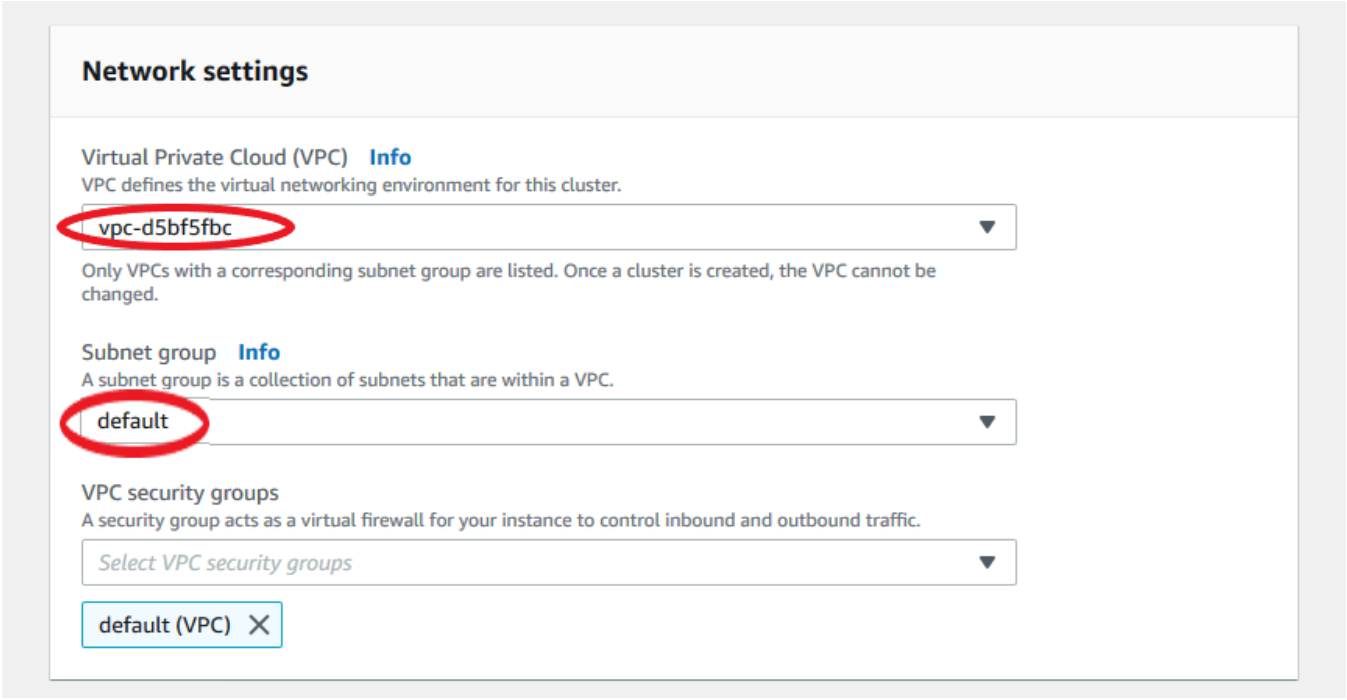


6. 詳細設定の表示 を選択します。

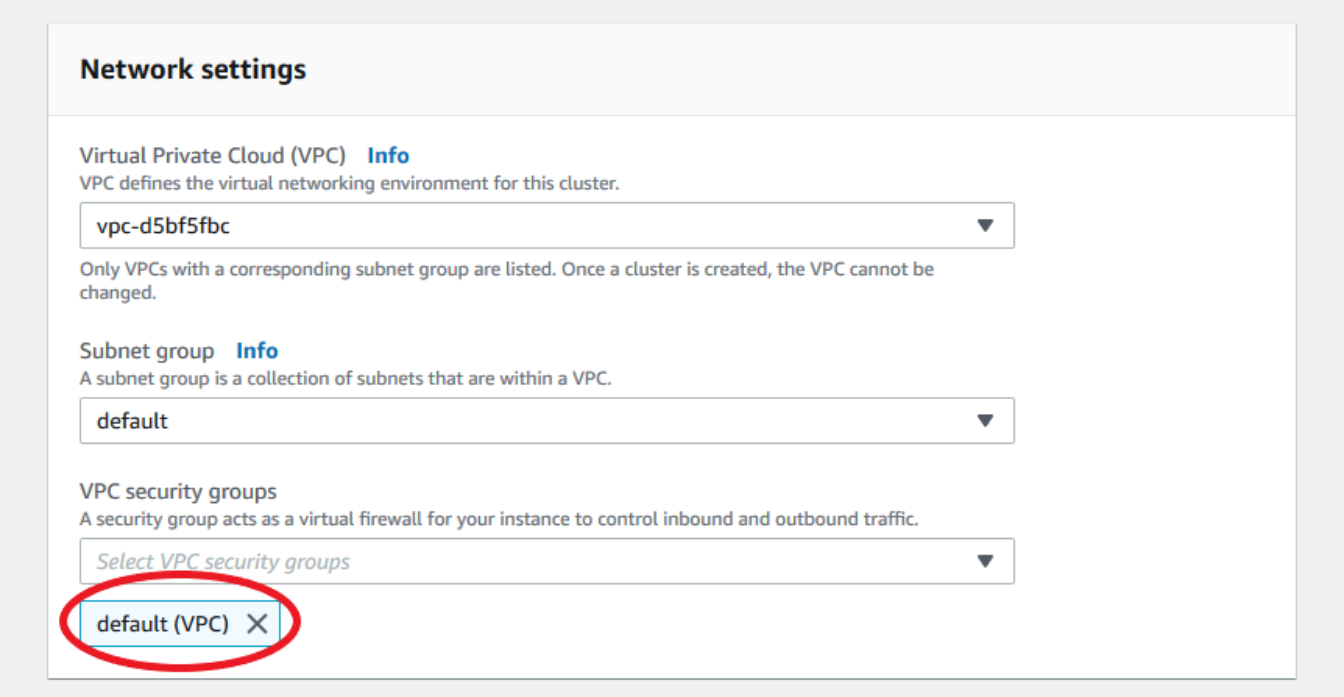


7. ネットワーク設定セクションで、次の操作を行います。

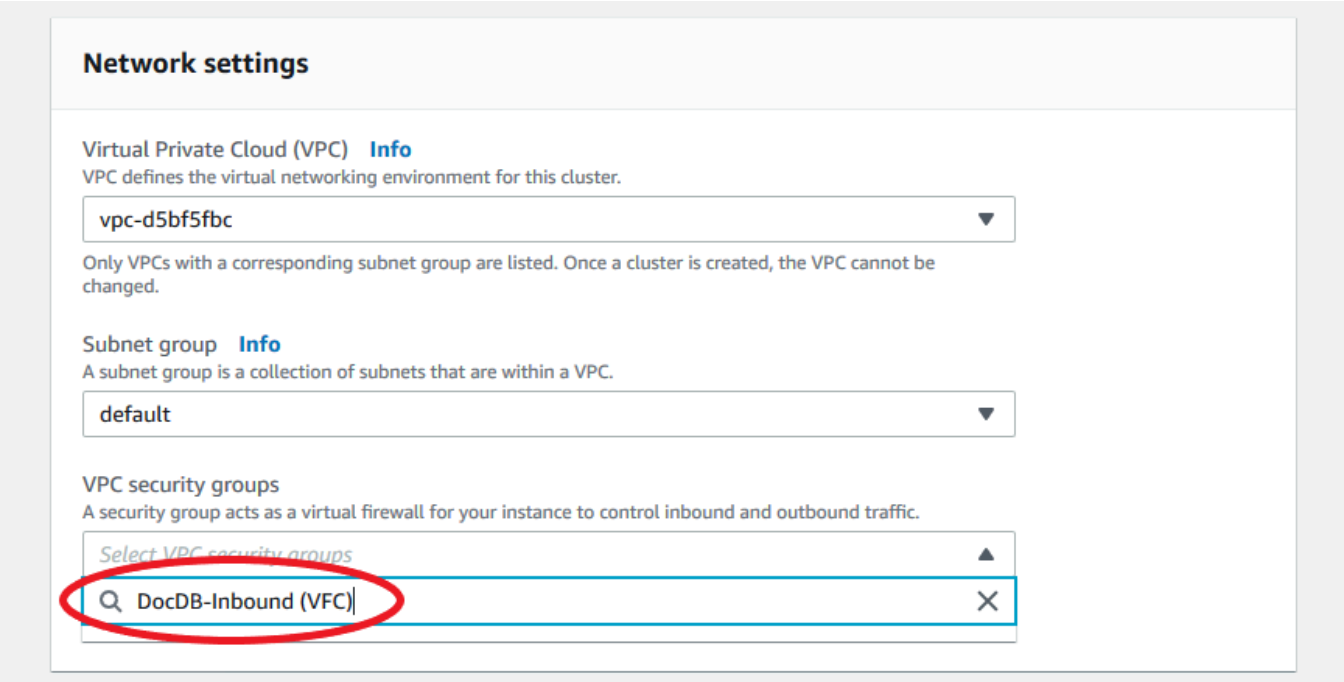
   - Virtual Private Cloud (VPC) とサブネットグループのデフォルトオプションを保持します。



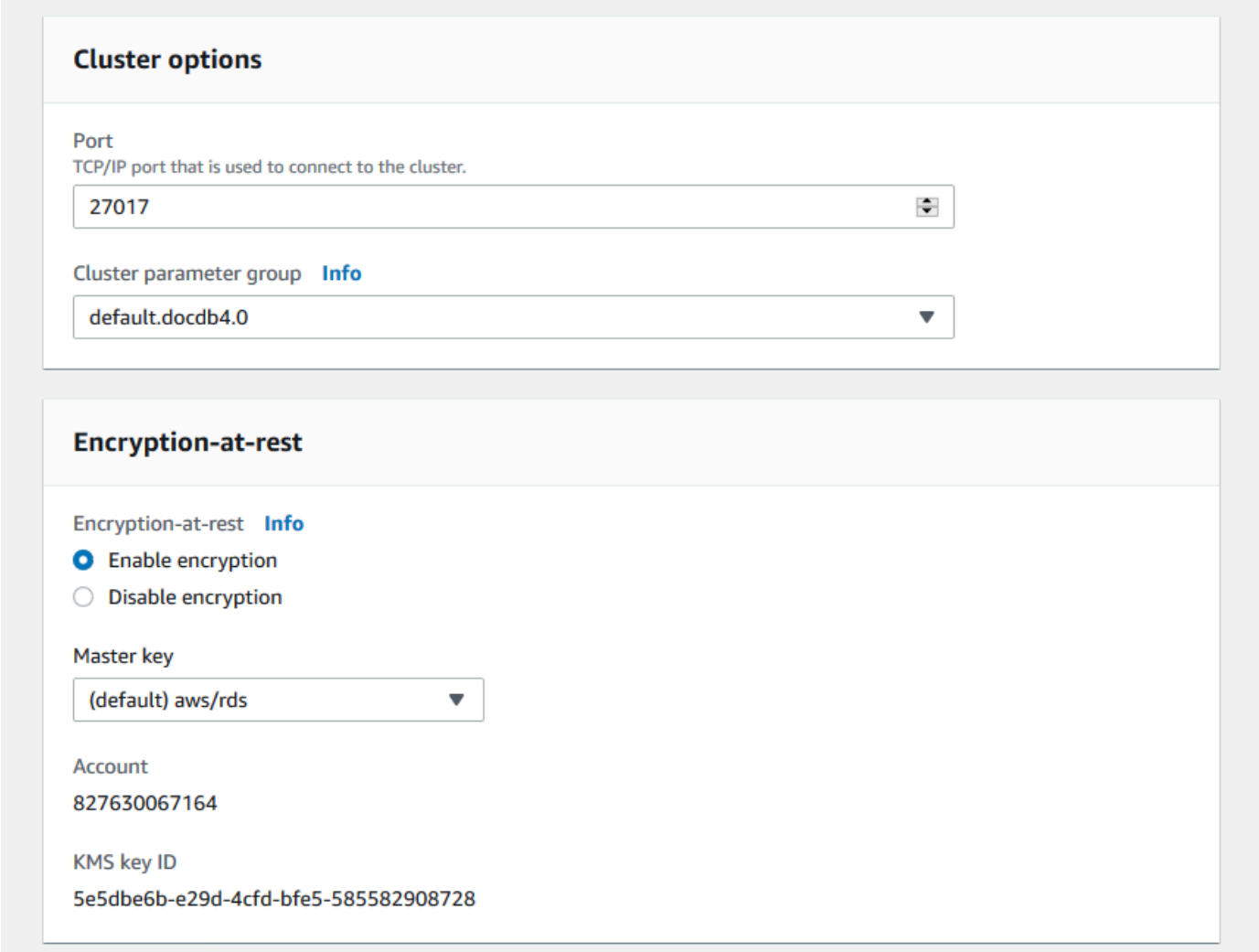
   - VPC セキュリティグループの場合、デフォルト (VPC) がすでに追加されている必要があります。



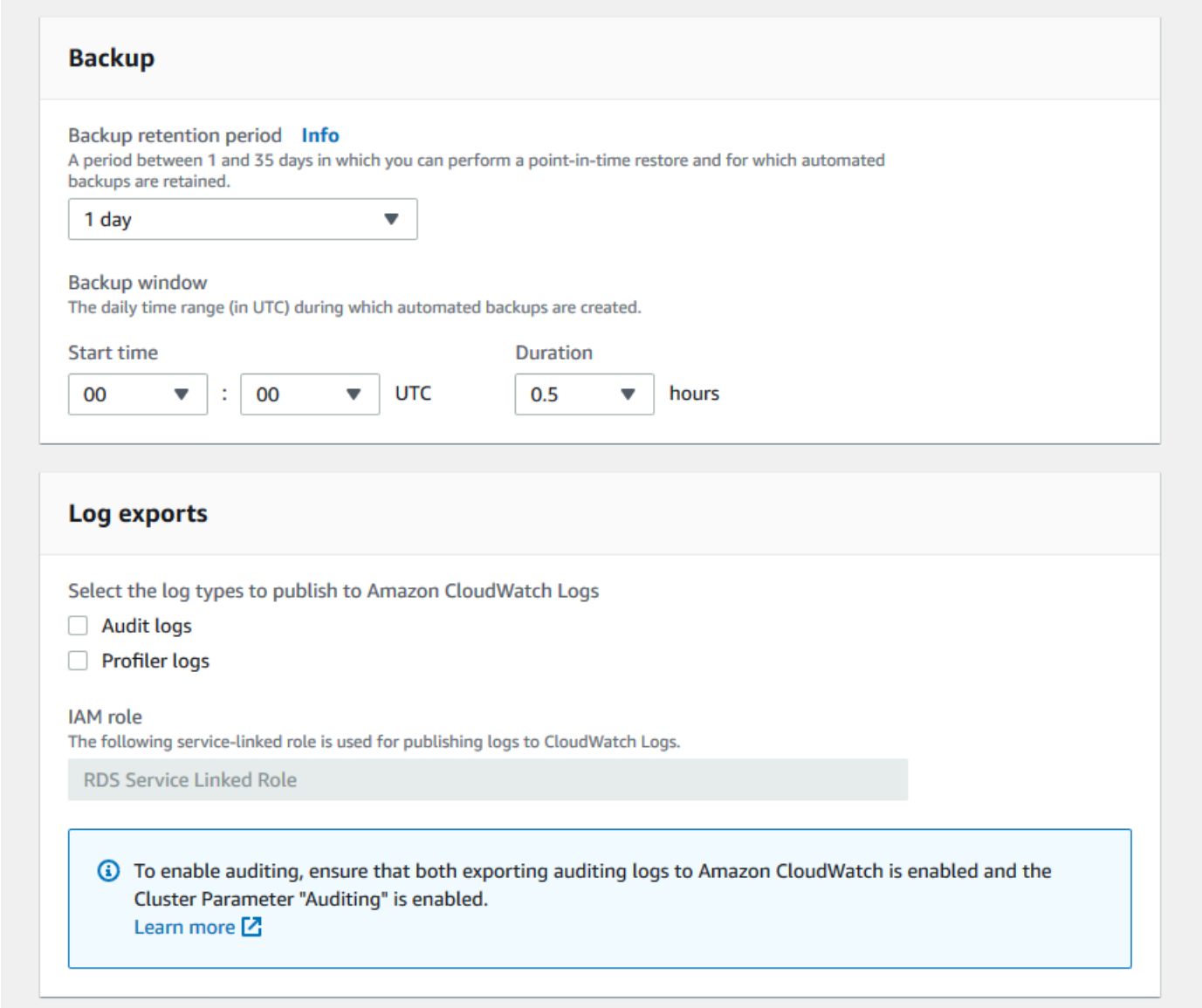
- VPC セキュリティグループフィールドに DocDBと入力し、DocDB-Inbound (VFC) を選択します。



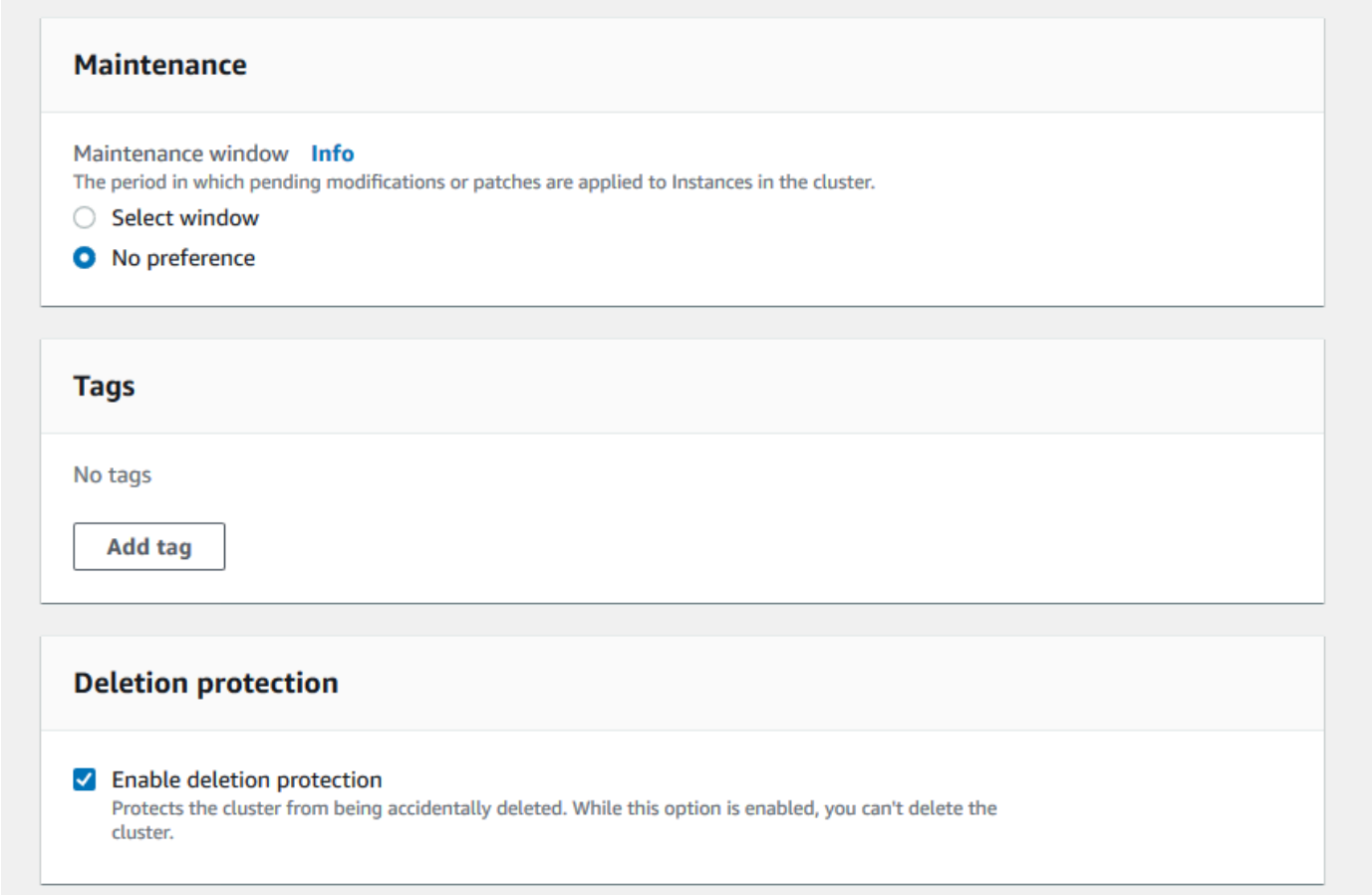
8. クラスターオプションとEncryption-at-restでは、デフォルトの選択のままにします。



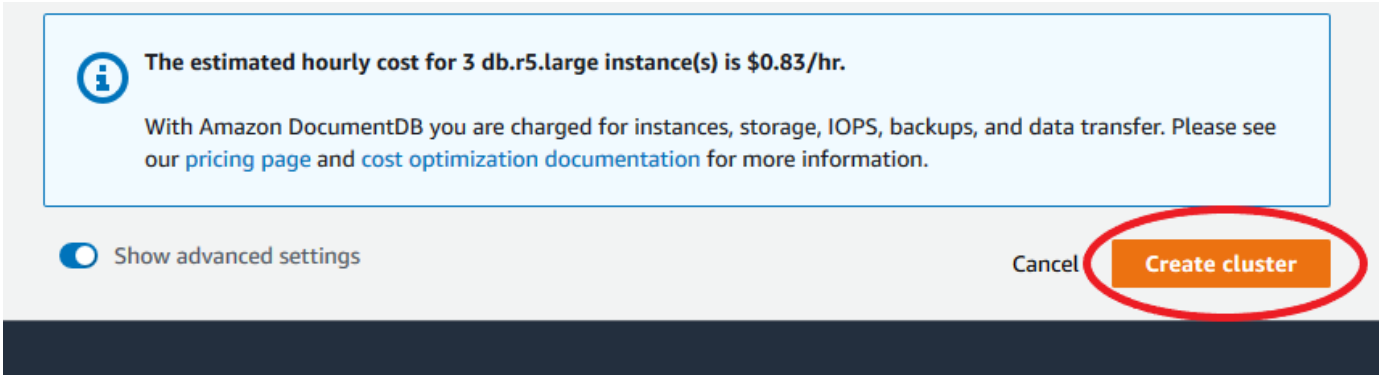
9. バックアップとログのエクスポートでは、デフォルトの選択のままにします。



10. メンテナンス、タグ、削除保護の場合は、デフォルトの選択のままにします。



11. 次に、「クラスターの作成」というボタンをクリックします。



の使用 AWS CLI

Amazon DocumentDB リージョンクラスターを作成するには、create-global-cluster AWS CLIを呼び出します。次の AWS CLI コマンドは、 という名前の Amazon DocumentDB クラスターを作成しますglobal-cluster-id。削除に対する保護についての詳細は、「Amazon DocumentDB クラスターの削除」を参照してください。

また、--engine-version はデフォルトで最新のメジャーエンジンバージョンとなるオプションのパラメータです。現在のメジャーエンジンバージョンは 5.0.0 です。新しいメジャーエンジンバージョンがリリースされると、 のデフォルトのエンジンバージョン--engine-versionが最新のメジャーエンジンバージョンを反映するように更新されます。その結果、本番ワークロード、特にスクリプト、オートメーション、または AWS CloudFormation テンプレートに依存するワークロードでは、目的のメジャーバージョン--engine-versionに を明示的に指定することをお勧めします。

db-subnet-group-name または が指定されvpc-security-group-idていない場合、Amazon DocumentDB は指定されたリージョンのデフォルトのサブネットグループと Amazon VPC セキュリティグループを使用します。

次の例では、各 *############* を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb create-db-cluster \
      --global-cluster-identifier global-cluster-id \
      --source-db-cluster-identifier arn:aws:rds:us-east-1:111122223333:cluster-id
```

Windows の場合:

```
aws docdb create-db-cluster ^
      --global-cluster-identifier global-cluster-id ^
      --source-db-cluster-identifier arn:aws:rds:us-east-1:111122223333:cluster-id
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "DBCluster": {
        "StorageEncrypted": false,
        "DBClusterMembers": [],
        "Engine": "docdb",
        "DeletionProtection" : "enabled",
        "ClusterCreateTime": "2018-11-26T17:15:19.885Z",
        "DBSubnetGroup": "default",
        "EngineVersion": "4.0.0",
        "MasterUsername": "masteruser",
        "BackupRetentionPeriod": 1,
```

```
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:cluster-id",
        "DBClusterIdentifier": "cluster-id",
        "MultiAZ": false,
        "DBClusterParameterGroup": "default.docdb4.0",
        "PreferredBackupWindow": "09:12-09:42",
        "DbClusterResourceId": "cluster-KQSGI4MHU4NTDDRVNLNTU7XVAY",
        "PreferredMaintenanceWindow": "tue:04:17-tue:04:47",
        "Port": 27017,
        "Status": "creating",
        "ReaderEndpoint": "cluster-id.cluster-ro-sfcrlcjcoroz.us-
east-1.docdb.amazonaws.com",
        "AssociatedRoles": [],
        "HostedZoneId": "ZNKXTT8WH85VW",
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-77186e0d",
                "Status": "active"
            }
        ],
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1c",
            "us-east-1e"
        ],
        "Endpoint": "cluster-id.cluster-sfcrlcjcoroz.us-east-1.docdb.amazonaws.com"
    }
}
```

クラスターが作成されるまでに数分かかります。 AWS Management Console または を使用して AWS CLI 、クラスターのステータスをモニタリングできます。詳細については、「Amazon DocumentDB クラスターのステータスのモニタリング」を参照してください。

> **Important**
>
> を使用して Amazon DocumentDB リージョンクラスター AWS CLI を作成する場合、インスタンスは作成されません。したがって、プライマリインスタンスと必要なレプリカインスタンスを明示的に作成してください。コンソールまたは を使用してインスタンス AWS CLI を作成できます。詳細については、「クラスターへの Amazon DocumentDB インスタンスの追加 および CreateDBCluster」がある「Amazon DocumentDB API リファレンス」を参照してください。

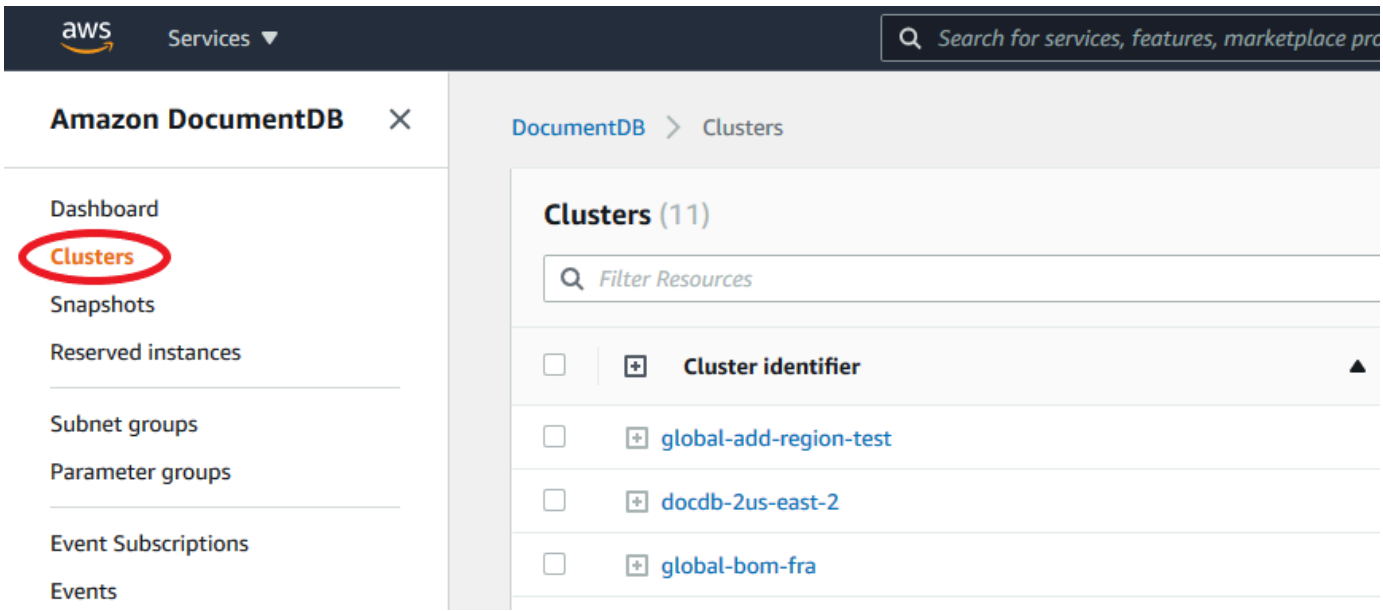
リージョンクラスターが使用可能になったら、次の手順に従って別のリージョンにセカンダリクラスターを追加できます: Amazon DocumentDB グローバルクラスター AWS リージョン への の追加。リージョンを追加すると、リージョンクラスターがプライマリクラスターになり、選択したリージョンに新しいセカンダリクラスターがあります。

Amazon DocumentDB グローバルクラスター AWS リージョン への の追加

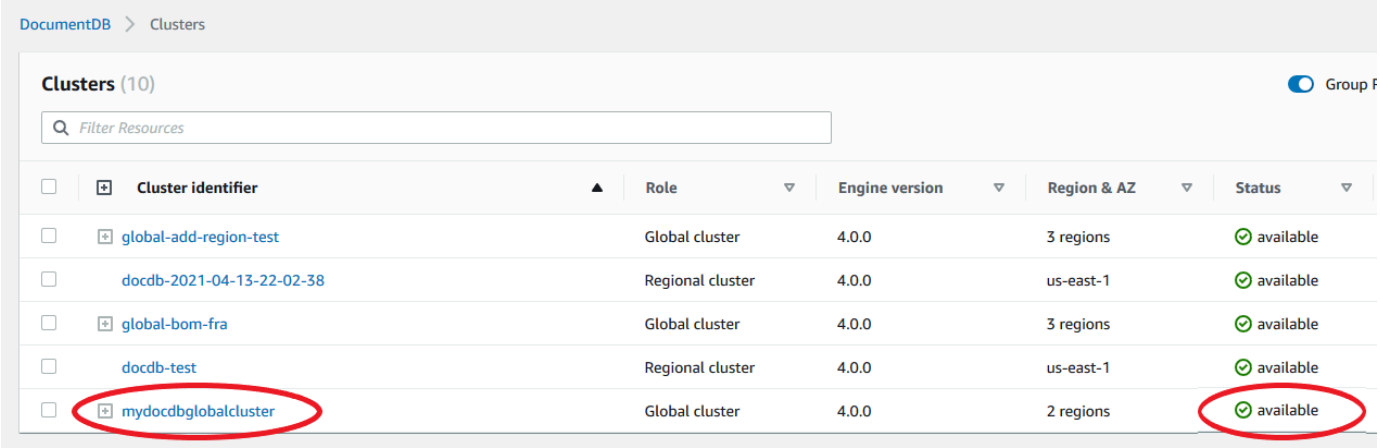
グローバルクラスターには、プライマリクラスターとは異なるリージョンに少なくとも 1 つのセカンダリクラスターが必要であり、最大 5 つのセカンダリクラスターを追加できます。追加する各セカンダリクラスターで、プライマリクラスターに許容されたレプリカの数を減らします。例えば、グローバルクラスターに 5 つのセカンダリリージョンがある場合、プライマリクラスターには 10 (15 ではなく) 個のレプリカのみを含めることができます。詳細については、「Amazon DocumentDB グローバルクラスターの設定要件」を参照してください。

の使用 AWS Management Console

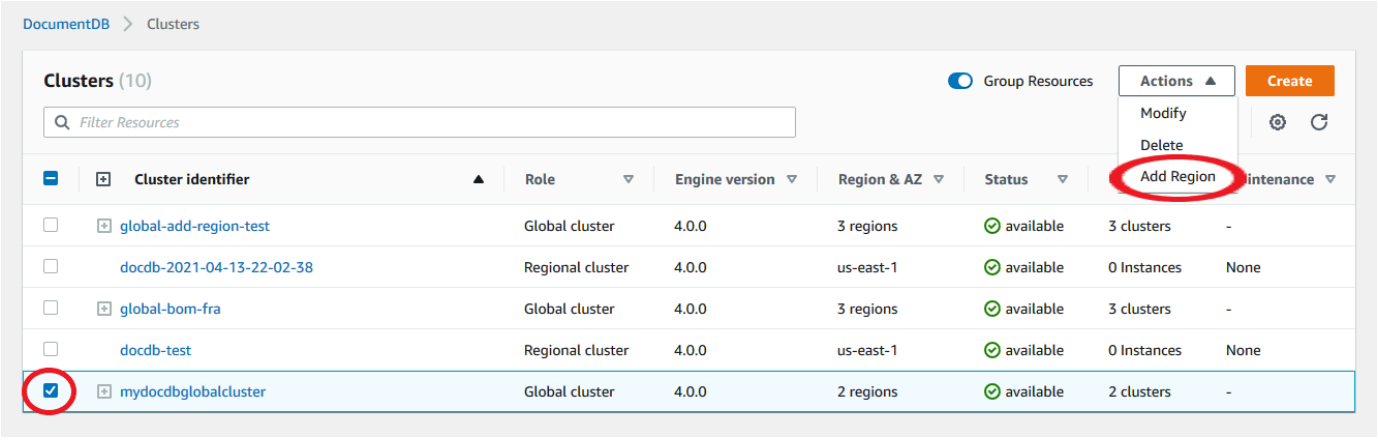
1. にサインイン AWS Management Console し、Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択します。



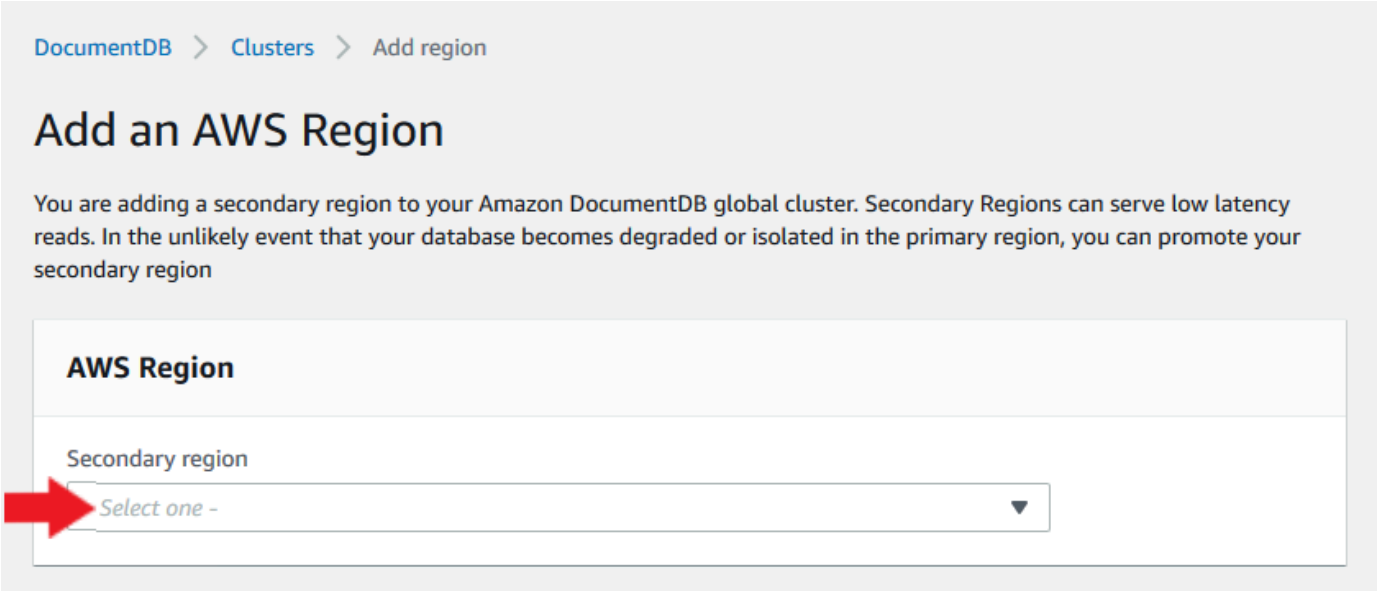
3. セカンダリクラスターを追加するクラスターを選択します。クラスターが `Available` であることを確認します。



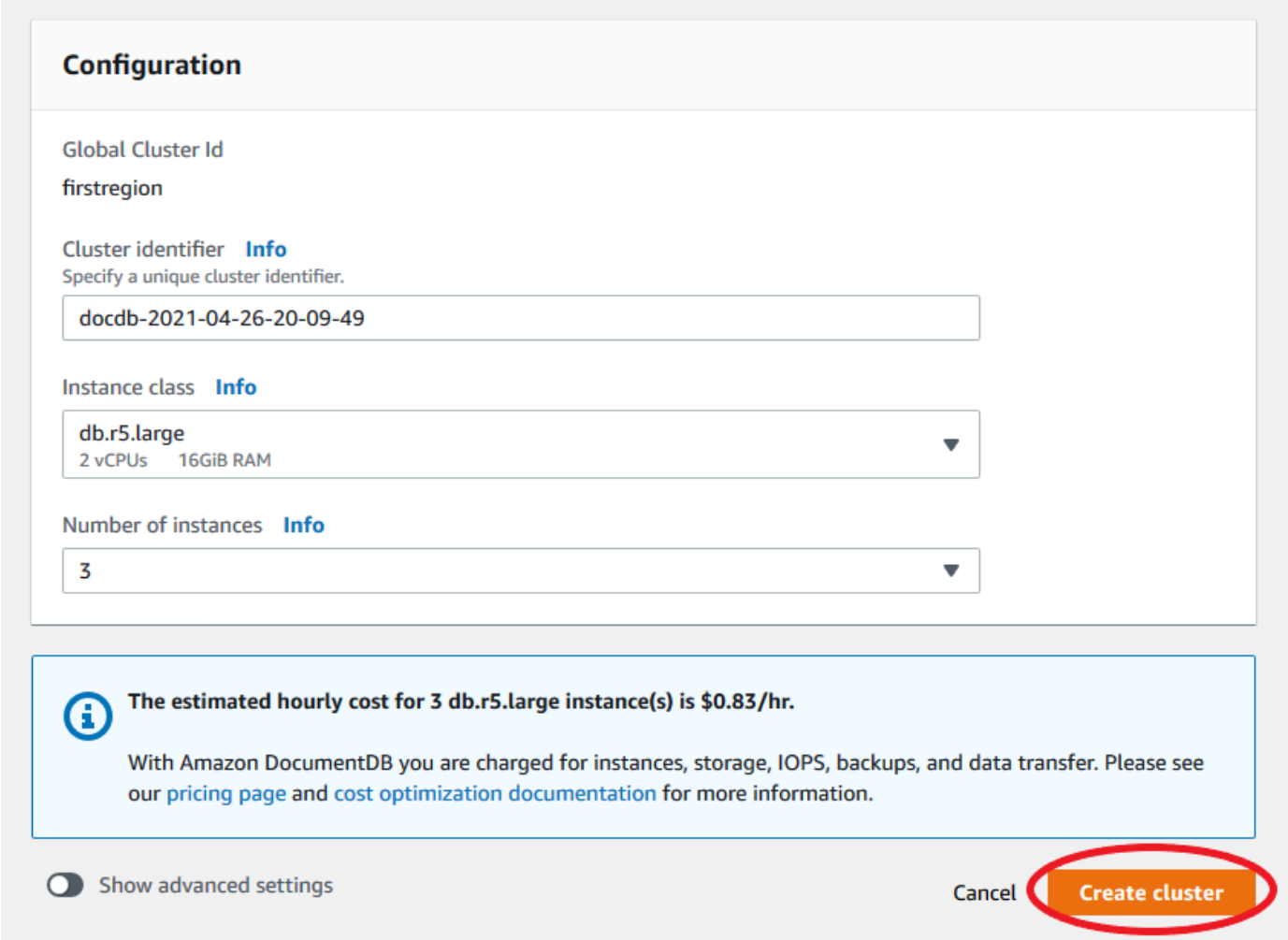
4. アクションのドロップダウンリストを選択し、リージョンの追加を選択します。



5. 「 AWS リージョンの追加」ページで、セカンダリリージョンを選択します。同じグローバルクラスターのセカンダリクラスターが既にあるリージョンを選択することはできません。また、プライマリクラスターと同じリージョンにすることはできません。追加する最初のリージョンである場合は、選択したグローバルクラスター識別子も指定する必要があります。



6. 新しいリージョンのセカンダリクラスターの残りのフィールドに入力し、クラスターの作成を選択します。リージョンの追加が完了したら、 のクラスターのリストに表示されます AWS Management Console。



の使用 AWS CLI

- お客様のグローバルクラスターの名前 `create-db-cluster` を指定して、(`--global-cluster-identifier`) CLI コマンドを使用します。他のパラメータでは、以下の操作を行います。
  - では`--region`、プライマリリージョン AWS リージョン とは異なる を選択します。
  - `--engine` および `--engine-version` パラメータの特定の値を選択します。
  - 暗号化されたクラスターの場合、暗号化`--source-region`用の AWS リージョン としてプライマリを指定します。

次の例では、新しい Amazon DocumentDB クラスターを作成し、それを読み取り専用セカンダリクラスターとしてグローバルクラスターにアタッチします。最後のステップでは、インスタンスを新しいクラスターに追加します。

次の例では、各 ############## を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb --region secondary-region-id \
  create-db-cluster \
    --db-cluster-identifier cluster-id \
    --global-cluster-identifier global-cluster-id \
    --engine-version version \
    --engine docdb

aws docdb --region secondary-region-id \
  create-db-instance \
    --db-cluster-identifier cluster-id \
    --global-cluster-identifier global-cluster-id \
    --engine-version version \
    --engine docdb
```

Windows の場合:

```
aws docdb --region secondary-region-id ^
  create-db-cluster ^
    --db-cluster-identifier cluster-id ^
    --global-cluster-identifier global-cluster-id ^
    --engine-version version ^
    --engine docdb

aws docdb --region secondary-region-id ^
  create-db-instance ^
    --db-cluster-identifier cluster-id ^
    --global-cluster-identifier global-cluster-id ^
    --engine-version version ^
    --engine docdb
```

Amazon DocumentDB グローバルクラスターのスナップショットの使用

Amazon DocumentDB クラスターのスナップショットを復元することで、グローバルクラスターの開始ポイントとして使用することができます。これを行うには、スナップショットを復元し、新しいクラスターを作成する必要があります。これは、グローバルクラスタのプライマリクラスターとして

機能します。その後、復元されたクラスターに別のリージョンを追加して、グローバルクラスターに変換できます。

Amazon DocumentDB グローバルクラスターの管理

グローバルクラスターを構成する個別のクラスターに対して、ほとんどの管理オペレーションを実行できます。コンソールで、[Group related resources (グループ関連のリソース)] を[クラスター] ページで選択すると、プライマリクラスターとセカンダリクラスターが、関連付けられたグローバルクラスターの下にグループとして表示されます。

グローバルクラスターの設定タブには、クラスターが実行されている AWS リージョン 、バージョン、グローバルクラスター識別子が表示されます。

トピック

- Amazon DocumentDB グローバルクラスターの変更
- Amazon DocumentDB グローバルクラスターのパラメータの変更
- Amazon DocumentDB グローバルクラスターからのクラスターの削除
- Amazon DocumentDB グローバルクラスターからのクラスターの削除
- セカンダリリージョンでのヘッドレス Amazon DocumentDB クラスターの作成

Amazon DocumentDB グローバルクラスターの変更

のクラスターページには、すべてのグローバルクラスターが AWS Management Console 一覧表示され、それぞれのプライマリクラスターとセカンダリクラスターが表示されます。グローバルクラスターには、独自の構成設定があります。具体的には、プライマリクラスターとセカンダリクラスターに関連付けられたリージョンがあります。

グローバルクラスターに変更を加えると、変更をキャンセルする機会が与えられます。

[続行] を選択したら、変更を確認します。

Amazon DocumentDB グローバルクラスターのパラメータの変更

グローバルクラスター内の各クラスターで、クラスターパラメータグループを個別に設定することができます。ほとんどのパラメータの動作は、他の種類の Amazon DocumentDB クラスターと同じです。グローバルデータベース内のすべてのクラスター間で設定の一貫性を保つことをお勧めします。これにより、セカンダリクラスターをプライマリクラスターに昇格した場合に、予期しない動作の変更が生じることを回避できます。

例えば、別のクラスターがプライマリクラスターを肩代わりしたときに動作が変わらないように、タイムゾーンと文字セットに同じ設定を使用します。

Amazon DocumentDB グローバルクラスターからのクラスターの削除

グローバルクラスターからクラスターを削除したい場合がいくつかあります。例えば、プライマリクラスターの性能低下や切断が生じて、グローバルクラスターからクラスターを削除したい場合があります。その後、新しいグローバルクラスターを作成するために使用できるスタンドアロンのプロビジョニングクラスターになります。詳細については[Amazon DocumentDB グローバルクラスターの手動フェイルオーバーの実行]を参照してください。

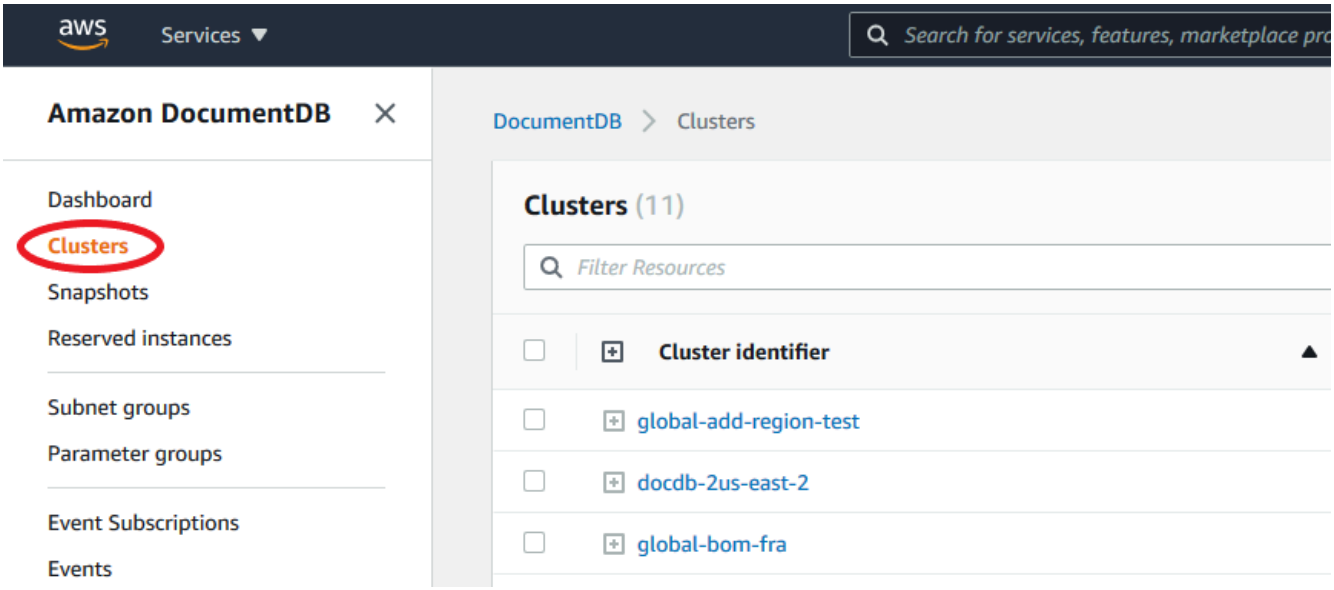
また、不要になったグローバルクラスターを削除するために、クラスターを削除する場合も可能です。関連するすべてのクラスターを削除 (デタッチ) した後、プライマリが最後に残るまで、グローバルクラスターを削除することはできません。詳細については、「[Amazon DocumentDB グローバルクラスターからのクラスターの削除]」を参照してください。

> **Note**
>
> クラスターがグローバルクラスターからデタッチされると、プライマリは同期されなくなります。これは、完全な読み取り/書き込み機能を備えたスタンドアロンのプロビジョニングクラスターになります。さらに、Amazon DocumentDB コンソールに表示されなくなりました。これは、クラスターが配置されたコンソールでリージョンを選択した場合にのみ表示されます。

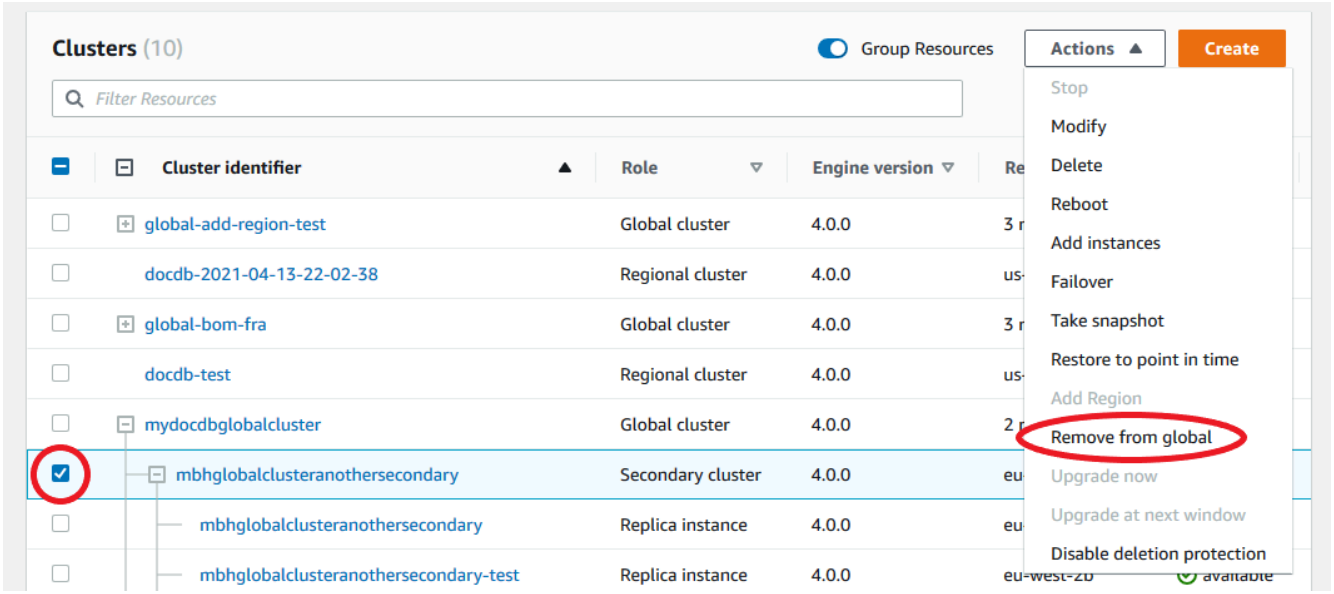
グローバルクラスターからクラスターを削除するには AWS Management Console、、 AWS CLI、または RDS API を使用します。

Using the AWS Management Console

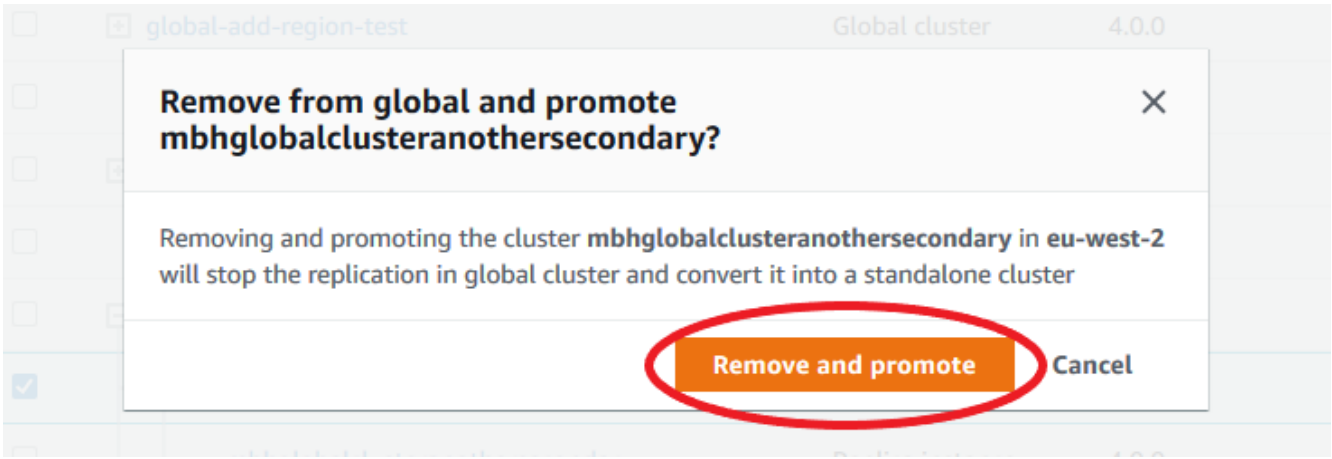
1. にサインイン AWS Management Console し、Amazon DocumentDB コンソールに移動します。
2. 左のナビゲーションで [Clusters (クラスター)] を選択します。



3. グローバルクラスターを展開して、すべてのセカンダリクラスターを表示します。削除するセカンダリクラスターを選択します。[アクション] を選択し、ドロップダウンメニューで [グローバルから削除] を選択します。



4. グローバルクラスターからセカンダリをデタッチするかどうかを確認するプロンプトが表示されます。グローバルクラスターからクラスターを削除するには、[削除して昇格する] を選択します。



クラスターは、セカンダリとして機能しなくなり、プライマリクラスターと同期しなくなります。これは、完全な読み取り/書き込み機能を備えたスタンドアロンクラスターです。

すべてのセカンダリクラスターを解除または削除したら、次にプライマリクラスターを同じ方法で解除できます。すべてのセカンダリクラスターを削除するまで、グローバルクラスターからプライマリクラスターをデタッチ (削除) することはできません。グローバルクラスターは、リージョンと AZ が 0 のままクラスターリストに残り続ける場合があります。今後このグローバルクラスターを使用しない場合は、削除できます。

Using the AWS CLI

グローバルクラスターからクラスターを削除するには、次のパラメータを指定して、`remove-from-global-cluster` CLI コマンドを実行します。

- `--global-cluster-identifier` - グローバルクラスターの名前 (識別子)。
- `--db-cluster-identifier` - グローバルクラスターから削除する各クラスターの名前。

以下の例では、まずセカンダリクラスターを削除し、次にグローバルクラスターからプライマリクラスターを削除します。

Linux、macOS、Unix の場合:

```
aws docdb --region secondary_region \
  remove-from-global-cluster \
    --db-cluster-identifier secondary_cluster_ARN \
    --global-cluster-identifier global_cluster_id

aws docdb --region primary_region \
```

```
  remove-from-global-cluster \
    --db-cluster-identifier primary_cluster_ARN \
    --global-cluster-identifier global_cluster_id
```

グローバルクラスターの各セカンダリリージョンで `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` コマンドを繰り返し実行します。

Windows の場合:

```
aws docdb --region secondary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier secondary_cluster_ARN ^
    --global-cluster-identifier global_cluster_id

aws docdb --region primary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier primary_cluster_ARN ^
    --global-cluster-identifier global_cluster_id
```

グローバルクラスターの各セカンダリリージョンで `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` コマンドを繰り返し実行します。

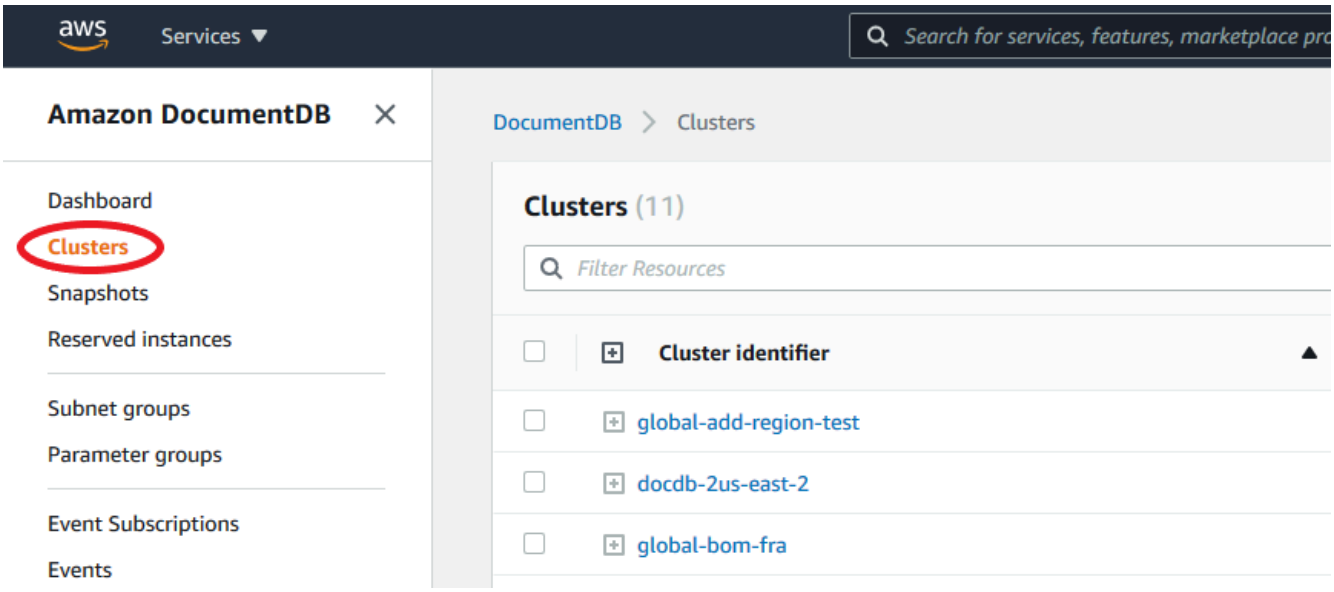
Amazon DocumentDB グローバルクラスターからのクラスターの削除

グローバルクラスターを削除するには、次の手順を実行します。

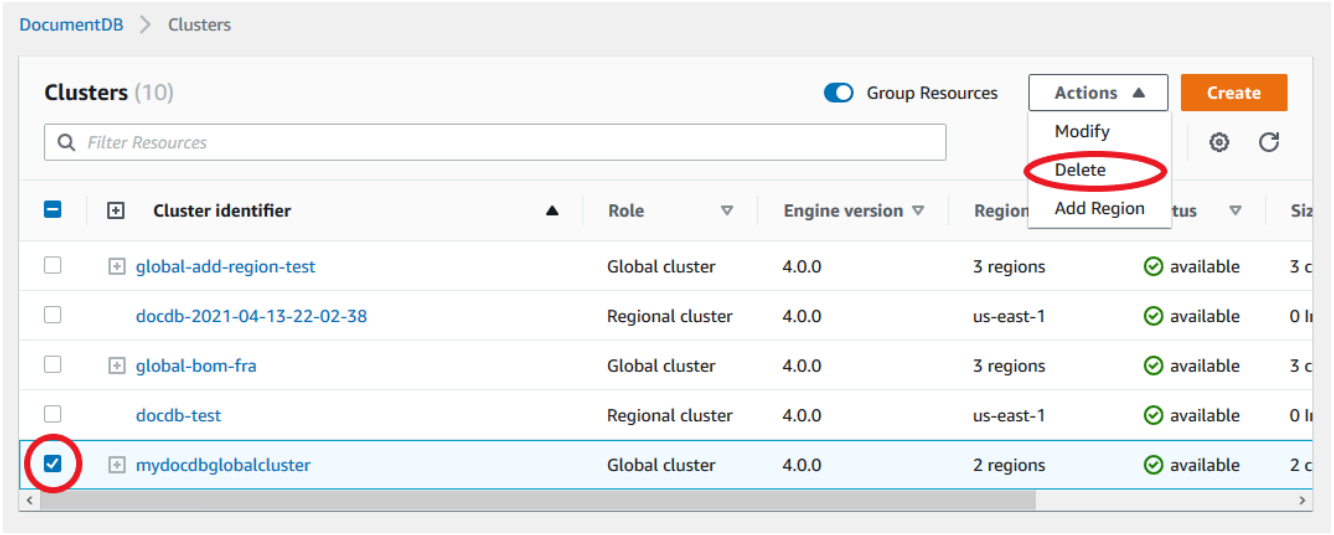
- グローバルクラスターからすべてのセカンダリクラスターを削除します。各クラスターは、スタンドアロンクラスターになります。前のセクション「」を参照してくださいAmazon DocumentDB グローバルクラスターからのクラスターの削除。
- 各スタンドアロンクラスターから、すべてのレプリカを削除します。
- グローバルクラスターからプライマリクラスターを削除します。これは、スタンドアロンクラスターになります。
- プライマリクラスターから、まずすべてのレプリカを削除し、次にプライマリインスタンスを削除します。新しいスタンドアロンクラスターからプライマリインスタンスを削除すると、通常クラスターとグローバルクラスターも削除されます。

Using the AWS Management Console

1. にサインイン AWS Management Console し、Amazon DocumentDB コンソールに移動します。
2. [クラスター] を選択し、削除するグローバルクラスターを見つけます。



3. グローバルクラスターを選択した状態で、[削除] を [アクション] メニューから選択します。



グローバルクラスターからすべてのクラスターが削除されていることを確認します。グローバルクラスターでは、リージョンと AZ が 0で、クラスターのサイズが 0 と表示されるはずです。グローバルクラスターにクラスターが含まれている場合は、これを削除できません。まず、前のステップ「」の手順に従う必要がありますAmazon DocumentDB グローバルクラスターからのクラスターの削除。

Using the AWS CLI

グローバルクラスターを削除するには、次の例に示すように、 の名前 AWS リージョン とグローバルクラスター識別子を使用して `delete-global-cluster` CLI コマンドを実行します。

Linux、macOS、Unix の場合:

```
aws docdb --region primary_region delete-global-cluster \
   --global-cluster-identifier global_cluster_id
```

Windows の場合:

```
aws docdb --region primary_region delete-global-cluster ^
   --global-cluster-identifier global_cluster_id
```

セカンダリリージョンでのヘッドレス Amazon DocumentDB クラスターの作成

Amazon DocumentDB グローバルクラスターには、プライマリ AWS リージョン とは異なる に少なくとも 1 つのセカンダリクラスターが必要ですが、セカンダリクラスターにはヘッドレス設定を使用できます。ヘッドレスセカンダリ Amazon DocumentDB クラスターは、インスタンスがないクラスターのことです。この種類の設定では、グローバルクラスターの費用を削減できます。Amazon DocumentDB クラスターでは、コンピューティングとストレージは分離されています。インスタンスがない場合、コンピューティングに対する課金は発生せず、ストレージに対してのみが課金されます。正しく設定されていれば、ヘッドレスセカンダリのストレージボリュームはプライマリクラスターと同期したままになります。

Amazon DocumentDB グローバルクラスターの作成時に通常行うように、セカンダリクラスターを追加します。ただし、プライマリクラスターがセカンダリへのレプリケーションを開始した後、セカンダリクラスターから読み取り専用インスタンスを削除します。インスタンスがなくなったため、このセカンダリクラスターは「ヘッドレス」とみなされます。ただし、ストレージボリュームはプライマリ Amazon DocumentDB クラスターと同期されたままとなります。

> **Important**
>
> リージョン全体の障害を 15 分以上許容できるお客様にのみ、ヘッドレスクラスターをお勧めします。これは、ヘッドレスセカンダリクラスターを使用してリージョン全体の障害から復旧するには、フェイルオーバー後にユーザーが新しいインスタンスを作成する必要があるためです。新しいインスタンスが利用可能になるまで 10 ~ 15 分かかることがあります。

ヘッドレスセカンダリクラスターをお客様のグローバルクラスターに追加するには

1. にサインイン AWS Management Console し、Amazon DocumentDB コンソールを開きます。
2. 左のナビゲーションで [Clusters (クラスター)] を選択します。
3. セカンダリクラスターを必要とするグローバルクラスターを選択します。プライマリクラスターが `Available` であることを確認します。
4. [アクション] で、[リージョンの追加] を選択します。
5. [リージョンの追加] ページで、セカンダリリージョンを選択します。

   > **Note**
   >
   > 同じグローバルクラスターのセカンダリクラスターがすでに存在するリージョンを選択することはできません。また、プライマリクラスターと同じリージョンにすることはできません。

6. 新しいリージョンのセカンダリクラスターの残りのフィールドに入力してください。これらは、他のクラスターインスタンスと同じ設定オプションです。
7. リージョンを追加します。グローバルクラスターへのリージョンの追加が完了すると、`Clusters` の AWS Management Consoleのリストに表示されます。
8. AWS Management Console または を使用して、続行する前にセカンダリクラスターとそのリーダーインスタンスのステータスを確認します AWS CLI。以下に、 AWS CLIを使用した場合のコマンドの例を示します。

   ```
   $ aws docdb describe-db-clusters --db-cluster-identifier secondary-cluster-id --
   query '*[].[Status]' --output text
   ```

   新しく追加されたセカンダリクラスターのステータスが作成中から使用可能に変わるまでに数分かかる場合があります。クラスターが使用可能になったら、リーダーインスタンスを削除できます。
9. セカンダリクラスターでリーダーインスタンスを選択し、[Delete (削除)] を選択します。
10. リーダーインスタンスを削除しても、セカンダリクラスターはグローバルクラスターの一部にとどまります。関連付けられたインスタンスは存在しません。

> **Note**
>
> このヘッドレスセカンダリの Amazon DocumentDB クラスターを使用して、Amazon DocumentDB グローバルクラスターをプライマリリージョンの予期しない停止から手動で復旧できます。

Amazon DocumentDB グローバルクラスターに接続する

グローバルクラスターへの接続方法は、クラスターへの書き込みと、クラスターからの読み取りの、どちらを必要とするのかに応じて異なります。

- 読み取り専用のリクエストまたはクエリ場合、 AWS リージョンにあるクラスターのリーダーエンドポイントに接続します。
- データ操作言語 (DML) またはデータ定義言語 (DDL) のステートメントを実行するには、プライマリクラスターのクラスターエンドポイントに接続します。このエンドポイントは、アプリケーション AWS リージョン とは異なる にある可能性があります。

コンソールでグローバルクラスターを表示すると、そのすべてのクラスターに関連付けられているすべての汎用エンドポイントを表示できます。

グローバルクラスターへの接続方法は、データベースへの書き込みと、データベースからの読み取りの、どちらを必要とするのかに応じて異なります。プライマリリージョンから提供する DDL、DML、および読み取りオペレーションの場合は、プライマリクラスターに接続する必要があります。レプリカセットモードのクラスターエンドポイントを使用し、プライマリクラスターに接続し、読み取り設定を `secondaryPreferred=true` にすることをお勧めします。これにより、プライマリクラスターのライターインスタンスに書き込みトラフィックがルーティングされ、読み取りトラフィックがプライマリクラスターのレプリカインスタンスにルーティングされます。

クロスリージョン、読み取り専用トラフィックの場合は、いずれかのセカンダリクラスターに接続する必要があります。レプリカセットモードでクラスターエンドポイントを使用してセカンダリクラスターに接続することをお勧めします。すべてのインスタンスは読み取り専用レプリカインスタンスであるため、読み取り設定を指定する必要はありません。レイテンシーを最小限にするときは、お使いのリージョンか、最寄りのリージョンの、いずれかにあるリーダーエンドポイントを選択します。

Amazon DocumentDB グローバルクラスターのモニタリング

Amazon DocumentDB (MongoDB 互換性) は CloudWatch と統合することで、クラスターのオペレーションメトリクスを収集および分析できます。これらのメトリクスは、CloudWatch コンソール、Amazon DocumentDB コンソール、 AWS Command Line Interface （ AWS CLI ）、または CloudWatch API を使用してモニタリングできます。

グローバルクラスターをモニタリングするには、次の CloudWatch メトリクスを使用します。

メトリクス	説明
`GlobalClusterReplicatedWriteIO`	プライマリ のクラスターボリュームから セカンダリ のクラスターボリューム AWS リージョン にレプリケートされた課金書き込み I/O オペレーションの平均数。5 分間隔で AWS リージョン報告されます。各セカンダリリージョンへレプリケートされた `ReplicatedWriteIOs` の数は、リージョン内の `VolumeWriteIOPs` の数と同じで、プライマリリージョンによって実行されます。
`GlobalClusterDataTransferBytes`	プライマリクラスターからセカンダリクラスター AWS リージョン に転送されたデータの量 AWS リージョン。バイト単位で測定します。
`GlobalClusterReplicationLag`	変更イベントをプライマリクラスターの から セカンダリクラスターの AWS リージョン に レプリケートする際の、ミリ秒単位の遅延の量 AWS リージョン

これらのメトリクスの表示方法の詳細については、「CloudWatch データを表示する」を参照してください。

ディザスタリカバリと Amazon DocumentDB グローバルクラスターについて

トピック

- Amazon DocumentDB グローバルクラスターのマネージドフェイルオーバーの実行
- Amazon DocumentDB グローバルクラスターの手動フェイルオーバーの実行
- Amazon DocumentDB グローバルクラスターのスイッチオーバーの実行
- グローバルクラスターのスイッチオーバーまたはフェイルオーバーのブロック解除

グローバルクラスターを使用すると、リージョンの障害などの災害から迅速に復旧できます。災害からの復旧は、通常、RTO と RPO (目標復旧時点) の値を使用して測定します。

- 目標復旧時間 (RTO)– 災害後にシステムが稼働状態に戻るまでにかかる時間。つまり、RTO はダウンタイムを測定します。グローバルクラスターの場合、RTO は分単位です。
- 目標復旧時点 (RPO) – 損失する可能性があるデータの量 (時間単位)。グローバルクラスターの場合、目標復旧時点 (RPO) は通常、秒単位で測定されます。
- 予期しない停止から復旧するには、グローバルクラスター内のセカンダリの 1 つのクロスリージョンフェイルオーバーを実行できます。グローバルクラスターに複数のセカンダリリージョンがある場合、プライマリとして昇格させたいすべてのセカンダリリージョンをデタッチしてください。次に、これらのセカンダリリージョンの 1 つを新しいプライマリ AWS リージョンに昇格させます。最後に、他の各セカンダリリージョンに新しいクラスターを作成し、それらのクラスターをグローバルクラスターにアタッチします。

Amazon DocumentDB グローバルクラスターのマネージドフェイルオーバーの実行

このアプローチは、真の地域災害やサービスレベルの全面的な停止が発生した場合でも事業を継続できるようにするためのものです。

マネージドフェイルオーバー中、プライマリクラスターは選択したセカンダリリージョンにフェイルオーバーされ、お使いの Amazon DocumentDB グローバルクラスターの既存のレプリケーショントポロジが維持されます。選択したセカンダリクラスターは、読み取り専用ノードの 1 つを完全な読み取り状態に昇格します。このステップにより、クラスターがプライマリクラスターのロールを引き受けることができます。クラスターが新しいロールを引き受ける間、データベースは短時間使用できなくなります。古いプライマリクラスターから選択したセカンダリクラスターにレプリケートされなかったデータは、このセカンダリクラスターが新しいプライマリクラスターになると失われます。古いプライマリボリュームは、新しいプライマリと同期する前にスナップショットを取得するべく、レプリケートされていないデータがスナップショットに保持されるように最善を尽くします。

> **Note**
>
> プライマリクラスターとすべてのセカンダリクラスターのエンジンバージョンが同じである場合にのみ、Amazon DocumentDB グローバルクラスターでマネージドクロスリージョンクラスターフェイルオーバーを実行できます。ご使用のエンジンバージョンに互換性がない場合は、Amazon DocumentDB グローバルクラスターの手動フェイルオーバーの実行 の手順に従ってフェイルオーバーを手動で実行できます。
> リージョンのエンジンバージョンが一致しない場合、フェイルオーバーはブロックされます。保留中のアップグレードを確認して適用し、すべてのリージョンのエンジンバージョンが一致し、グローバルクラスターフェイルオーバーがブロック解除されていることを確認してください。詳細については、「グローバルクラスターのスイッチオーバーまたはフェイルオーバーのブロック解除」を参照してください。

データの損失を最小限に抑えるため、この機能を使用する前に次のことを行うことをお勧めします。

- 各アプリケーションをオフラインにして、Amazon DocumentDB グローバルクラスターのプライマリクラスターへの書き込みが送信されないようにします。
- すべての Amazon DocumentDB セカンダリクラスターのラグタイムをチェックします。レプリケーションの遅延が最も少ないセカンダリリージョンを選択すると、現在障害が発生しているプライマリリージョンでのデータ損失を最小限に抑えることができます。Amazon CloudWatch 上で `GlobalClusterReplicationLag` メトリクスを確認して、グローバルクラスター内のすべての Amazon DocumentDB セカンダリクラスターのラグタイムを確認します。これらのメトリクスは、各セカンダリクラスターへのレプリケーションがプライマリクラスターに対してどの程度遅れているかをミリ秒単位で示します。

  Amazon DocumentDB 向け CloudWatch メトリクスの詳細については、「Amazon DocumentDB のメトリクス」を参照してください。

マネージドフェイルオーバー中、選択されたセカンダリクラスターは、プライマリとして新しいロールに昇格されます。ただし、プライマリクラスターのさまざまな設定オプションは引き継がれません。構成の不一致は、パフォーマンスの問題、ワークロードの非互換性、およびその他の異常な動作につながる可能性があります。このような問題を回避するには、Amazon DocumentDB グローバルクラスター間の次のような相違点を解消しておくことをお勧めします。

- 新しいプライマリ向けに Amazon DocumentDB のクラスターパラメータグループを構成する (必要な場合) — Amazon DocumentDB グローバルクラスター内の各クラスターごとに、Amazon

DocumentDB クラスターパラメータグループを構成できます。ただし、セカンダリクラスターを昇格してプライマリロールを引き継ぐ場合、セカンダリからのパラメータグループは、プライマリとは異なる設定になっている可能性があります。その場合は、昇格されたセカンダリクラスターのパラメータグループを、プライマリクラスターの設定に適合するように変更してください。この方法の詳細は、「Amazon DocumentDB クラスターパラメータグループを変更する」を参照してください。

- 各種モニタリングツールやオプションを構成する (Amazon CloudWatch Events やアラームなど) - グローバルクラスターに必要なログ機能、アラームなどを使用して、昇格されたクラスターの設定を行います。パラメータグループと同様に、フェイルオーバープロセス中にこれらの機能の設定がプライマリから継承されることはありません。レプリケーションラグなどの一部の CloudWatch メトリクスは、セカンダリリージョンのみで使用できます。そのため、フェイルオーバーによってメトリクスの表示方法やアラームの設定方法が変わり、定義済みのダッシュボードを変更する必要が生じる場合があります。Amazon DocumentDB クラスターとモニタリングの詳細については、「Amazon DocumentDB のモニタリング」を参照してください。

通常、選択したセカンダリクラスターは 1 分以内にプライマリロールを引き受けます。新しいプライマリリージョンのライターノードが使用可能になり次第、アプリケーションをそのライターノードに接続してワークロードを再開できます。Amazon DocumentDB が新しいプライマリクラスターを昇格させると、追加のセカンダリリージョンクラスターはすべて自動的にリビルドされます。

Amazon DocumentDB グローバルクラスターは非同期レプリケーションを使用するため、レプリケーションラグはセカンダリリージョンごとに異なる場合があります。これに対し Amazon DocumentDB は、これらのセカンダリリージョンを、新しいプライマリリージョンクラスターとまったく同じポイントインタイムデータを持つようにリビルドします。ストレージボリュームのサイズとリージョン間の距離によっては、再構築タスクが完了するまでに数分から数時間かかることがあります。セカンダリージョンのクラスターが新しいプライマリリージョンからの再構築を完了すると、読み取りアクセスが可能になります。新しいプライマリライターが昇格して使用可能になると、新しいプライマリリージョンのクラスターが、Amazon DocumentDB グローバルクラスターの読み取りおよび書き込みオペレーションを処理できるようになります。

グローバルクラスターの元のトポロジを復元するために、Amazon DocumentDB は古いプライマリリージョンの可用性を監視します。そのリージョンが再び正常な状態になり利用可能になるとすぐに、Amazon DocumentDB はそのリージョンをセカンダリリージョンとしてグローバルクラスターに自動的に追加します。古いプライマリリージョンに新しいストレージボリュームを作成する前に、Amazon DocumentDB は障害発生時の古いストレージボリュームのスナップショットを取得しようとします。これにより、欠落しているデータを回復することができます。このオペレーションが成功すると、Amazon DocumentDB は「rds:docdb-unplanned-global-failover-name-of-old-primary-

DB-cluster-timestamp」という名前のスナップショットを AWS Management Consoleのスナップショットセクションに格納します。このスナップショットは、`DescribeDBClusterSnapshots` API オペレーションによって返される情報にもリストアップされます。

> **Note**
>
> 古いストレージボリュームのスナップショットは、古い主クラスターに設定されたバックアップ保持期間の対象となるシステムスナップショットです。このスナップショットを保存期間外に保存するには、スナップショットをコピーして手動スナップショットとして保存できます。価格などのスナップショットのコピーの詳細については、「クラススナップショットのコピー」を参照してください。

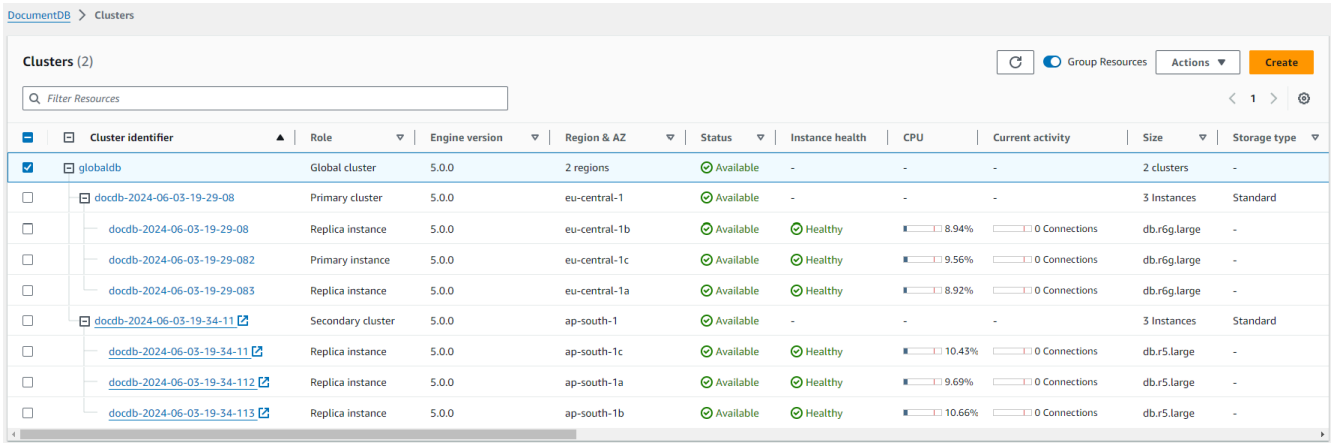
元のトポロジを復元できたら、ビジネスとワークロードにとって最も都合のよいタイミングでスイッチオーバーオペレーションを実行することで、グローバルクラスターを元のプライマリリージョンにフェイルバックします。これを行うには、「Amazon DocumentDB グローバルクラスターのスイッチオーバーの実行」の手順を実行します。

Amazon DocumentDB グローバルクラスターは AWS Management Console、、 AWS CLI、または Amazon DocumentDB API を使用してフェイルオーバーできます。

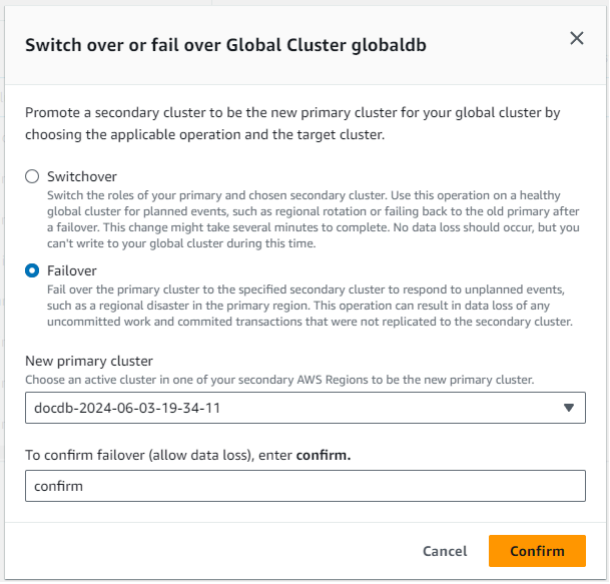
Using the AWS Management Console

Amazon DocumentDB グローバルクラスターでマネージドフェイルオーバーを実行する

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. フェイルオーバーしたい Amazon DocumentDB グローバルクラスターを選択します。

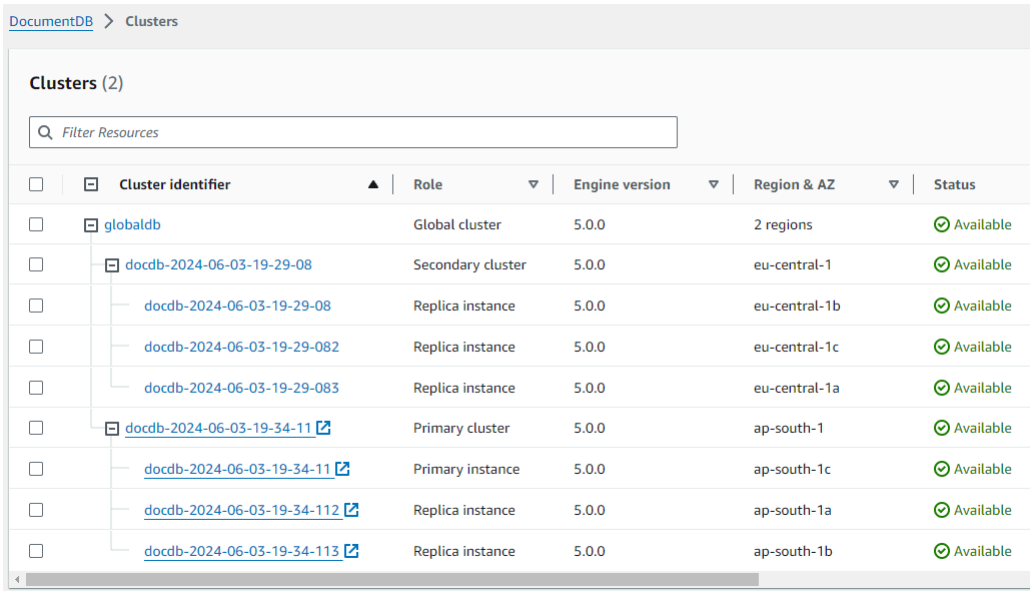


4. [アクション] メニューから [スイッチオーバーまたはフェイルオーバーする] を選択します。
5. 表示されるダイアログボックスで、[フェイルオーバー] を選択し、[新しいプライマリクラスター] フィールドのドロップダウンリストからセカンダリクラスターを選択します。



6. 最後のフィールドに「confirm」 (確定する) とキー入力します。[Confirm] (確認) を選択します。

   プライマリクラスターのステータスが「フェイルオーバー」に変わります。この状態になるまでには約 1 分かかります。この間、新しいプライマリクラスターのステータスは「変更中」と表示されます。新しいプライマリが昇格されると、「利用可能」の表示に変わり、読み取りおよび書き込みトランザクションを処理できるようになります。古いプライマリを含むセカンダリリージョンは、「再同期中」と表示され、新しいプライマリに再同期されます。新しいプライマリと同様に、古いプライマリも、ステータスが「利用可能」に変わってからのみトランザクションの処理が可能となります。
7. 完了すると、元のプライマリクラスターはセカンダリクラスターになります。選択したセカンダリクラスターがプライマリクラスターになります。



Using the AWS CLI

Amazon DocumentDB グローバルクラスターでマネージドフェイルオーバーを実行する

`failover-global-cluster` CLI コマンドを実行して、Amazon DocumentDB グローバルクラスターをフェイルオーバーします。コマンドを使用して、次のオプションの値を渡します。

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`
- `--allow-data-loss`

次の例では、各*###############*をクラスターの情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb failover-global-cluster \
   --region region_of_selected_secondary \
   --global-cluster-identifier global_cluster_id \
   --target-db-cluster-identifier arn_of_secondary_to_promote \
   --allow-data-loss
```

Windows の場合:

```
aws docdb failover-global-cluster ^
   --region region_of_selected_secondary ^
   --global-cluster-identifier global_cluster_id ^
   --target-db-cluster-identifier arn_of_secondary_to_promote ^
   --allow-data-loss
```

Amazon DocumentDB グローバルクラスターの手動フェイルオーバーの実行

1 つの 内のクラスター全体が使用 AWS リージョン できなくなった場合は、グローバルクラスター内の別のクラスターを昇格させて読み取り/書き込み機能を持つことができます。

別の AWS リージョン にあるクラスターがプライマリクラスターとしてより適格であると判断される場合は、グローバルクラスターフェイルオーバー機能を手動で起動することができます。たとえば、特定のセカンダリクラスターの容量を増やして、プライマリクラスターに昇格させることもできます。または、 間のアクティビティのバランスが AWS リージョン 変化する可能性があるため、プライマリクラスターを別のクラスターに切り替えると、書き込みオペレーションのレイテンシーが低下する AWS リージョン 可能性があります。

次の手順では、Amazon DocumentDB グローバルクラスターのセカンダリクラスターの 1 つを昇格させる方法を説明します。

セカンダリクラスターを昇格させるには

1. 停止 AWS リージョン した のプライマリクラスターへの DML ステートメントやその他の書き込みオペレーションの発行を停止します。
2. 新しいプライマリクラスター AWS リージョン として使用するセカンダリクラスターを特定します。グローバルクラスター AWS リージョン に 2 つ (またはそれ以上) のセカンダリがある場合は、ラグタイムが最も短いセカンダリクラスターを選択します。
3. 選択したセカンダリクラスターをグローバルクラスターからデタッチします。

   グローバルクラスターからセカンダリクラスターを削除すると、プライマリからこのセカンダリへのレプリケーションが直ちに停止され、完全な読み取り/書き込み機能を備えたスタンドアロンのプロビジョニングクラスターへと昇格されます。停止しているリージョン内のプライマリクラスターに関連付けられたその他のセカンダリクラスターは引き続き利用可能で、アプリケーションからの呼び出しを受け付けることができます。また、リソースを使用することになります。グローバルクラスターを再作成するため、スプリットブレインなどの問題を回避するために、以下のステップで新しいグローバルクラスターを作成する前に、他のセカンダリクラスターを削除します。

アタッチ解除の詳細なステップについては、Amazon DocumentDB グローバルクラスターからのクラスターの削除 を参照してください。

4. このクラスターは、次のステップでリージョンを追加すると、新しいグローバルクラスターのプライマリクラスターになります。
5. クラスター AWS リージョン に を追加します。これを行うと、プライマリからセカンダリへのレプリケーションプロセスがスタートされます。
6. 必要に応じて追加 AWS リージョン し、アプリケーションをサポートするために必要なトポロジを再作成します。グローバルクラスター内 (スプリットブレインの問題) のクラスター間のデータの不整合を避けるために、これらの変更を行う前、最中、および後に、アプリケーションの書き込みが正しいクラスターに送信されていることを確認してください。
7. 停止状態が解決されて元の AWS リージョン をプライマリクラスターとして再度割り当てる準備が完了したら、同じステップを逆に実行します。
8. グローバルクラスターからセカンダリクラスターの 1 つを削除します。これにより、読み取り/書き込みトラフィックを提供できるようになります。
9. 元の AWS リージョンのプライマリクラスターにすべての書き込みトラフィックをリダイレクトします。
10. を追加して AWS リージョン 、以前 AWS リージョン と同じ に 1 つ以上のセカンダリクラスターを設定します。

Amazon DocumentDB グローバルクラスターは AWS SDKs を使用して管理できるため、ディザスタリカバリとビジネス継続性計画のユースケースのグローバルクラスターフェイルオーバープロセスを自動化するソリューションを作成できます。そのようなソリューションの 1 つは、Apache 2.0 ライセンスに基づいてお客様に提供されており、こちらのツールリポジトリからアクセスできます。このソリューションは、エンドポイント管理に Amazon Route 53 を活用し、適切なイベントに基づいてトリガーできる AWS Lambda 機能を提供します。

Amazon DocumentDB グローバルクラスターのスイッチオーバーの実行

スイッチオーバーを使用すると、プライマリクラスターのリージョンを定期的に変更できます。この機能は、運用メンテナンス、その他の計画された運用手順など、管理されたシナリオを対象としています。

スイッチオーバーを使用する一般的なユースケースは 3 つあります。

- 特定の業界に課せられる「リージョナルローテーション」要件向け。たとえば、金融サービス規制では、ディザスタリカバリ手順が定期的に実施されるように、Tier-0 システムを別の地域に数か月間切り替えることが求められる場合があります。
- マルチリージョンの「follow-the-sun」アプリケーション向け。たとえば、ある企業が、さまざまなタイムゾーンの営業時間に基づいて、さまざまなリージョンで低レイテンシーの書き込みを提供したいとします。
- データ損失ゼロの方法として、フェイルオーバー後に元のプライマリリージョンにフェイルバックします。

> **Note**
>
> スイッチオーバーは、正常な Amazon DocumentDB グローバルクラスターでの使用向けに設計された機能です。予期しないシステム停止から回復するには、Amazon DocumentDB グローバルクラスターの手動フェイルオーバーの実行 の該当する手順に従ってください。
> スイッチオーバーを実行するには、すべてのセカンダリリージョンがプライマリとまったく同じエンジンバージョンを実行している必要があります。リージョンのエンジンバージョンが一致しない場合、スイッチオーバーはブロックされます。保留中のアップグレードを確認して適用し、すべてのリージョンのエンジンバージョンが一致し、グローバルクラスターのスイッチオーバーがブロック解除されていることを確認してください。詳細については、「グローバルクラスターのスイッチオーバーまたはフェイルオーバーのブロック解除」を参照してください。

スイッチオーバー中、Amazon DocumentDB は、グローバルクラスターの既存のレプリケーショントポロジを維持しながら、プライマリクラスターを選択されたセカンダリリージョンにスイッチオーバーします。なお、Amazon DocumentDB は、すべてのセカンダリリージョンクラスターがプライマリリージョンクラスターと完全に同期されるまで待ってからスイッチオーバープロセスを開始します。次に、プライマリリージョンの DB クラスターは読み取り専用になり、選択したセカンダリ DB クラスターは、読み取り専用ノードの 1 つを、フルライターステータスに昇格させます。このノードをライターに昇格させると、そのセカンダリクラスターがプライマリクラスターの役割を引き受けることができます。プロセスの開始時にはすべてのセカンダリクラスターがプライマリと完全に同期されているため、新しいプライマリはデータを失うことなく Amazon DocumentDB グローバルクラスターのオペレーションを継続できます。プライマリクラスターと選択したセカンダリクラスターが新しいロールを引き受ける間、短時間データベースが使用できなくなります。

アプリケーションの可用性を最適化するには、この機能を使用する前に、次の操作を行うことをお勧めします。

- この操作は、ピーク時以外の、プライマリクラスターへの書き込みが最小限である時間帯に実行してください。
- 各アプリケーションをオフラインにして、Amazon DocumentDB グローバルクラスターのプライマリクラスターへの書き込みが送信されないようにします。
- Amazon CloudWatch 上で `GlobalClusterReplicationLag` メトリクスを確認して、グローバルクラスター内のすべての Amazon DocumentDB セカンダリクラスターのラグタイムを確認します。このメトリクスは、セカンダリクラスターへのレプリケーションがプライマリクラスターに対してどのくらい遅れているかをミリ秒単位でを示します。この値は、Amazon DocumentDB がスイッチオーバーを完了するのにかかる時間と完全に比例します。したがって、遅延値が大きいほど、スイッチオーバーにかかる時間は長くなります。

  Amazon DocumentDB 向け CloudWatch メトリクスの詳細については、「Amazon DocumentDB のメトリクス」を参照してください。

スイッチオーバー中、選択したセカンダリ DB クラスターは、プライマリとして新しいロールに昇格されます。ただし、プライマリ DB クラスターのさまざまな設定オプションは継承されません。構成の不一致は、パフォーマンスの問題、ワークロードの非互換性、およびその他の異常な動作につながる可能性があります。このような問題を回避するには、Amazon DocumentDB グローバルクラスター間の次のような相違点を解消しておくことをお勧めします。

- 新しいプライマリ向けに Amazon DocumentDB のクラスターパラメータグループを構成する (必要な場合) — Amazon DocumentDB グローバルクラスター内の各クラスターごとに、Amazon DocumentDB クラスターパラメータグループを構成できます。つまり、セカンダリ DB クラスターを昇格してプライマリロールを引き継ぐ場合、セカンダリからのパラメータグループは、プライマリとは異なる設定になることがあります。その場合は、プロモートされたセカンダリ DB クラスターのパラメータグループを、プライマリクラスターの設定に適合するように変更します。この方法については、「Amazon DocumentDB クラスターパラメータグループの管理」を参照してください。
- 各種モニタリングツールやオプションを構成する (Amazon CloudWatch Events やアラームなど) - グローバルクラスターに必要なログ機能、アラームなどを使用して、昇格されたクラスターの設定を行います。パラメータグループと同様に、スイッチオーバープロセス中にこれらの機能の設定がプライマリから継承されることはありません。レプリケーションラグなどの一部の CloudWatch メトリクスは、プライマリリージョンのみで使用できます。そのため、スイッチオーバーによってメトリクスの表示方法やアラームの設定方法が変わり、定義済みのダッシュボードを変更する必要が

生じる場合があります。詳細については、「Amazon DocumentDB のモニタリング」を参照してください。

> **Note**
>
> 通常、ロールスイッチオーバーには数分かかることがあります。

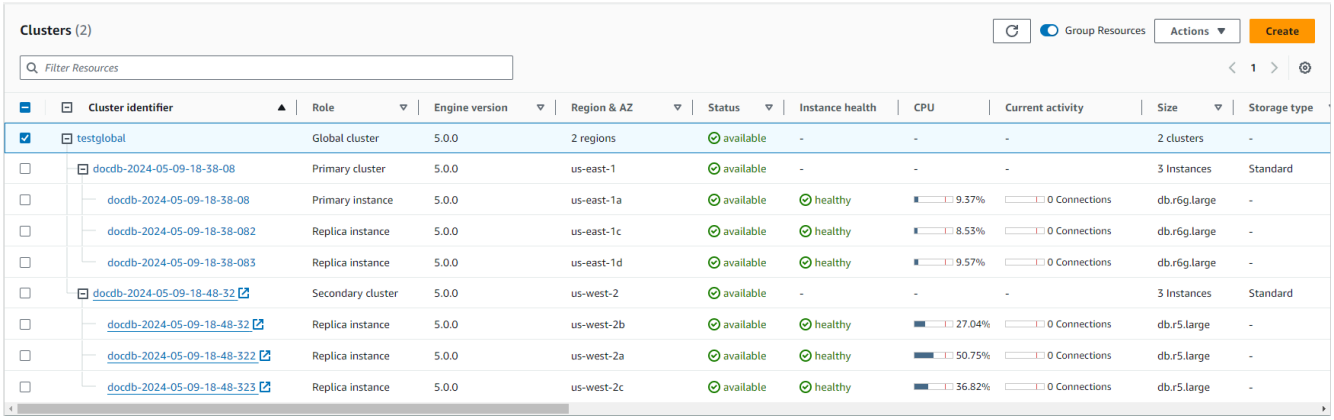
スイッチオーバープロセスが完了すると、昇格された Amazon DocumentDB クラスターはグローバルクラスターへの書き込みオペレーションを処理できるようになります。

AWS Management Console または を使用してAmazon DocumentDB グローバルクラスターを切り替えることができます AWS CLI。

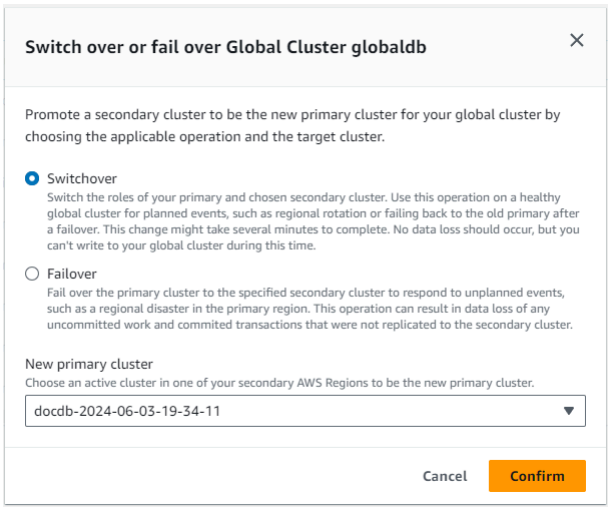
Using the AWS Management Console

Amazon DocumentDB グローバルクラスターでスイッチオーバーを実行する

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. スイッチオーバーしたい Amazon DocumentDB グローバルクラスターを検索して選択します。



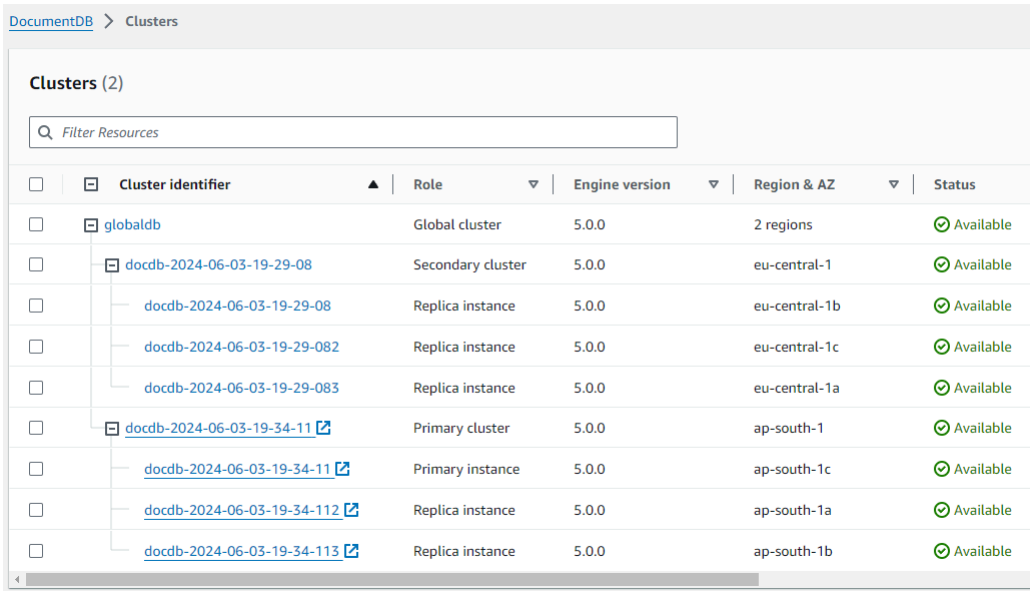
4. [アクション] メニューから [スイッチオーバーまたはフェイルオーバーする] を選択します。
5. 表示されるダイアログボックスで、[スイッチオーバー] を選択し、[新しいプライマリクラスター] フィールドのドロップダウンリストからセカンダリクラスターを選択します。



6. [確認] を選択してください。

   プライマリクラスターのステータスが「スイッチオーバー中」に変わります。この状態になるまでには約 3 分かかります。この間、すべてのリージョンクラスターのステータスは「変更中」と表示されます。すべてのリージョンが同期され、新しいプライマリが昇格されると、すべてのステータスフィールドの表示が「利用可能」となり、トランザクションを処理できるようになります。

7. 完了すると、元のプライマリクラスターはセカンダリクラスターになります。選択したセカンダリクラスターがプライマリクラスターになります。



Using the AWS CLI

Amazon DocumentDB グローバルクラスターでスイッチオーバーを実行する

switchover-global-cluster CLI コマンドを実行して、Amazon DocumentDB グローバルクラスターを切り替えます。コマンドを使用して、次のオプションの値を渡します。

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`

次の例では、各*###############*をクラスターの情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb switchover-global-cluster \
   --region region_of_primary \
   --global-cluster-identifier global_cluster_id \
   --target-db-cluster-identifier arn_of_secondary_to_promote
```

Windows の場合:

```
aws docdb switchover-global-cluster ^
   --region region_of_primary ^
   --global-cluster-identifier global_cluster_id ^
   --target-db-cluster-identifier arn_of_secondary_to_promote
```

グローバルクラスターのスイッチオーバーまたはフェイルオーバーのブロック解除

グローバルクラスターのすべてのリージョンクラスターが同じエンジンバージョンにない場合、グローバルクラスターのスイッチオーバーとフェイルオーバーはブロックされます。バージョンが一致しない場合、スイッチオーバーまたはフェイルオーバーを呼び出すと、このエラーが表示されることがあります。指定されたターゲット DB クラスターは、ソース DB クラスターとは異なるパッチレベルでエンジンバージョンを実行しています。最新のエンジンバージョンを定期的に適用して、最新の更新を実行し、グローバルクラスターを正常な状態に保つことをお勧めします。

このエラーを解決するには、まずすべてのセカンダリリージョンを更新し、次に保留中のメンテナンスアクション項目を適用してプライマリリージョンを同じエンジンバージョンに更新してください。保留中のメンテナンスアクション項目を表示し、問題を修正するために必要な変更を適用するには、次のいずれかのタブの手順を実行します。

Using the AWS Management Console

グローバルクラスターのスイッチオーバーまたはフェイルオーバーのブロックを解除するには、クラスターに対して保留中のメンテナンスアクションがあるかどうかを判断し、適用する必要があります。メンテナンスアクションを表示および適用するには、次の手順に従います。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. クラスターテーブルで、クラスター識別子列でグローバルクラスターを見つけます。グローバルクラスターで、指定されたグローバルクラスターの各セカンダリクラスターとプライマリクラスターを書き留め、それぞれに対して次の手順を実行します。
4. 各セカンダリクラスターについて：
   a. クラスターで更新が利用可能な場合は、メンテナンス列に「利用可能」、「必須」、または「次のウィンドウ」と表示されます。
   b. アクションを実行するには、クラスターを選択して詳細を表示し、メンテナンスとバックアップを選択します。[保留中のメンテナンス] 項目が表示されます。
   c. 説明 で、「新しいメンテナンス更新が利用可能」と示されている場合は、それを選択し、今すぐ適用を選択します。
5. プライマリクラスターの場合：
   a. クラスターで更新が利用可能な場合は、メンテナンス列に「利用可能」、「必須」、または「次のウィンドウ」と表示されます。
   b. アクションを実行するには、クラスターを選択して詳細を表示し、メンテナンスとバックアップを選択します。[保留中のメンテナンス] 項目が表示されます。
   c. 説明 で、「新しいメンテナンス更新が利用可能」と示されている場合は、それを選択し、今すぐ適用を選択します。

Using the AWS CLI

グローバルクラスターのスイッチオーバーまたはフェイルオーバーのブロックを解除するには、クラスターに対して保留中のメンテナンスアクションがあるかどうかを判断し、適用する必要があります。メンテナンスアクションを最初にセカンダリクラスターで表示して適用し、次にグローバルクラスターのプライマリクラスターで適用するには、次の手順に従います。

1. まず各セカンダリリージョンのリージョンクラスターで以下を実行し、次にプライマリリージョンのリージョンクラスターで以下を実行します。
2. `--resource-identifier` オプションを指定して [describe-pending-maintenance-actions](#) CLI コマンドを実行し、Amazon DocumentDB リージョンクラスターで利用可能なメンテナンスアクションがあるかどうかを確認します。

   次の例では、各*###############*をクラスターの情報に置き換えます。

   Linux、macOS、Unix の場合:

   ```
   aws docdb describe-pending-maintenance-action \
      --resource-identifier arn:aws:rds:us-
   east-1:001234567890:cluster:docdb-2025-03-27-19-21-15
   ```

   Windows の場合:

   ```
   aws docdb describe-pending-maintenance-action ^
      --resource-identifier arn:aws:rds:us-
   east-1:001234567890:cluster:docdb-2025-03-27-19-21-15
   ```

   結果は次のようになります。

   ```
   {
       "PendingMaintenanceActions": [
           {
               "ResourceIdentifier": "arn:aws:rds:us-
   east-1:001234567890:cluster:docdb-2025-03-27-19-21-15",
               "PendingMaintenanceActionDetails": [
                   {
                       "Action": "system-update",
                       "CurrentApplyDate": "2025-04-11T03:01:00Z",
                       "Description": "db-version-upgrade",
                       "ForcedApplyDate": "2025-06-18T03:01:00Z",
                       "AutoAppliedAfterDate": "2025-05-11T03:01:00Z"
                       "OptInStatus": "pending"
                   }
               ]
           }
       ]
   }
   ```

3. メンテナンスアクションが必要な場合は、次のオプションを指定して apply-pending-maintenance-action CLI コマンドを実行します。

   - --resource-identifier
   - --apply-action
   - --opt-in-type
   - --region

   次の例では、各###############をクラスターの情報に置き換えます。

   Linux、macOS、Unix の場合:

```
aws docdb apply-pending-maintenance-action \
   --resource-identifier arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15 \
   --apply-action system-update \
   --opt-in-type immediate \
   --region us-east-1
```

   Windows の場合:

```
aws docdb apply-pending-maintenance-action ^
   --resource-identifier arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15 ^
   --apply-action system-update ^
   --opt-in-type immediate ^
   --region us-east-1
```

4. メンテナンスアクションが完了したら、 describe-pending-maintenance-actions コマンドを再度実行して、クラスターに対して保留中の他のアクションがないことを確認します。

   必要な結果は次のとおりです。

```
{
    "PendingMaintenanceActions": []
}
```

グローバルクラスターのスイッチオーバーまたはフェイルオーバーのブロックを解除するには、クラスターに対して保留中のメンテナンスアクションがあるかどうかを判断し、適用する必要があります。次の APIsを使用して、メンテナンスアクションを表示および適用します。

1. まず各セカンダリリージョンのリージョンクラスターで以下を実行し、次にプライマリリージョンのリージョンクラスターで以下を実行します。
2. `PendingMaintenanceAction` API を呼び出して、Amazon DocumentDB グローバルクラスターで利用可能なメンテナンスアクションがあるかどうかを確認します。
3. `ApplyPendingMaintenanceAction` API を呼び出して変更を適用します。

Amazon DocumentDB クラスターの管理

Amazon DocumentDB クラスターを管理するには、適切な Amazon DocumentDB コントロールプレーンアクセス許可を持つ IAM ポリシーが必要です。これらのアクセス許可により、クラスターとインスタンスを作成、変更、削除できます。`AmazonDocDBFullAccess` ポリシーは、 Amazon DocumentDB クラスターを管理するために必要なすべてのアクセス許可を提供します。

以下のトピックでは、Amazon DocumentDB クラスターを操作するときのさまざまなタスクの実行方法を示します。これには、クラスターの作成、削除、変更、接続、表示が含まれます。

トピック

- クラスターについて
- Amazon DocumentDB クラスターの設定
- Amazon DocumentDB クラスターストレージ設定
- クラスターのステータスの評価
- Amazon DocumentDB クラスターのライフサイクル
- Amazon DocumentDB クラスターのスケーリング
- Amazon DocumentDB クラスターのボリュームのクローン作成
- Amazon DocumentDB クラスターの耐障害性について理解する

クラスターについて

Amazon DocumentDB は、コンピューティングとストレージを分離し、データレプリケーションとバックアップをクラスターボリュームにオフロードします。クラスターボリュームは、3 つのアベイラビリティーゾーンにわたって 6 つの方法でデータをレプリケートする、耐久性、信頼性、可用性に優れたストレージレイヤーを提供します。レプリカにより、データの可用性と読み取りスケーリングが高まります。各クラスターは、最大 15 個のレプリカにスケールアップできます。

名詞	説明	API オペレーション (動詞)

クラスター	1 つ以上のインスタンスと、これらのインスタンスのデータを管理する 1 つのクラスターストレージボリュームで構成されます。	`create-db-cluster`
		`delete-db-cluster`
		`describe-db-clusters`
		`modify-db-cluster`

インスタンス	クラスターストレージボリュームとのデータの読み取りと書き込みは、インスタンスを介して行われます。クラスターには、プライマリとレプリカの 2 種類のインスタンスがあります。クラスターには常に 1 つのプライマリインスタンスがあり、0 ~ 15 のレプリカを持つことができます。	`create-db-instance`
		`delete-db-instance`
		`describe-db-instances`
		`modify-db-instance`
		`describe-orderable-db-instance-options`
		`reboot-db-instance`

クラスターボリューム	3 つのアベイラビリティーゾーンにまたがる仮想データベースストレージボリューム。各アベイラビリティーゾーンにはクラスターデータの 2 つのコピーが格納されます。	該当なし

名詞	説明	API オペレーション (動詞)
プライマリインスタンス	読み取りと書き込みの両方のオペレーションをサポートし、クラスターボリュームに対するすべてのデータ変更を実行します。各クラスターには 1 つのプライマリインスタンスがあります。	該当なし
レプリカインスタンス	読み取りオペレーションのみをサポートします。各 Amazon DocumentDB クラスターには、プライマリインスタンスに加えて 15 個までのレプリカインスタンスを含めることができます。複数のレプリカが読み取りワークロードを分散します。別のアベイラビリティーゾーンにレプリカを配置することによって、データベースの可用性を高めることもできます。	該当なし
クラスターエンドポイント	クラスターの現在のプライマリインスタンスに接続する Amazon DocumentDB クラスターのエンドポイント。Amazon DocumentDB クラスターごとに 1 つのクラスターエンドポイントと 1 つのプライマリインスタンスがあります。	該当なし

名詞	説明	API オペレーション (動詞)
リーダーエンドポイント	Amazon DocumentDB クラスターで利用できるレプリカの 1 つに接続する、そのクラスターのエンドポイント。各 Amazon DocumentDB クラスターには、リーダーエンドポイントがあります。複数のレプリカがある場合、リーダーエンドポイントは各接続リクエストを Amazon DocumentDB レプリカのいずれかにルーティングします。	該当なし
インスタンスエンドポイント	特定のインスタンスに接続する Amazon DocumentDB クラスターのインスタンス用のエンドポイント。クラスターの各インスタンスには、インスタンスタイプにかかわらず、独自のインスタンスエンドポイントがあります。	該当なし

Amazon DocumentDB クラスターの設定

クラスターを作成または変更するときは、どのパラメータが変更不可能で、どのパラメータがクラスターの作成後に変更可能かを理解することが重要です。次の表に、クラスターに固有のすべての設定 (つまり、パラメータ) をリストします。表に示されているように、変更可能なものもあれば、そうでないものもあります。

> **Note**
>
> これらの設定は、Amazon DocumentDB クラスターのパラメータグループおよびそのパラメータと混同しないでください。クラスターのパラメータグループの詳細については、「Amazon DocumentDB クラスターパラメータグループの管理」を参照してください。

パラメータ	変更可能	コメント
DBClusterIdentifier	はい	命名に関する制約: • 1 – 63 個の英字、数字、またはハイフンで構成します。 • 1 字目は英字である必要があります。 • ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。 • Amazon RDS、Amazon Neptune、Amazon DocumentDB のすべてのクラスターで AWS アカウント、リージョンごとに一意である必要があります。
Engine	いいえ	docdb を指定してください。
BackupRetentionPeriod	はい	1 ~ 35 日を指定する必要があります。
DBClusterParameterGroupName	はい	命名に関する制約: • 長さは 1 ~ 255 バイト個の英数字です。 • 1 字目は英字である必要があります。 • ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
DBSubnetGroupName	いいえ	クラスターが作成された後で、クラスターのサブネットを変更することはできません。
EngineVersion	いいえ	値は `5.0.0` (デフォルト)、`4.0.0`、または `3.6.0` です。
KmsKeyId	いいえ	クラスターの暗号化を選択した場合、クラスターの暗号化に使用した AWS KMS キーを変更することはできません。
MasterUsername	いいえ	クラスターが作成された後で、`MasterUsername` を変更することはできません。

パラメータ	変更可能	コメント
		命名に関する制約: • 長さは 1 ~ 63 バイトの英数字です。 • 1 字目は英字である必要があります。 • データベースエンジンの予約語は使用できません。

MasterUserPassword	はい	制約:
		• 長さは 8 ~ 100 バイトの印刷可能な ASCII 文字です。
		• 以下のものを除くすべての印刷可能な ASCII 文字を使用できます。
		  • **/** (スラッシュ)
		  • **"** (二重引用符)
		  • **@** (アットマーク)

Port	はい	ポート番号は、クラスター内のすべてのインスタンスに適用されます。
PreferredBackupWindow	はい	
PreferredMaintenanceWindow	はい	
StorageEncrypted	いいえ	クラスターの暗号化を選択した場合は、暗号化を解除することはできません。

パラメータ	変更可能	コメント
StorageType	はい	DB クラスターのストレージタイプ: 標準 (standard) または I/O 最適化 (iopt1)。 デフォルト: standard このパラメータは、CreateDBCluster と ModifyDBCluster で設定できます。 詳細については、「Amazon DocumentDB クラスターストレージ設定」を参照してください。
Tags	はい	
VpcSecurityGroupIds	いいえ	クラスターが作成された後で、クラスターが存在する VPC を変更することはできません。

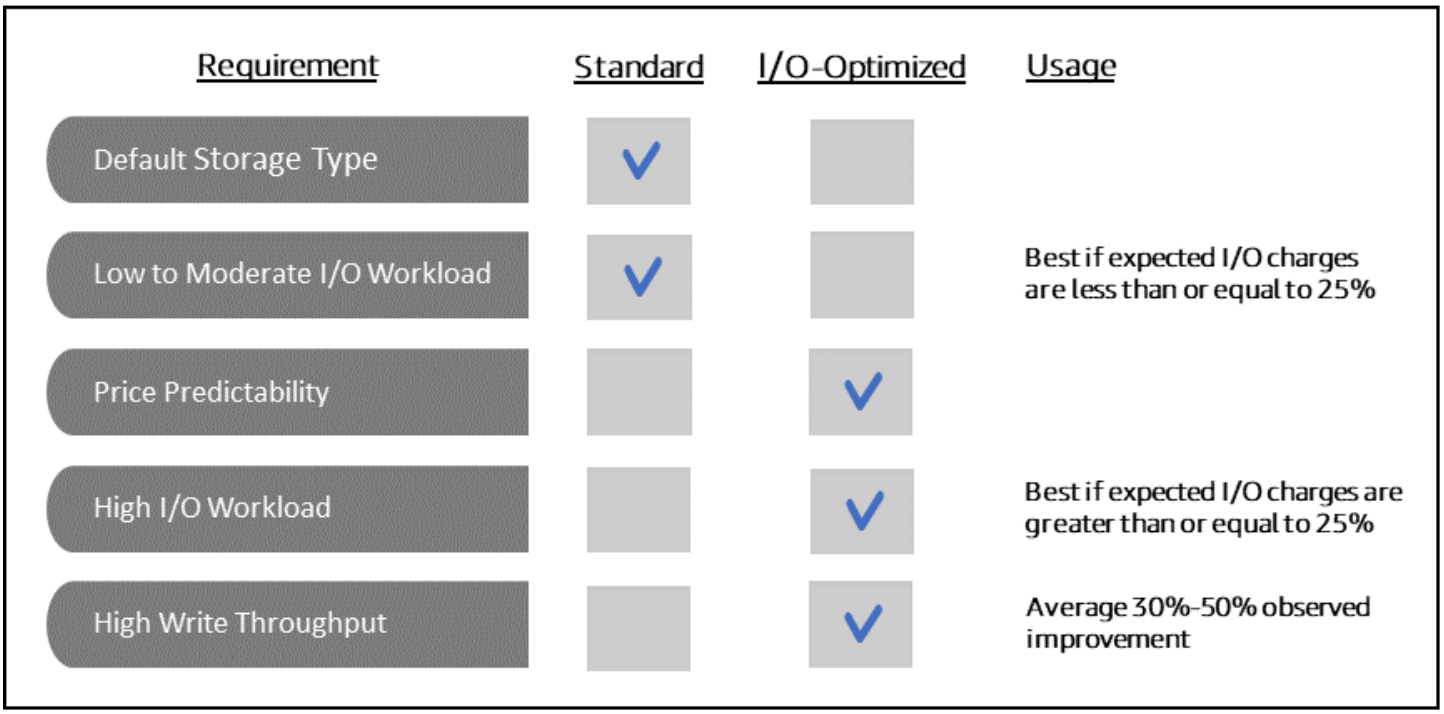
Amazon DocumentDB クラスターストレージ設定

Amazon DocumentDB 5.0 以降、インスタンスベースのクラスターは 2 つのストレージ設定タイプをサポートしています。

- Amazon DocumentDB 標準ストレージ: I/O 消費量が低から中レベルのお客様向けに設計されたオプションです。I/O コストが Amazon DocumentDB クラスター全体の 25% 未満になると予想される場合は、このタイプが適していると思われます。Amazon DocumentDB 標準ストレージ設定では、インスタンスとストレージの料金に加えて、リクエスト回数に応じて I/O 料金が請求されます。このため、請求額は使用量に応じてサイクルごとに異なります。設定は、お使いのアプリケーションの I/O 需要の変動に合わせて調整されます。
- Amazon DocumentDB I/O 最適化ストレージ : コストの予測可能性を優先するか、あるいは I/O 集約型アプリケーションを使用するお客様向けに設計されたオプションです。I/O 最適化設定は、I/O 集約型ワークロードを使用するお客様向けに、パフォーマンスの向上、スループットの向上、レイテンシーの低減を実現します。I/O コストが Amazon DocumentDB クラスターコスト全体の 25% を超えると予想される場合は、このオプションを選択いただくと価格パフォーマンスが向上します。Amazon DocumentDB I/O 最適化ストレージ設定では、I/O オペレーション回数に基づく従量課金は行われず、請求サイクルごとの請求額を確実に予想できます。この設定により、パフォーマンスを向上させながらコストを安定させることができます。

30 日に 1 回 のタイミングで、既存のデータベースクラスターを Amazon DocumentDB I/O 最適化ストレージに切り替えることができます。Amazon DocumentDB 標準ストレージにはタイミングを問わず随時切り替え可能です。ストレージ設定を I/O 最適化に変更する次の日付は、 `describe-db-clusters` コマンドを使用して、 AWS CLI またはクラスターの設定ページの AWS Management Console を使用して追跡できます。

Amazon DocumentDB I/O 最適化設定を含む新しいデータベースクラスターを作成するか、あるいは AWS Management Console 上での数回のクリック、もしくは AWS Command Line Interface (AWS CLI) での 1 つのパラメータの変更、または AWS SDK の使用により既存のデータベースクラスターを変換することができます。ストレージ設定の変更中、または変更後にダウンタイムは発生せず、インスタンスの再起動の必要もありません。

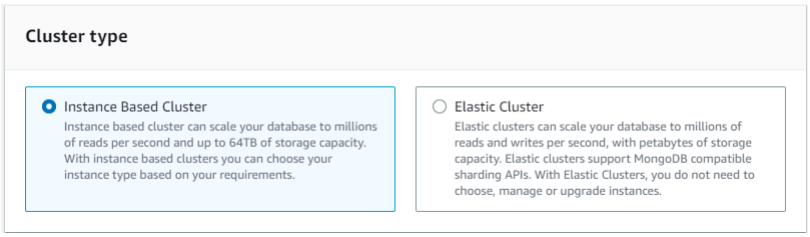


I/O 最適化クラスターの作成

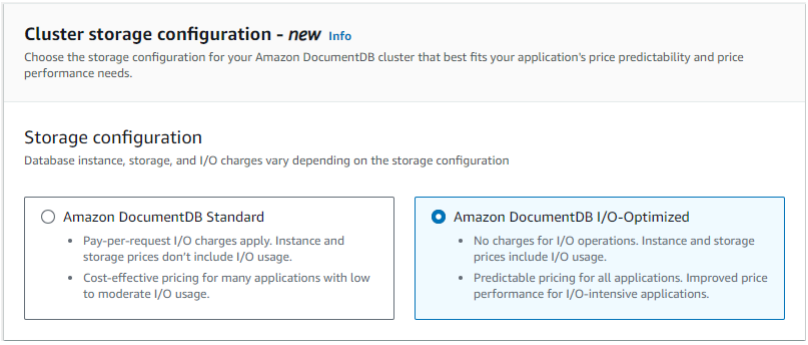
Using the AWS Management Console

を使用して I/O 最適化クラスターを作成または変更するには AWS Management Console：

1. Amazon DocumentDB 管理コンソール内の [クラスター] で、[作成] を選択するか、もしくは必要なクラスターを選択して [アクション] を選択し、[変更] を選択します。
2. 新しいクラスターを作成する場合は、クラスタータイプセクションでインスタンスベースのクラスターを選択してください (これがデフォルトのオプションです)。



3. 設定セクションのクラスターストレージ設定で、Amazon DocumentDB I/O 最適化 を選択します。



4. クラスターの作成または変更を完了したら、[クラスターの作成] または [クラスターの変更] を選択します。

   クラスターの作成プロセスの詳細については、「を使用したクラスターとプライマリインスタンスの作成 AWS Management Console」を参照してください。

   クラスターの変更プロセスの詳細については、「Amazon DocumentDB クラスターの変更」を参照してください。

Using the AWS CLI

を使用して I/O 最適化クラスターを作成するには AWS CLI :

次の例では###############をユーザー自身の情報で置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb create-db-cluster \
      --db-cluster-identifier sample-cluster \
      --engine docdb \
      --engine-version 5.0.0 \
      --storage-type iopt1 \
      --deletion-protection \
      --master-username username \
```

```
      --master-user-password password
```

Windows の場合:

```
aws docdb create-db-cluster ^
      --db-cluster-identifier sample-cluster ^
      --engine docdb ^
      --engine-version 5.0.0 ^
      --storage-type iopt1 ^
      --deletion-protection ^
      --master-username username ^
      --master-user-password password
```

適切なストレージ設定を決定するためのコスト分析

Amazon DocumentDB では、データベースクラスターごとにストレージ設定を柔軟に選択できます。標準クラスターと I/O 最適化クラスターを適切に使い分けるため、Amazon DocumentDB のコストをクラスター単位で追跡できます。そのためには、既存のクラスターにタグを追加し、AWS Billing and Cost Management ダッシュボード で [コスト配分タグ付け] を有効にすることで、AWS Cost Explorer Service での個別のクラスターのコストを分析できます。コスト分析の詳細については、ブログ「コスト配分タグの使用」を参照してください。

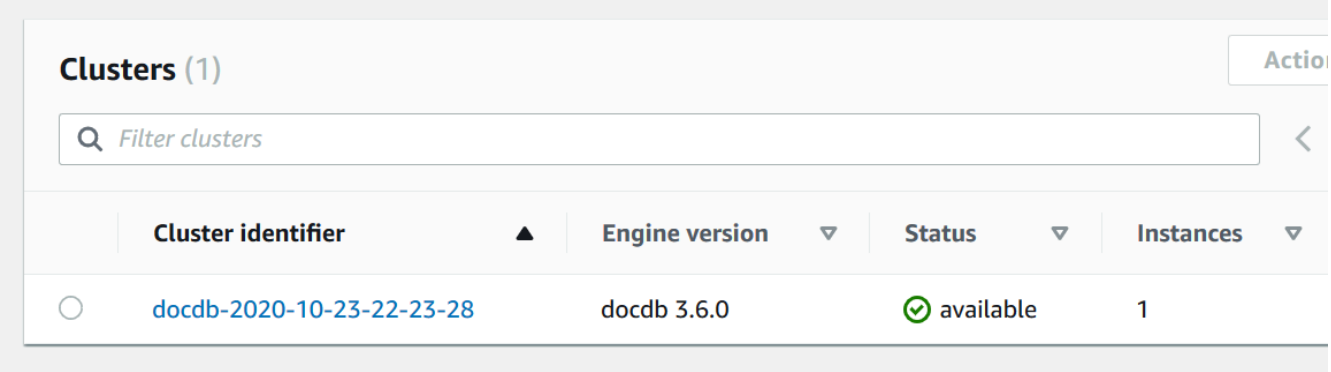
クラスターのステータスの評価

クラスターのステータスは、 AWS Management Console または を使用して判断できます AWS CLI。

Using the AWS Management Console

次の手順を使用して、 を使用して Amazon DocumentDB クラスターのステータスを確認します。 AWS Management Console

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. [クラスター ID] 列で、関心があるクラスターの名前を探します。次に、クラスターのステータスを確認するには、次に示すように行全体を参照して [ステータス] 列を見つけます。



Using the AWS CLI

`describe-db-clusters` オペレーション を使用して、 AWS CLIを使用した Amazon DocumentDB クラスターのステータスを確認します。

次のコードでは、クラスター `sample-cluster` のステータスを確認します。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster  \
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Windows の場合:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster  ^
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
[
    [
        "sample-cluster",
        "available"
    ]
]
```

Amazon DocumentDB クラスターのライフサイクル

Amazon DocumentDB クラスターのライフサイクルには、クラスターの作成、記述、変更、削除が含まれます。このセクションでは、これらのプロセスを完了する方法について説明します。

トピック

- Amazon DocumentDB クラスターの作成
- Amazon DocumentDB クラスター の説明
- Amazon DocumentDB クラスターの変更
- 実施待ちのメンテナンスの確認
- クラスターのエンジンバージョンへのパッチ更新の実行
- Amazon DocumentDB クラスターの停止と開始
- Amazon DocumentDB クラスターの削除

Amazon DocumentDB クラスターの作成

Amazon DocumentDB クラスターは、インスタンスと、クラスターのデータを表すクラスターボリュームで構成されます。クラスターボリュームは、3 つのアベイラビリティーゾーン間で 1 つの仮想ボリュームとして 6 つの方法でレプリケートされます。クラスターには、プライマリインスタンスと、オプションで最大 15 個のレプリカインスタンスが含まれます。

以下のセクションでは、 AWS Management Console または を使用して Amazon DocumentDB クラスターを作成する方法を示します AWS CLI。次に、そのクラスターのレプリカインスタンスを追加できます。コンソールを使用して Amazon DocumentDB クラスターを作成すると、プライマリインスタンスが自動的に作成されます。を使用して Amazon DocumentDB クラスター AWS CLI を作成する場合、クラスターのステータスが使用可能になったら、そのクラスターのプライマリインスタンスを作成する必要があります。

前提条件

Amazon DocumentDB クラスターを作成するための前提条件を以下に示します。

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

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

サインアップ手順の一環として、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。

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

VPC 前提条件

Amazon DocumentDB クラスターは Amazon Virtual Private Cloud (Amazon VPC) でのみ作成できます。Amazon VPC には、Amazon DocumentDB クラスターで使用するために、少なくとも 2 つのアベイラビリティーゾーンのそれぞれに、1 つ以上のサブネットが必要です。アベイラビリティーゾーンにまたがってクラスターインスタンスを配信することで、ごく稀にアベイラビリティーゾーンに障害が発生した場合でも、クラスター内で使用できるインスタンスを確保できます。

サブネットの前提条件

Amazon DocumentDB クラスターを作成するときは、その VPC 内の VPC と対応するサブネットグループを選択して、クラスターを起動する必要があります。サブネットにより、アベイラビリティーゾーン と、そのアベイラビリティーゾーン内でインスタンスの起動に使用する IP 範囲が決まります。ここでは、サブネットとアベイラビリティーゾーンという用語を同じ意味で使用します。サブネットグループは、サブネット (またはアベイラビリティーゾーン) の名前付きセットです。サブネットグループでは、Amazon DocumentDB インスタンスの起動に使用するアベイラビリティーゾーンを指定できます。たとえば、3 つのインスタンスを含むクラスターでは、各インスタンスを別々のアベイラビリティーゾーンにプロビジョニングして高可用性を実現することをお勧めします。これにより、1 つのアベイラビリティーゾーンがダウンした場合、1 つのインスタンスにのみ影響します。

Amazon DocumentDB インスタンスは現在、最大 3 つの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。サブネットグループに 3 つ以上のサブネットがある場合でも、それらのサブネットのうちの 3 つのみを使用して Amazon DocumentDB クラスターを作成できます。したがって、サブネットグループを作成するときは、インスタンスのデプロイ先として 3 つのサブネットのみを選択することをお勧めします。米国東部 (バージニア北部) では、サブネットグループに 6 つのサブネット (アベイラビリティーゾーン) を含めることができます。ただし、Amazon DocumentDB クラスターがプロビジョニングされると、Amazon DocumentDB によってインスタンスのプロビジョニングに使用される 3 つのアベイラビリティーゾーンが選択されます。

例えば、クラスターを作成する際、Amazon DocumentDB でアベイラビリティーゾーン (1A、1B、および 1C) を選択するとします。この場合、アベイラビリティーゾーン {1D} にインスタンスを作成しようとすると、その API コールは失敗します。ただし、特定のアベイラビリティーゾーンを指定せずにインスタンスを作成した場合、Amazon DocumentDB がお客様に代わってアベイラビリティーゾーンを選択します。Amazon DocumentDB では、複数のアベイラビリティーゾーン間でロードバランスのインスタンスの負荷を分散するアルゴリズムを使用して、高可用性が実現されます。たとえば、3 つのインスタンスがプロビジョニングされる場合、デフォルトでは 3 つのアベイラビリティーゾーンに別々にプロビジョニングされ、1 つのアベイラビリティーゾーンにまとめてプロビジョニングされません。

推奨事項:

- 特に理由がない限り、常に 3 つのサブネットを含むサブネットグループを作成します。その結果、3 つのアベイラビリティーゾーン間でインスタンスがプロビジョニングされるため、3 つ以上のインスタンスを含むクラスターで高可用性を実現できます。
- 高可用性を実現するには、常に複数のアベイラビリティーゾーンにインスタンスを分散させます。クラスターのすべてのインスタンスを 1 つのアベイラビリティーゾーンにまとめて配置しないでください。
- フェイルオーバーイベントはいつでも発生する可能性があるため、プライマリインスタンスまたはレプリカインスタンスが常に特定のアベイラビリティーゾーンに存在すると想定しないでください。

追加の前提条件

Amazon DocumentDB クラスターを作成するための追加の前提条件をいくつか次に示します。

- AWS Identity and Access Management (IAM) 認証情報 AWS を使用して に接続する場合、IAM アカウントには、Amazon DocumentDB 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。

  IAM アカウントを使用して Amazon DocumentDB コンソールにアクセスする場合は、まず IAM アカウント AWS Management Console で にサインインする必要があります。Amazon DocumentDB コンソールを次の場所で開きます。https://console.aws.amazon.com/docdb
- クラスターの設定パラメータを調整する場合は、必須のパラメータ設定を定義したクラスターパラメータグループとパラメータグループを指定する必要があります。クラスターパラメータグループまたはパラメータグループの作成または変更については、「Amazon DocumentDB クラスターパラメータグループの管理」を参照してください。

- クラスター用に指定する TCP/IP ポート番号を決定する必要があります。会社のファイアウォールによっては、Amazon DocumentDB のデフォルトポートへの接続がブロックされます。会社のファイアウォールがデフォルトのポートをブロックする場合は、お客様のクラスター用に別のポートを選択します。クラスターのすべてのインスタンスは同じポートを使用します。

を使用したクラスターとプライマリインスタンスの作成 AWS Management Console

以下の手順では、コンソールを使用して 1 つまたは複数のインスタンスで Amazon DocumentDB クラスターを起動する方法について説明します。

クラスターの作成: デフォルト設定を使用した作成

を使用してデフォルト設定を使用してインスタンスを含むクラスターを作成するには AWS Management Console

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. 米国東部 (バージニア北部) リージョン AWS リージョン 以外の でクラスターを作成する場合は、コンソールの右上セクションにあるリストからリージョンを選択します。
3. 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。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン
   > (≡)
   > を選択します。

4. [Amazon DocumentDB クラスターの作成] ページで、[設定] ペインに入力します。

   a. クラスター識別子 – Amazon DocumentDB 指定の名前を使用するか、クラスターの名前 (**`sample-cluster`** など) を使用します。

      クラスターの名前付けの制約:

      - 長さは [1 ～ 63] 個の英字、数字、またはハイフンです。
      - 1 字目は英字である必要があります。
      - ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

- Amazon RDS、Neptune、Amazon DocumentDB のすべてのクラスターで AWS アカウント、リージョンごとに一意である必要があります。

b. エンジンバージョン — デフォルトのエンジンバージョン 5.0.0 を受け入れるか、オプションで 4.0.0 または 3.6.0 を選択します。

c. [インスタンスクラス] - デフォルトの `db.r5.large` を使用するか、リストから必要なインスタンスクラスを選択します。

d. [インスタンス数] - リストで、このクラスターに作成するインスタンスの数を選択します。最初のインスタンスはプライマリインスタンスになり、他のインスタンスはすべて読み取り専用レプリカインスタンスになります。後でインスタンスを追加したり削除したりできます。デフォルトでは、Amazon DocumentDB クラスターは、3 つのインスタンス (1 つのプライマリと 2 つのレプリカ) 構成で起動します。

5. [クラスターストレージ設定] セクションの設定を完了します。

   [Amazon DocumentDB 標準] (デフォルト) または [Amazon DocumentDB I/O 最適化] を選択します。詳細については、「Amazon DocumentDB クラスターストレージ設定」を参照してください。

6. [認証] ペインに入力します。

   a. [ユーザー名] - プライマリユーザーの名前を入力します。クラスターにログインする際には、このプライマリユーザー名を使用する必要があります。

      プライマリユーザーの命名に関する制約事項:

      - 長さは 1 ~ 63 バイトの英数字です。
      - 1 字目は英字である必要があります。
      - データベースエンジンの予約語は使用できません。

   b. 次のいずれかのパスワードオプションを選択します。

      - Managed in AWS Secrets Manager — プライマリユーザーのパスワードを自動的に管理する場合は AWS Secrets Manager 、このオプションを選択します。

        このオプションを選択した場合は、独自の を作成するか、Secrets Manager が作成するキーを使用して KMS キーを設定します。

      - セルフマネージド - プライマリユーザーのパスワードを自己管理する場合は、このオプションを選択します。このオプションを選択した場合は、プライマリユーザーのパスワー

ドを入力し、確認します。クラスターにログインする際には、このプライマリユーザーパスワードを使用する必要があります。

パスワードの制約:

- 長さは、[8 ~ 100] 個の印刷可能な ASCII 文字です。
- 以下のものを除くすべての印刷可能な ASCII 文字を使用できます。
  - **/** (スラッシュ)
  - **"** (二重引用符)
  - **@** (アットマーク)

7. 画面の下部で、以下のいずれかを選択します。
   - ここでクラスターを作成するには、[クラスターの作成] を選択します。
   - クラスターを作成しない場合は、[Cancel (取り消し)] を選択します。
   - 作成前にクラスターをさらに設定するには、[Show additional configurations (追加設定の表示)] を選択し、「クラスターを作成する: その他の設定」に進みます。

   [追加設定] セクションの設定内容は次のとおりです。

   - [ネットワーク設定] - デフォルトでは、`default` VPC セキュリティグループを使用します。
   - [クラスターオプション] - デフォルトでは、ポート 27017 およびデフォルトのパラメータグループを使用します。
   - [暗号化] - デフォルトでは、`(default) aws/rds` キーを使用した暗号化が有効になります。

     > **Important**
     > クラスターはいったん暗号化されると、復号できません。

   - バックアップ - デフォルトでは、バックアップが 1 日間保持され、Amazon DocumentDB によってバックアップウィンドウが選択されます。
   - ログのエクスポート - デフォルトでは、監査ログは CloudWatch Logs にエクスポートされません。
   - メンテナンス - デフォルトでは、Amazon DocumentDB によってメンテナンスウィンドウが選択されます。

- [削除保護] - クラスターが誤って削除されることから保護します。コンソールを使用して作成されたクラスターのデフォルトでは、有効 になっています。

デフォルト設定を使用しても、これらの設定のほとんどは、後でクラスターを修正して変更できます。

8. クラスターのセキュリティグループに対してインバウンド接続を有効にします。

クラスターのデフォルト設定を変更しなかった場合は、特定のリージョン内のデフォルト VPC のデフォルトセキュリティグループを使用してクラスターが作成されました。Amazon DocumentDB に接続するには、クラスターのセキュリティグループのポート 27017 (または選択したポート) でインバウンド接続を有効にする必要があります。

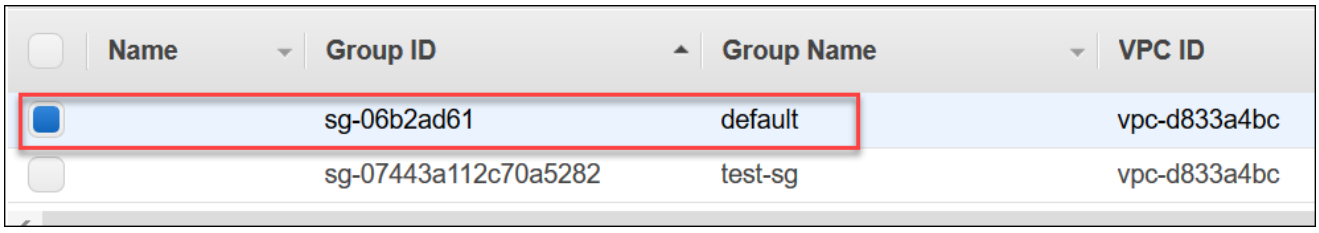
クラスターのセキュリティグループにインバウンド接続を追加するには

a. にサインイン AWS Management Console し、https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。

b. メインウィンドウの [Resources (リソース)] セクションで、[Security groups (セキュリティグループ)] を選択します。



c. セキュリティグループのリストから、クラスターの作成時に使用したセキュリティグループを探し (おそらくデフォルトのセキュリティグループです)、セキュリティグループの名前の左側にあるボックスを選択します。



d. [Actions (アクション)] メニューから、[Edit inbound rules (インバウンドルールを編集する)] を選択し、次にルールの制約を選択するか入力します。

   i. タイプ - リストから、プロトコルを選択してネットワークトラフィックを開きます。

   ii. [プロトコル] - リストから、プロトコルのタイプを選択します。

   iii. [ポート範囲] - カスタムルールでは、ポート番号またはポート範囲を選択します。ポート番号またはポート範囲には、クラスターを作成したときに指定したポートが含まれていることを確認してください (デフォルト: 27017)。

   iv. ソース - インスタンスに到達できるトラフィックを指定します。リストからトラフィックの送信元を選択します。[Custom (カスタム)] を選択する場合は、単一の IP アドレスまたは IP アドレス範囲を CIDR 表記で指定します (例: 203.0.113.5/32)。

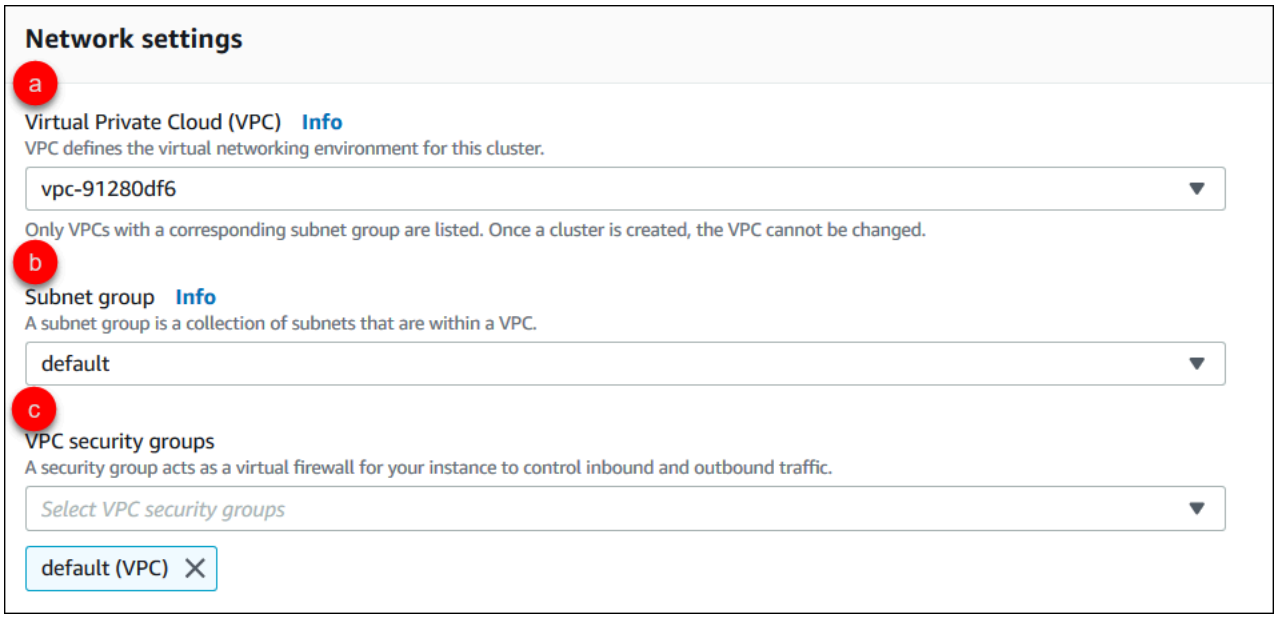
   v. [Description] - ルールの説明を入力します。

   vi. ルールの作成を終了したら、[Save (保存)] を選択します。

クラスターを作成する: その他の設定

クラスターのデフォルト設定を使用する場合は、以下のステップをスキップして [クラスターの作成] を選択します。

1. [Network settings (ネットワーク設定)] ウィンドウに入力します。



   a. [Virtual Private Cloud (VPC)] - リストで、このクラスターを起動する Amazon VPC を選択します。

b. [サブネットグループ] - リストで、このクラスターに使用するサブネットグループを選択します。

c. [VPC セキュリティグループ] - リストで、このクラスターの VPC セキュリティグループを選択します。

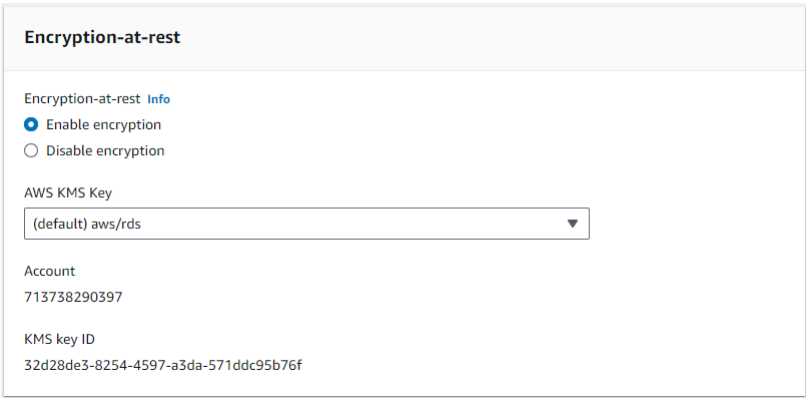
2. [Cluster options (クラスターオプション)] ペインに入力します。



a. [データベースポート] - 上下の矢印を使用して、アプリケーションによってインスタンスへの接続に使用される TCP/IP ポートを設定します。

b. [クラスターパラメータグループ] - パラメータグループのリストで、このクラスターのクラスターパラメータグループを選択します。

3. [Encryption (暗号化)] ペインに入力します。



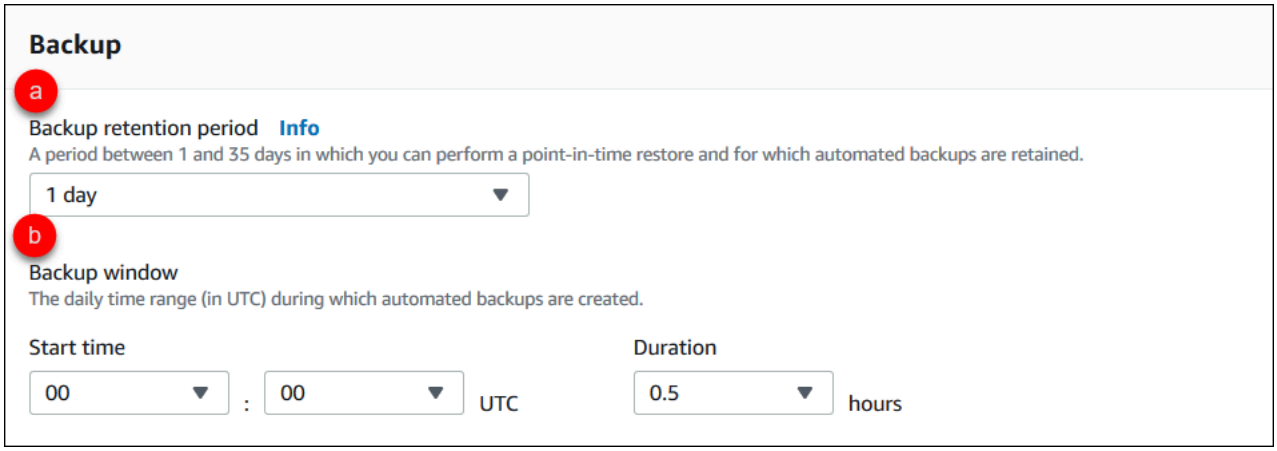
a. [保管時に暗号化] - 以下のいずれかを選択します。

- 暗号の有効化 - デフォルト すべてのデータは保管時に暗号化されます。データの暗号化を選択した場合、この操作を元に戻すことはできません。
- [暗号化の無効化] - データは暗号化されません。

b. AWS KMS キー — これはデータを暗号化している場合にのみ使用できます。リストで、このクラスターのデータの暗号化に使用するキーを選択します。デフォルトは (default) aws/rds です。

[キーの ARN を入力] を選択した場合は、キーの Amazon リソースネーム (ARN) を入力する必要があります。

4. [バックアップ] ペインに入力します。



a. [バックアップ保持期間] - リストで、このクラスターの自動バックアップを削除するまで保持する日数を選択します。

b. [バックアップウィンドウ] - Amazon DocumentDB によってこのクラスターのバックアップが作成される毎日の時刻と期間を設定します。

i. [開始時刻] - 最初のリストで、自動バックアップを開始する時間 (UTC 時) を選択します。2 番目のリストで、自動バックアップを開始する時間 (UTC 分) を選択します。

ii. [期間] - リストで、自動バックアップの作成に割り当てる時間を選択します。

5. ログのエクスポート ペインで、CloudWatch Logs にエクスポートするログのタイプを選択して完了します。



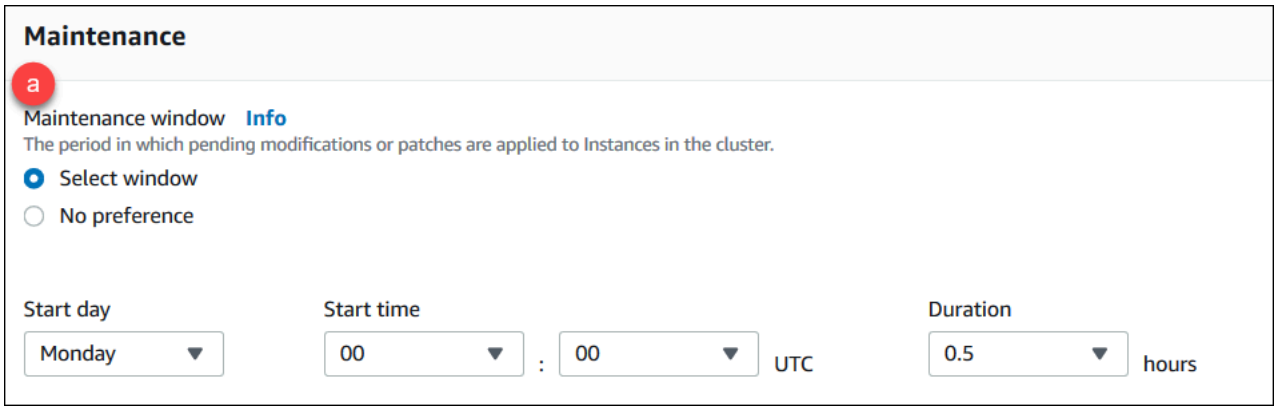
- [Audit logs (監査ログ)] - Amazon CloudWatch Logs にエクスポートするには、このオプションを選択します。[Audit logs (監査ログ)] を選択した場合は、クラスターのカスタムパラメータグループで audit_logs を有効にする必要があります。詳細については、「Amazon DocumentDB イベントの監査」を参照してください。

- [Profiler logs (プロファイラーログ)] - オペレーションのプロファイラーログの Amazon CloudWatch Logs へのエクスポートを有効にするには、このオプションを選択します。[Profiler logs (プロファイラーログ)] を選択した場合は、クラスターのカスタムパラメータグループの次のパラメータも変更する必要があります。

  - `profiler` — `enabled` に設定します。
  - `profiler_threshold_ms` の値に設定して、プロファイリングオペレーションのしきい値を `[0-INT_MAX]` に設定します。
  - `profiler_sampling_rate` プロファイルする低速オペレーションの割合を設定するには、`[0.0-1.0]` の値に設定します。

  詳細については、「Amazon DocumentDB オペレーションのプロファイリング」を参照してください。

6. [メンテナンス] ペインに入力します。



- 以下のいずれかを選択します。

  - [Select window (ウィンドウの選択)] - Amazon DocumentDB によってクラスターでメンテナンスが実行される曜日、UTC 開始時刻、期間を指定できます。

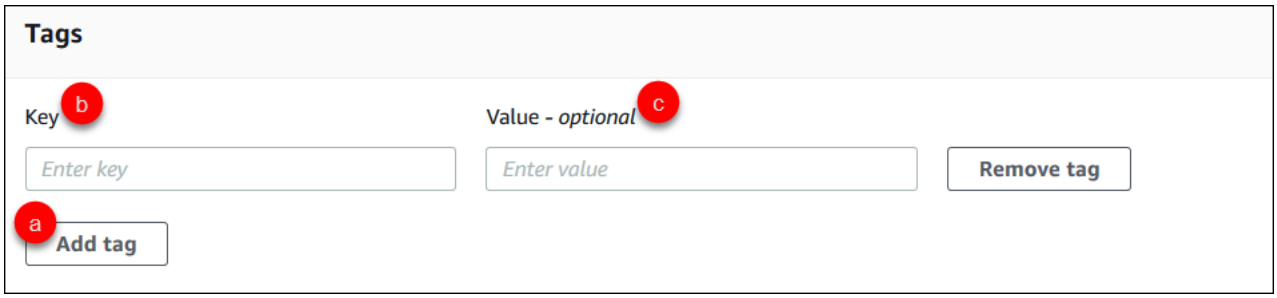
    a. [Start day (開始日)] - リストから、クラスターのメンテナンスを開始する曜日を選択します。

    b. [Start time (開始時刻)] リストで、メンテナンスを開始する時間と分 (UTC) を選択します。

    c. [Duration (期間)] - リストで、クラスターメンテナンスに割り当てる時間を選択します。指定した期間内にメンテナンスが完了できない場合、メンテナンスプロセスは指定した期間を過ぎて終了するまで続行されます。

- [No preference (指定なし)] - Amazon DocumentDB によってメンテナンスが実行される曜日、開始時刻、期間が選択されます。

7. このクラスターに 1 つ以上のタグを追加する場合、[Tags (タグ)] ペインを完了します。



クラスターに追加するタグごとに、以下のステップを繰り返します。クラスターでは最大 10 まで保持できます。

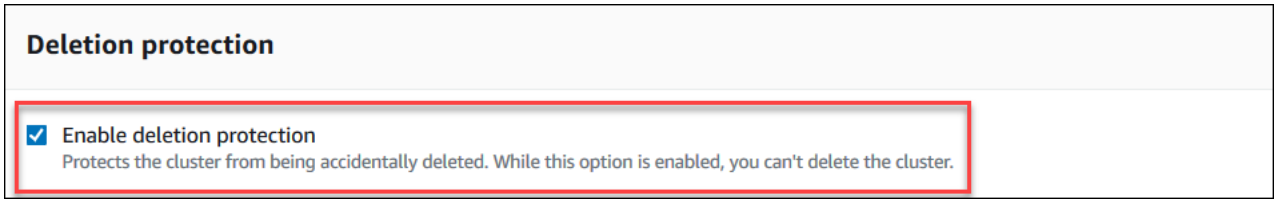
a. [Add tags (タグの追加)] を選択します。

b. タグの [キー] を入力します。

c. 必要に応じてタグの [値] を入力します。

タグを削除するには、[タグの削除] を選択します。

8. コンソールを使用してクラスターを作成する場合、[Deletion Protection (削除に対する保護)] がデフォルトで有効になっています。削除に対する保護を無効化するには、[Enable deletion protection (削除に対する保護の有効化)] のチェックを外します。有効にすると、削除に対する保護はクラスターが削除されることを防ぎます。削除に対して保護されているクラスターを削除するには、まずクラスターに変更を加えて削除に対する保護を無効化する必要があります。



削除に対する保護についての詳細は、「Amazon DocumentDB クラスターの削除」を参照してください。

9. クラスターを作成するには、[Create cluster (クラスターの作成)] を選択します。または、[キャンセル] を選択します。

を使用したクラスターの作成 AWS CLI

次の手順では、 を使用して Amazon DocumentDB クラスターを起動し AWS CLI 、Amazon DocumentDB レプリカを作成する方法について説明します。

パラメータ

- **`--db-cluster-identifier`** — 必須 このクラスターを識別する小文字の文字列。

  クラスターの名前付けの制約:

  - 長さは 1 – 63 個の英字、数字、またはハイフンで構成します。
  - 1 字目は英字である必要があります。
  - ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
  - AWS アカウントごと、リージョンごとに、すべてのクラスター (Amazon RDS、Amazon Neptune、Amazon DocumentDB 全体) で一意である必要があります。

- **`--engine`** — 必須 **`docdb`** を指定してください。
- **`--deletion-protection | --no-deletion-protection`** - オプション 削除に対する保護が有効にされている場合、これによってクラスターが削除されることを防ぎます。を使用する場合 AWS CLI、デフォルト設定では削除保護が無効になっています。

  削除に対する保護についての詳細は、「Amazon DocumentDB クラスターの削除」を参照してください。

- **`--storage-type standard | iopt1`** - オプションです。デフォルト: **`standard`**。クラスターのストレージ設定 有効な値は、`standard` (標準) または `iopt1` (I/O 最適化) です。
- **`--master-username`** — 必須 ユーザーの認証に使用されるユーザー名。

  マスターユーザーの名前付けの制約:

  - 長さは、[1 ~ 63] 個の英数字です。
  - 1 字目は英字である必要があります。
  - データベースエンジンの予約語は使用できません。

- **`--master-user-password`** - オプションです。ユーザーの認証に使用されるユーザーのパスワード。

マスターパスワードの制約:

- 長さは、[8 ~ 100] 個の印刷可能な ASCII 文字です。
- 以下のものを除くすべての印刷可能な ASCII 文字を使用できます。
  - **/** (スラッシュ)
  - **"** (二重引用符)
  - **@** (アットマーク)

- **`--manage-master-user-password`** - オプション。Amazon DocumentDB はマスターユーザーパスワードを生成し、Secrets Manager でライフサイクルを通じて管理します。

その他のパラメータについては、「CreateDBCluster」を参照してください。

を使用して Amazon DocumentDB クラスターを起動するには AWS CLI

Amazon DocumentDB クラスターを作成するには、 を呼び出します`create-db-cluster` AWS CLI。次の AWS CLI コマンドは、削除保護を有効に`sample-cluster`して という名前の Amazon DocumentDB クラスターを作成します。削除に対する保護についての詳細は、「Amazon DocumentDB クラスターの削除」を参照してください。

また、`--engine-version` は、デフォルトで最新のメジャーエンジンバージョンとなるオプションのパラメータです。現在のメジャーエンジンバージョンは 5.0.0 です。新しいメジャーエンジンバージョンがリリースされると、 のデフォルトのエンジンバージョン`--engine-version`が更新され、最後のメジャーエンジンバージョンが反映されます。そのため、本稼働ワークロード、特にスクリプト、オートメーション、または AWS CloudFormation テンプレートに依存するワークロードの場合は、目的のメジャーバージョン`--engine-version`に を明示的に指定することをお勧めします。

> **Note**
>
> もし `db-subnet-group-name` または `vpc-security-group-id` が指定されていない場合、Amazon DocumentDB は、指定されたリージョンに対してデフォルトのサブネットグループと Amazon VPC セキュリティグループを使用します。

Linux、macOS、Unix の場合:

```
aws docdb create-db-cluster \
```

```
      --db-cluster-identifier sample-cluster \
      --engine docdb \
      --engine-version 5.0.0 \
      --deletion-protection \
      --master-username masteruser \
      --master-user-password password
```

Windows の場合:

```
aws docdb create-db-cluster ^
      --db-cluster-identifier sample-cluster ^
      --engine docdb ^
      --engine-version 5.0.0 ^
      --deletion-protection ^
      --master-username masteruser ^
      --master-user-password password
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "DBCluster": {
        "StorageEncrypted": false,
        "DBClusterMembers": [],
        "Engine": "docdb",
        "DeletionProtection" : "enabled",
        "ClusterCreateTime": "2018-11-26T17:15:19.885Z",
        "DBSubnetGroup": "default",
        "EngineVersion": "5.0.0",
        "MasterUsername": "masteruser",
        "BackupRetentionPeriod": 1,
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",
        "DBClusterIdentifier": "sample-cluster",
        "MultiAZ": false,
        "DBClusterParameterGroup": "default.docdb5.0",
        "PreferredBackupWindow": "09:12-09:42",
        "DbClusterResourceId": "cluster-KQSGI4MHU4NTDDRVNLNTU7XVAY",
        "PreferredMaintenanceWindow": "tue:04:17-tue:04:47",
        "Port": 27017,
        "Status": "creating",
        "ReaderEndpoint": "sample-cluster.cluster-ro-sfcrlcjcoroz.us-
east-1.docdb.amazonaws.com",
        "AssociatedRoles": [],
        "HostedZoneId": "ZNKXTT8WH85VW",
```

```
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-77186e0d",
                "Status": "active"
            }
        ],
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1c",
            "us-east-1e"
        ],
        "Endpoint": "sample-cluster.cluster-sfcrlcjcoroz.us-east-1.docdb.amazonaws.com"
    }
}
```

クラスターが作成されるまでに数分かかります。 AWS Management Console または を使用して AWS CLI 、クラスターのステータスをモニタリングできます。詳細については、「Amazon DocumentDB クラスターのステータスのモニタリング」を参照してください。

> **Important**
>
> を使用して Amazon DocumentDB クラスター AWS CLI を作成する場合、インスタンスは作成されません。したがって、プライマリインスタンスと必要なレプリカインスタンスを明示的に作成してください。コンソールまたは を使用してインスタンス AWS CLI を作成できます。詳細については、「クラスターへの Amazon DocumentDB インスタンスの追加」を参照してください。

詳細については、[Amazon DocumentDB API リファレンス の `CreateDBCluster`] を参照してください。

Amazon DocumentDB クラスター の説明

Amazon DocumentDB マネジメントコンソールまたは AWS CLI を使用して、Amazon DocumentDB クラスターに関連する接続エンドポイント、セキュリティグループ、VPCs、パラメータグループなどの詳細を表示できます。

詳細については次を参照してください:

- Amazon DocumentDB クラスターのステータスのモニタリング

- クラスターのエンドポイントの検索

Using the AWS Management Console

次の手順を使用して、コンソールを使用して指定された Amazon DocumentDB クラスターの詳細を表示します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. クラスターの一覧で、詳細を表示するクラスターの名前を選択します。クラスターに関する情報は、次のグループで分けられます。

   - [概要] - エンジンバージョン、クラスターのステータス、保留中のメンテナンス、パラメータグループのステータスなど、クラスターに関する一般情報。
   - [接続とセキュリティ] - [接続] セクションには、mongo シェルやアプリケーションを使用してこのクラスターに接続するための接続エンドポイントが一覧表示されます。[セキュリティグループ] セクションには、このクラスターに関連付けられているセキュリティグループと、その VPC ID や説明が一覧表示されます。
   - [設定] - [クラスターの詳細] セクションには、クラスターの Amazon リソースネーム (ARN)、エンドポイント、パラメータグループなど、クラスターに関する詳細が一覧表示されます。また、クラスターのバックアップ情報、メンテナンスの詳細、セキュリティとネットワークの設定も一覧表示されます。[クラスターのインスタンス] セクションには、このクラスターに属するすべてのインスタンスが一覧表示されます。各インスタンスのロールとクラスターパラメータグループのステータスも一緒に表示されます。
   - [モニタリング] — このクラスターの Amazon CloudWatch Logs メトリクス。詳細については、「Amazon DocumentDB と CloudWatch のモニタリング」を参照してください。
   - [イベントとタグ] — [最近のイベント] セクションには、このクラスターの最近のイベントの一覧が表示されます。Amazon DocumentDB は、クラスター、インスタンス、スナップ

ショット、セキュリティグループ、およびクラスターパラメータグループに関連するイベントのレコードを保持します。この情報には、各イベントに関連する日付、時刻、およびメッセージが含まれます。[タグ] セクションには、このクラスターにアタッチされているタグが一覧表示されます。

Using the AWS CLI

を使用して Amazon DocumentDB クラスターの詳細を表示するには AWS CLI、以下の例に示すように `describe-db-clusters` コマンドを使用します。詳細については、「Amazon DocumentDB リソース管理 API リファレンス」の「`DescribeDBClusters`」を参照してください。

> **Note**
>
> クラスターやインスタンスのライフサイクル管理などの特定の管理機能において、Amazon DocumentDB は Amazon RDS と共有される運用テクノロジーを使用します。`filterName=engine,Values=docdb` フィルターパラメータは Amazon DocumentDB クラスターのみを返します。

Example

例 1: すべての Amazon DocumentDB クラスターを一覧表示する

次の AWS CLI コードは、リージョン内のすべての Amazon DocumentDB クラスターの詳細を一覧表示します。

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBClusters": [
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
```

```
            "DBClusterIdentifier": "sample-cluster-1",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        },
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-2",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        },
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-3",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        }
    ]
}
```

Example

例 2: 指定した Amazon DocumentDB クラスターのすべての詳細を一覧表示する

次の AWS CLI コードは、クラスター の詳細を一覧表示しますsample-cluster。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-clusters \
```

```
   --filter Name=engine,Values=docdb \
   --db-cluster-identifier sample-cluster
```

Windows の場合:

```
aws docdb describe-db-clusters ^
   --filter Name=engine,Values=docdb ^
   --db-cluster-identifier sample-cluster
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBClusters": [
        {
            "AllocatedStorage": 1,
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1a",
                "us-east-1d"
            ],
            "BackupRetentionPeriod": 2,
            "DBClusterIdentifier": "sample-cluster",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            "EarliestRestorableTime": "2023-11-07T22:34:08.148000+00:00",
            "Endpoint": "sample-cluster.node.us-east-1.amazon.com",
            "ReaderEndpoint": "sample-cluster.node.us-east-1.amazon.com",
            "MultiAZ": false,
            "Engine": "docdb",
            "EngineVersion": "5.0.0",
            "LatestRestorableTime": "2023-11-10T07:21:16.772000+00:00",
            "Port": 27017,
            "MasterUsername": "chimeraAdmin",
            "PreferredBackupWindow": "22:22-22:52",
            "PreferredMaintenanceWindow": "sun:03:01-sun:03:31",
            "ReadReplicaIdentifiers": [],
            "DBClusterMembers": [
                {
                    "DBInstanceIdentifier": "sample-instance-1",
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "PromotionTier": 1
```

```
                },
                {
                    "DBInstanceIdentifier": "sample-instance-2",
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "PromotionTier": 1
                },

            ],
            "VpcSecurityGroups": [
                {
                    "VpcSecurityGroupId": "sg-9084c2ec",
                    "Status": "active"
                }
            ],
            "HostedZoneId": "Z06853723JYKYBXTJ49RB",
            "StorageEncrypted": false,
            "DbClusterResourceId": "cluster-T4LGLANHVAPGQYYULWUDKLVQL4",
            "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-
cluster",
            "AssociatedRoles": [],
            "IAMDatabaseAuthenticationEnabled": false,
            "ClusterCreateTime": "2023-11-06T18:05:41.568000+00:00",
            "EngineMode": "provisioned",
            "DeletionProtection": false,
            "HttpEndpointEnabled": false,
            "CopyTagsToSnapshot": false,
            "CrossAccountClone": false,
            "DomainMemberships": [],
            "TagList": [],
            "StorageType": "iopt1",
            "AutoMinorVersionUpgrade": false,
            "NetworkType": "IPV4",
            "IOOptimizedNextAllowedModificationTime":
 "2023-12-07T18:05:41.580000+00:00"
        }
    ]
}
```

Example

例 3: Amazon DocumentDB クラスターの特定の詳細をリストする

を使用してクラスターの詳細のサブセットを一覧表示するには AWS CLI、`describe-db-clusters`オペレーションが一覧表示するクラスターメンバー`--query`を指定する を追加します。`--db-cluster-identifier` パラメータは、詳細を表示する対象のクラスターの識別子です。クエリの詳細については、「で出力をフィルタリングする方法 `--query` オプション の AWS Command Line Interface ユーザーガイド 」を参照してください。

次の例では、リージョン内のすべての Amazon DocumentDB クラスターを一覧表示します。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-clusters \
    --filter Name=engine,Values=docdb \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterMembers]'
```

Windows の場合:

```
aws docdb describe-db-clusters ^
    --filter Name=engine,Values=docdb ^
    --db-cluster-identifier sample-cluster ^
    --query 'DBClusters[*].[DBClusterMembers]'
```

このオペレーションによる出力は、次のようになります。

```
[
    [
        [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-instance-2",
                "IsClusterWriter": false,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ]
    ]
]
```

Amazon DocumentDB クラスターの変更

クラスターを変更するには、クラスターが使用可能な状態である必要があります。停止しているクラスターを変更することはできません。クラスターが停止している場合は、まずクラスターを起動し、クラスターが使用可能になるまで待ってから、必要な変更を加えます。詳細については、「Amazon DocumentDB クラスターの停止と開始」を参照してください。

Using the AWS Management Console

コンソールを使用して特定の Amazon DocumentDB クラスターを変更するには、次の手順を使用します。

Amazon DocumentDB クラスターの変更

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

   > **Tip**
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. クラスター名の左側にあるボタンを選択して、変更するクラスターを指定します。
4. [アクション] を選択してから、[変更] を選択します。
5. [クラスターの変更: <cluster-name>] ペインで、必要な変更を行います。次の分野で変更を行うことができます。

   - クラスターの仕様 — クラスターの名前、セキュリティグループ、認証情報管理。
   - [クラスターストレージ設定] — クラスターのデータストレージモードです。標準設定と I/O 最適化設定のいずれかを選択します。
   - [クラスターオプション] -クラスターのポートおよびパラメータグループ。
   - [バックアップ] - クラスターの保持期間およびバックアップ保持期間。

- [ログのエクスポート] - 監査ログまたはプロファイラーログのエクスポートを有効または無効にします。
- [メンテナンス] - クラスターのメンテナンスウィンドウを設定します。
- [削除保護] - クラスターで削除保護を有効または無効にします。削除保護は、デフォルトで有効になります。

6. 完了したら、[続行] を選択して変更の概要を表示します。
7. 変更に問題がなければ、[クラスターの変更] を選択してクラスターを変更できます。または、[戻る] を選択して変更を編集したり、[キャンセル] を選択して変更をキャンセルしたりすることもできます。

変更が適用されるまでに数分かかります。クラスターを使用できるのは、そのステータスが [available (使用可能)] であるときのみです。コンソールまたは AWS CLIを使用して、クラスターのステータスをモニタリングできます。詳細については、「Amazon DocumentDB クラスターのステータスのモニタリング」を参照してください。

Using the AWS CLI

指定したクラスターを `modify-db-cluster` で変更するには、 AWS CLIオペレーションを使用します。詳細については、[Amazon DocumentDB API リファレンス の `ModifyDBCluster`] を参照してください。

パラメータ

- **`--db-cluster-identifier`** — 必須 変更しようとしている Amazon DocumentDB クラスターの識別子。
- **`--backup-retention-period`** - オプション。自動バックアップを保管する日数。有効な値は、1 – 35 です。
- **`--storage-type`** - オプション。クラスターのストレージ設定 有効な値は、`standard` (標準) または `iopt1` (I/O 最適化) です。
- **`--db-cluster-parameter-group-name`** - オプション。クラスターに使用するクラスターパラメータグループの名前。
- **`--manage-master-user-password`** - オプション。Amazon DocumentDB はマスターユーザーパスワードを生成し、Secrets Manager でライフサイクルを通じて管理します。
- **`--rotate-master-user-password`** - オプションです。Secrets Manager は、既存のシークレットの新しいシークレットバージョンを生成します。シークレットの新しいバージョンには、新しいプライマリユーザーのパスワードが含まれています。Amazon DocumentDB は、新

しいシークレットバージョンのパスワードと一致するようにクラスターのプライマリユーザーパスワードを変更します。

プライマリパスワードを更新するときは、 **--apply-immediately**オプションを指定する必要があります。

- **--master-user-password** - オプションです。プライマリデータベースユーザー向け新しいパスワード。

  パスワードの制約:

  - 長さは 8 ~ 100 バイトの印刷可能な ASCII 文字です。
  - 以下のものを除くすべての印刷可能な ASCII 文字を使用できます。
    - **/** (スラッシュ)
    - **"** (二重引用符)
    - **@** (アットマーク)
- **--new-db-cluster-identifier** - オプション。クラスターの名前を変更する場合のクラスターの新しいクラスター識別子。この値は小文字で保存されます。

  命名に関する制約:

  - 1 – 63 個の英字、数字、またはハイフンで構成します。
  - 1 字目は英字である必要があります。
  - ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
  - Amazon RDS、Amazon Neptune、Amazon DocumentDB のすべてのクラスターで AWS アカウント、リージョンごとに一意である必要があります。
- **--preferred-backup-window** - オプション。自動バックアップが作成される毎日の時間帯 (世界協定時刻 (UTC))。
  - 形式: `hh24:mm-hh24:mm`
- **--preferred-maintenance-window** - オプション。週 1 回のシステムメンテナンスを実行できる時間帯 (UTC)。
  - 形式: `ddd:hh24:mm-ddd:hh24:mm`
  - 有効な日: `Sun`、`Mon`、`Tue`、`Wed`、`Thu`、`Fri`、および `Sat`。
- **--deletion-protection** または **--no-deletion-protection** - オプション。このクラスターで削除に対する保護を有効にするかを指定します。削除保護では、クラスターを変更して削除保護を無効にするまで、クラスターが誤って削除されないようにします。詳細については、「Amazon DocumentDB クラスターの削除」を参照してください。

- **--apply-immediately** または **--no-apply-immediately** – 今すぐ変更を適用するには、--apply-immediately を使用します。クラスターの次のメンテナンスウィンドウ時に変更するには、--no-apply-immediately を使用します。

Example

次のコードは、クラスター sample-cluster のバックアップ保持期間を変更します。

Linux、macOS、Unix の場合:

```
aws docdb modify-db-cluster \
       --db-cluster-identifier sample-cluster \
       --apply-immediately \
       --backup-retention-period 7
```

Windows の場合:

```
aws docdb modify-db-cluster ^
       --db-cluster-identifier sample-cluster ^
       --apply-immediately ^
       --backup-retention-period 7
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBCluster": {
        "BackupRetentionPeriod": 7,
        "DbClusterResourceId": "cluster-VDP53QEWST7YHM36TTXOPJT5YE",
        "Status": "available",
        "DBClusterMembers": [
            {
                "PromotionTier": 1,
                "DBClusterParameterGroupStatus": "in-sync",
                "DBInstanceIdentifier": "sample-cluster-instance",
                "IsClusterWriter": true
            }
        ],
        "ReadReplicaIdentifiers": [],
        "AvailabilityZones": [
            "us-east-1b",
            "us-east-1c",
            "us-east-1a"
```

```
        ],
        "ReaderEndpoint": "sample-cluster.cluster-ro-ctevjxdlur57.us-
east-1.rds.amazonaws.com",
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",
        "PreferredMaintenanceWindow": "sat:09:51-sat:10:21",
        "EarliestRestorableTime": "2018-06-17T00:06:19.374Z",
        "StorageEncrypted": false,
        "MultiAZ": false,
        "AssociatedRoles": [],
        "MasterUsername": "<your-master-user-name>",
        "DBClusterIdentifier": "sample-cluster",
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-77186e0d"
            }
        ],
        "HostedZoneId": "Z2SUY0A1719RZT",
        "LatestRestorableTime": "2018-06-18T21:17:05.737Z",
        "AllocatedStorage": 1,
        "Port": 27017,
        "Engine": "docdb",
        "DBClusterParameterGroup": "default.docdb3.4",
        "Endpoint": "sample-cluster.cluster-ctevjxdlur57.us-
east-1.rds.amazonaws.com",
        "DBSubnetGroup": "default",
        "PreferredBackupWindow": "00:00-00:30",
        "EngineVersion": "3.4",
        "ClusterCreateTime": "2018-06-06T19:25:47.991Z",
        "IAMDatabaseAuthenticationEnabled": false
    }
}
```

変更が適用されるまでに数分かかります。クラスターを使用できるのは、そのステータスが [available (使用可能)] であるときのみです。コンソールまたは AWS CLIを使用して、クラスターのステータスをモニタリングできます。詳細については、「Amazon DocumentDB クラスターのステータスのモニタリング」を参照してください。

実施待ちのメンテナンスの確認

Amazon DocumentDB エンジンのバージョンが最新であるかどうかを確認するには、保留中のクラスターメンテナンスがあるかどうかを確認します。

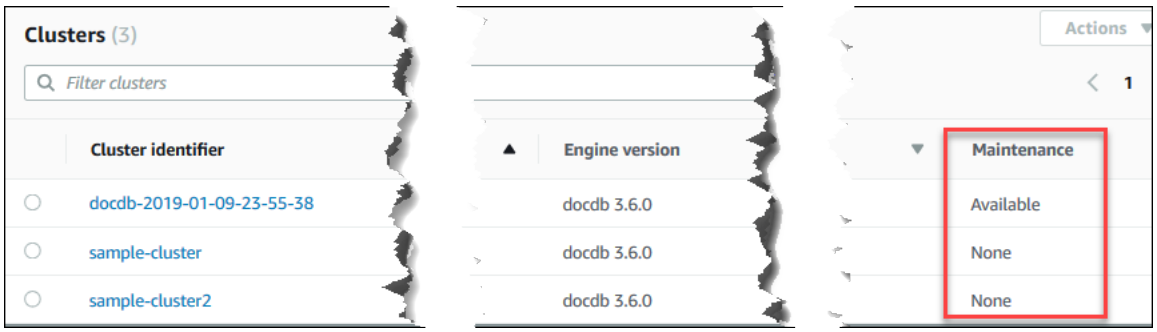
Using the AWS Management Console

を使用して AWS Management Console 、クラスターに保留中のメンテナンスがあるかどうかを判断できます。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. [メンテナンス] 列を探して、クラスターに保留中のメンテナンスがあるかどうかを確認します。



[なし] は、クラスターが最新のエンジンバージョンを実行していることを示します。[利用可能] は、クラスターに保留中のメンテナンスがあることを示し、エンジンのアップグレードが必要である可能性があります。

4. クラスターに保留中のメンテナンスがある場合は、クラスターのエンジンバージョンへのパッチ更新の実行 のステップに進みます。

Using the AWS CLI

を使用して、クラスターに最新のエンジンバージョンがあるかどうか AWS CLI を確認するには、次のパラメータを指定して `describe-pending-maintenance-actions`オペレーションを使用します。

パラメータ

- **--resource-identifier** - オプション。リソースの ARN (クラスター)。このパラメータを省略すると、すべてのクラスターの保留中のメンテナンスアクションがリストされます。
- **--region** - オプション。このオペレーションを実行する AWS リージョン (us-east-1 など)。

Example

Linux、macOS、Unix の場合:

```
aws docdb describe-pending-maintenance-actions \
   --resource-identifier arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster \
   --region us-east-1
```

Windows の場合:

```
aws docdb describe-pending-maintenance-actions ^
   --resource-identifier arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster ^
   --region us-east-1
```

このオペレーションによる出力は、次のようになります。

```
{
    "PendingMaintenanceActions": [
        {
            "ResourceIdentifier": "arn:aws:rds:us-
east-1:123456789012:cluster:sample-cluster",
            "PendingMaintenanceActionDetails": [
                {
                    "Description": "New feature",
                    "Action": "db-upgrade",
                    "ForcedApplyDate": "2019-02-25T21:46:00Z",
                    "AutoAppliedAfterDate": "2019-02-25T07:41:00Z",
                    "CurrentApplyDate": "2019-02-25T07:41:00Z"
                }
            ]
        }
    ]
```

```
}
```

クラスターに保留中のメンテナンスがある場合は、クラスターのエンジンバージョンへのパッチ更新の実行 のステップに進みます。

クラスターのエンジンバージョンへのパッチ更新の実行

このセクションでは、 AWS Management Console または を使用してパッチ更新をデプロイする方法について説明します AWS CLI。パッチ更新とは、同じエンジンバージョン内のアップデートです (たとえば、3.6 エンジンバージョンを新しい 3.6 エンジンバージョンにアップデートするなど)。すぐにアップグレードするか、クラスターの次回のメンテナンス期間中にアップデートできます。エンジンのアップグレードが必要かどうかを確認するには、実施待ちのメンテナンスの確認 を参照してください。更新を適用すると、クラスターでダウンタイムが生じることを理解しておいてください。

> **Note**
>
> 3.6 から 5.0 など、メジャーエンジンバージョンから別のエンジンバージョンにアップグレードする場合は、「Amazon DocumentDB のインプレースメジャーバージョンアップグレード」または「を使用した Amazon DocumentDB クラスターのアップグレード AWS Database Migration Service」を参照してください。メジャーバージョン一括アップグレードでは、ターゲットエンジンバージョンとして docdb 5.0 のみがサポートされます。

クラスターのエンジンバージョンの最新のパッチアップデートを取得するには、次の 2 つの構成要件があります。

- クラスターステータスは利用可能でなければなりません。
- クラスターが以前のエンジンバージョンで実行されている必要があります。

Using the AWS Management Console

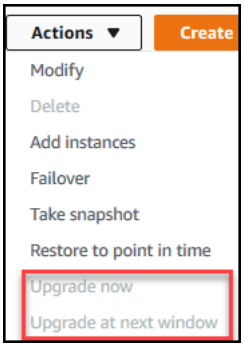
次の手順では、コンソールを使用して、パッチを適用し、クラスターのエンジンバージョンを最新のバージョンに更新します。すぐにアップグレードするか、クラスターの次のメンテナンスウィンドウ時に更新できます。

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

2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。クラスターのリストで、アップグレードするクラスターの左側にあるボタンを選択します。クラスターのステータスは利用可能でなければなりません。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. [アクション] メニューから、以下のいずれかのオプションを選択します。これらのメニューオプションは、選択したクラスターで最新のエンジンバージョンが実行されていない場合にのみ選択できます。



- [今すぐアップグレード] – 今すぐアップグレードプロセスが開始されます。最新のエンジンバージョンにアップグレードしている間、クラスターはしばらくオフラインになります。
- [次のウィンドウでアップグレード] - クラスターの次のメンテナンスウィンドウ時にアップグレードを開始します。クラスターは、最新のエンジンバージョンにアップグレードしている間、しばらくオフラインになります。

4. 確認ウィンドウが開いたら、次のいずれかを選択します。
   - [アップグレード] - 前のステップで選択したスケジュールに基づき、クラスターを最新のエンジンバージョンにアップグレードします。
   - [キャンセル] - クラスターのエンジンアップグレードをキャンセルし、クラスターの最新エンジンバージョンを引き続き使用します。

Using the AWS CLI

クラスターにパッチ更新を適用するには、 AWS CLI と apply-pending-maintenance-actionオペレーションを使用して、次のパラメータを指定します。

パラメータ

- **--resource-identifier** — 必須 アップグレードしようとしている Amazon DocumentDB クラスターの ARN。
- **--apply-action** — 必須 以下の値を指定できます。クラスターのエンジンバージョンをアップグレードするには、db-upgrade を使用します。
  - **db-upgrade**
  - **system-update**
- **--opt-in-type** — 必須 以下の値を指定できます。
  - immediate -メンテナンスアクションをすぐに適用します。
  - next-maintenance - 次回のメンテナンス期間中にメンテナンスアクションを適用します。
  - undo-opt-in -既存の next-maintenance オプトインリクエストをキャンセルします。

Example

次の例では、sample-cluster のエンジンバージョンをエンジン 4.0.0 にパッチを更新します。

Linux、macOS、Unix の場合:

```
aws docdb apply-pending-maintenance-action \
   --resource-identifier arn:aws:rds:us-east-1:123456789012\:cluster:sample-cluster
 \
   --apply-action db-upgrade \
   --opt-in-type immediate
```

Windows の場合:

```
aws docdb apply-pending-maintenance-action ^
   --resource-identifier arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster ^
   --apply-action db-upgrade ^
   --opt-in-type immediate
```

このオペレーションの出力は、次のようになります。

```
{
    "ResourcePendingMaintenanceActions": {
        "ResourceIdentifier": "arn:aws:rds:us-
east-1:444455556666:cluster:docdb-2019-01-09-23-55-38",
        "PendingMaintenanceActionDetails": [
            {
                "CurrentApplyDate": "2019-02-20T20:57:06.904Z",
                "Description": "Bug fixes",
                "ForcedApplyDate": "2019-02-25T21:46:00Z",
                "OptInStatus": "immediate",
                "Action": "db-upgrade",
                "AutoAppliedAfterDate": "2019-02-25T07:41:00Z"
            }
        ]
    }
}
```

Amazon DocumentDB クラスターの停止と開始

Amazon DocumentDB クラスターの停止と開始は、開発とテスト環境のコストを管理するために便利です。Amazon DocumentDB を使用するたびにクラスターとインスタンスを作成したり削除する代わりに、インスタンスが必要ではないときにクラスター内ですべてのインスタンスを一時的に停止できます。テストを再開するときに、このインスタンスを再開できます。

トピック

- クラスターの停止と開始の概要
- 停止時のクラスターのみで実行できるオペレーション

クラスターの停止と開始の概要

Amazon DocumentDB クラスターが必要ではない期間は、そのクラスターですべてのインスタンスを一度に停止することができます。そして、クラスターを使用する必要がある時はいつでもクラスターを開始できます。開始と停止は、継続的な可用性を必要としない開発、テスト、または類似のアクティビティに使用されるクラスターのセットアップと解放のプロセスを簡素化します。クラスター内のインスタンスの数に関係なく、 AWS Management Console または を使用して 1 つのアクション AWS CLI でクラスターを停止および開始できます。

クラスターが停止されている間、このクラスターのストレージボリュームは変更されません。指定された保持期間内でのストレージ、手動のスナップショット、および自動化されたバックアップストレージに対してのみ課金されます。インスタンス時間に対しては請求されません。Amazon DocumentDB は、7 日後に DB クラスターを自動的に開始するため、必要なメンテナンスの更新が遅延することはありません。7 日後にクラスターが開始すると、このクラスター内のインスタンスに対する課金が再開します。クラスターが停止している間は、ストレージボリュームに対するクエリを実行できません。クエリを実行するにはインスタンスが使用可能状態であることが必要です。

Amazon DocumentDB クラスターが停止されている場合、クラスターもそのインスタンスも、いかなる方法でも変更できません。これには、インスタンスの追加や削除、クラスターの削除が含まれます。

Using the AWS Management Console

次の手順では、利用可能な 1 つ以上のインスタンスがあるクラスターを停止したり、停止しているクラスターを開始する方法を示します。

Amazon DocumentDB クラスターを停止または開始するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

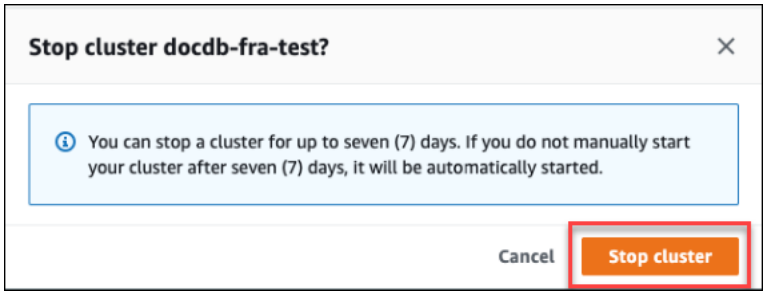
   > **Tip**
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. クラスターのリストで、停止または開始するクラスター名の左側にあるボタンを選択します。
4. [Actions (アクション)] を選択したら、このクラスターで実行するアクションを選択します。
   - クラスターを停止したいときに、このクラスターが利用可能な場合:
     a. [停止] を選択します。

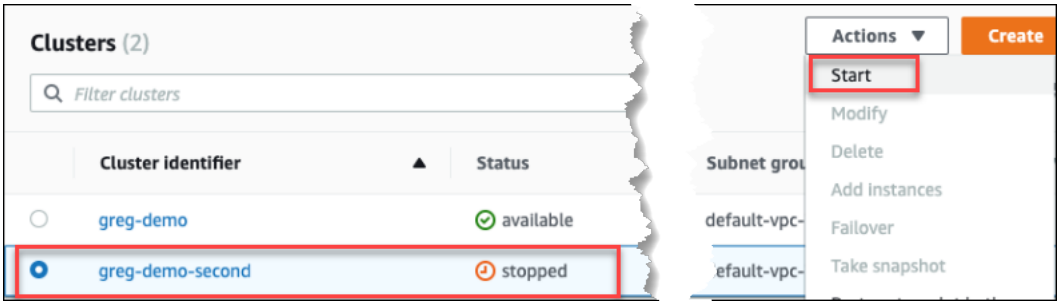


フェイルオーバーメカニズムが発動することを回避するため、停止オペレーションはまずレプリカインスタンスを停止してからプライマリインスタンスを停止します。

b. 確認ダイアログで、クラスターを停止するには [Stop cluster (クラスターの停止)]、あるいはクラスターを継続して実行するには [Cancel (キャンセル)] を選択して確認します。



- クラスターを開始したいときにこのクラスターが停止されている場合、[Start (開始)] を選択します。



5. クラスターおよびそのインスタンスの状態をモニタリングします。クラスターを開始すると、クラスターおよびそのインスタンスが利用可能な場合にこのクラスターの使用を再開できます。詳細については、「クラスターのステータスの評価」を参照してください。



Using the AWS CLI

次のコード例では、利用可能な状態にある 1 つ以上のインスタンスがあるクラスターを停止あるいは停止しているクラスターを開始する方法を示しています。

を使用して 1 つ以上の使用可能なインスタンスを持つクラスターを停止するには AWS CLI、`stop-db-cluster`オペレーションを使用します。停止しているクラスターを開始するには、 `start-db-cluster` オペレーションを使用します。どちらのオペレーションも `--db-cluster-identifier` パラメータを使用します。

パラメータ:

- **`--db-cluster-identifier`** — 必須 停止または開始するクラスターの名前。

Example — を使用してクラスターを停止するには AWS CLI

次のコードは、クラスター `sample-cluster` を停止します。クラスターには、利用可能な状態にある 1 つ以上のインスタンスがある必要があります。

Linux、macOS、Unix の場合:

```
aws docdb stop-db-cluster \
   --db-cluster-identifier sample-cluster
```

Windows の場合:

```
aws docdb stop-db-cluster ^
   --db-cluster-identifier sample-cluster
```

Example — を使用してクラスターを開始するには AWS CLI

次のコードは、クラスター sample-cluster を開始します。クラスターは現在停止している必要があります。

Linux、macOS、Unix の場合:

```
aws docdb start-db-cluster \
   --db-cluster-identifier sample-cluster
```

Windows の場合:

```
aws docdb start-db-cluster ^
   --db-cluster-identifier sample-cluster
```

停止時のクラスターのみで実行できるオペレーション

Amazon DocumentDB クラスター停止中に、指定された自動バックアップ保持期間内であれば、ポイントインタイムの復元を実行できます。ポイントインタイムの復元の実行の詳細については、「特定時点への復元」を参照してください。

クラスターを停止している間は、Amazon DocumentDB クラスターまたはその インスタンスの設定を変更することはできません。クラスターにインスタンスを追加または削除することも、関連するインスタンスがまだある場合はクラスターを削除することもできません。そのような管理アクションを実行する前に、クラスターを開始する必要があります。

Amazon DocumentDB は、スケジュールされたメンテナンスを、再び開始してからのみ停止したクラスターに適用します。7 日後、Amazon DocumentDB は停止したクラスターを自動的に開始するため、そのメンテナンス状態より過剰に遅れることはありません。クラスターが再開始すると、クラスター内のインスタンスへの課金が再開します。

クラスターが停止している間、Amazon DocumentDB は自動バックアップを一切実行せず、またバックアップ保持期間を延長しません。

Amazon DocumentDB クラスターの削除

Amazon DocumentDB クラスターは、 AWS Management Console または を使用して削除できます AWS CLI。クラスターを削除するには、クラスターが 使用可能状態であり、インスタンスが関連付けられていないことが必要です。クラスターが停止している場合は、まずクラスターを起動し、ク

ラスターが使用可能になるまで待ってから、クラスターを削除します。詳細については、「Amazon DocumentDB クラスターの停止と開始」を参照してください。

削除保護

クラスターが誤って削除されることを防ぐために、削除保護を有効にできます。コンソールを使用してクラスターを作成する場合は、デフォルトで削除保護が有効になっています。ただし、 AWS CLI を使用してクラスターを作成する場合、削除保護はデフォルトで無効になっています。

Amazon DocumentDB は、コンソールまたは AWS CLIを使用した削除オペレーションのどちらの実行でも、クラスターに対する削除保護を適用します。削除保護が有効になっている場合、クラスターを削除することはできません。削除に対する保護が有効化されているクラスターを削除するには、まずクラスターを変更して削除保護を無効化する必要があります。

クラスター上で削除保護を有効にしたコンソールを使用する場合、クラスターの最後のインスタンスを削除することはできません。これを削除するとクラスターも削除されるためです。 AWS CLIを使用すると、削除保護されているクラスターの最後のインスタンスを削除できます。ただし、クラスター自体は削除されず、データを保持されます。クラスターに新しいインスタンスを作成して、このデータにアクセスできます。削除保護の有効化および無効化の詳細については、以下を参照してください。

- Amazon DocumentDB クラスターの作成
- Amazon DocumentDB クラスターの変更

Using the AWS Management Console

を使用してクラスターを削除するには AWS Management Console、削除保護を無効にする必要があります。

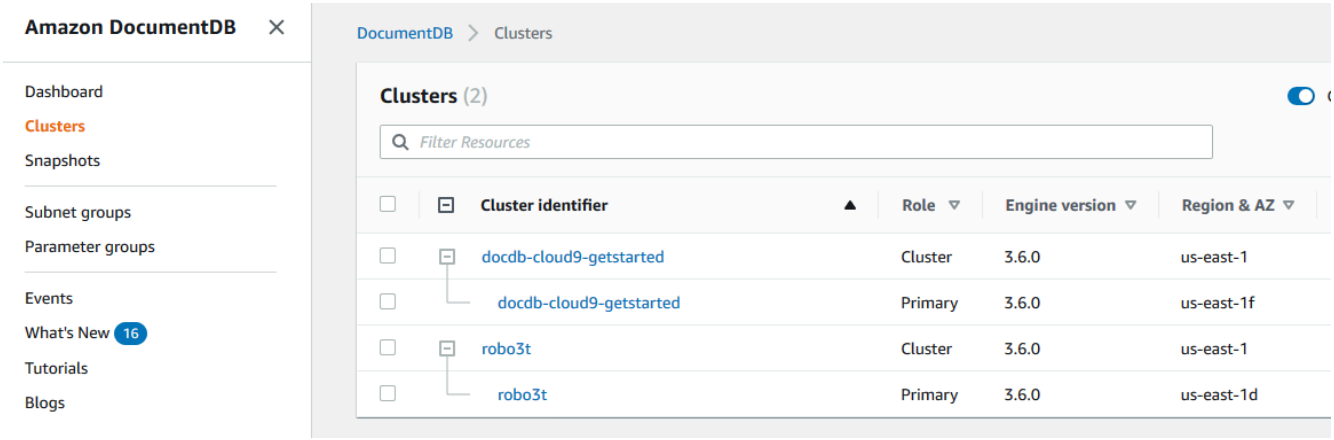
クラスターで削除保護が有効になっているかどうかを確認するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン

(≡)
を選択します。

3. クラスターナビゲーションボックスでは、クラスター識別子列にクラスターとインスタンスの両方が表示されます。インスタンスは、以下のスクリーンショットのように、クラスターの下に表示されます。



4. クラスターの名前を選択し、[設定] タブを選択します。[クラスターの詳細] セクションで、[削除保護] を見つけます。削除保護が有効化されている場合は、クラスターを変更して削除保護を無効にします。クラスターの変更については、「Amazon DocumentDB クラスターの変更」を参照してください。

   [Deletion protection (削除保護)] が無効化されると、クラスターを削除する準備が完了します。

クラスターを削除するには

1. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
2. クラスターの下にリストされているインスタンスのクラスター識別子列をチェックして、クラスターにインスタンスがあるかどうかを判断します。クラスターを削除する前に、そのすべてのインスタンスを削除する必要があります。詳細については、「Amazon DocumentDB インスタンスの削除」を参照してください。
3. クラスターにインスタンスがあるかどうかに応じて、次のいずれかの手順を実行します。
   - クラスターにインスタンスがない場合は、クラスター名の左側にあるボタンを選択し、[アクション] を選択します。ドロップダウンメニューから [削除] を選択します。[Delete <cluster-name> (<cluster-name> を削除)] ダイアログボックスを完了したら、[Delete (削除)] を選択します。

- クラスターに 1 つ以上のインスタンスがある場合は、次を行います。

  a. ナビゲーションペインで [Clusters] (クラスター) を選択してください。

  b. クラスターの名前の左側にあるチェックボックスをオンにして、クラスターの各インスタンスを削除します。[Actions (アクション)] を選択して、[Delete (削除)] を選択します。[Delete <cluster-name> (<cluster-name> を削除)] ダイアログボックスを完了したら、[Delete (削除)] を選択します。

     最後のインスタンスを削除すると、クラスターも削除されます。インスタンスの削除の詳細については、「」を参照してくださいAmazon DocumentDB インスタンスの削除。

クラスターが削除されるまでに数分かかります。クラスターのステータスをモニタリングするには、「Amazon DocumentDB クラスターのステータスのモニタリング」を参照してください。

Using the AWS CLI

インスタンスが関連付けられているクラスターを削除することはできません。クラスターに関連付けられているインスタンスを確認するには、`describe-db-clusters` コマンドを実行し、クラスターのすべてのインスタンスを削除します。次に、必要に応じてクラスターの削除保護を無効にし、最後にクラスターを削除します。

1. 最初に、クラスターのすべてのインスタンスを削除します。

   削除する必要があるインスタンスを決定するには、次のコマンドを実行します。

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].
[DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
```

   このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
[
    [
        "sample-cluster",
        [
            "sample-instance-1",
            "sample-instance-2"
        ]
```

```
    ]
]
```

削除するクラスターにインスタンスがある場合は、次に示すように、そのインスタンスを削除します。

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-instance
```

2. 次に、削除保護を無効にします。

AWS CLI を使用してクラスターのすべてのインスタンスを削除しても、クラスターは削除されません。クラスターも削除する必要があるときには、削除保護が無効化されている場合にのみこれを行うことができます。

クラスターに削除保護が有効化されているかどうかを判断するには、次のコマンドを実行します。

> **Tip**
>
> すべての Amazon DocumentDB クラスターの削除保護状態を確認するには、`--db-cluster-identifier` パラメータを省略します。

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DeletionProtection]'
```

このオペレーションによる出力は、次のようになります。

```
[
    [
        "sample-cluster",
        "true"
    ]
]
```

削除保護が有効になっている場合、クラスターを変更して削除保護を無効にします。クラスターの削除保護を無効にするには、次のコマンドを実行します。

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --no-deletion-protection \
    --apply-immediately
```

3. 最後に、クラスターを削除します。

削除保護を無効にしたら、クラスターを削除する準備が整います。クラスターを削除するには、以下のパラメータを指定して delete-db-cluster オペレーションを使用します。

- **--db-cluster-identifier** — 必須 削除するクラスターの識別子。
- **--final-db-snapshot-identifier** - オプション。最終スナップショットを作成する場合は、最終スナップショットの名前を指定して、このパラメータを含める必要があります。--final-db-snapshot-identifier または --skip-final-snapshot のどちらかを含める必要があります。

  命名に関する制約:

  - 1 – 63 個の英字、数字、またはハイフンで構成します。
  - 1 字目は英字である必要があります。
  - ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
  - Amazon RDS、Amazon Neptune、Amazon DocumentDB のすべてのクラスターで AWS アカウント、リージョンごとに一意である必要があります。
- **--skip-final-snapshot** - オプション。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しない場合のみ、このパラメータを使用します。デフォルトでは、最終スナップショットを作成します。--final-db-snapshot-identifier または --skip-final-snapshot のどちらかを含める必要があります。

次の AWS CLI コードは、最終スナップショットsample-clusterを持つクラスターを削除します。クラスターに関連付けられるインスタンスがある場合、あるいは削除保護が有効化されている場合、このオペレーションは失敗します。

Example

Linux、macOS、Unix の場合:

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --final-db-snapshot-identifier sample-cluster-final-snapshot
```

Windows の場合:

```
aws docdb delete-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --final-db-snapshot-identifier sample-cluster-final-snapshot
```

Example

次の AWS CLI コードは、最終スナップショットを作成sample-clusterせずにクラスターを削除します。

Linux、macOS、Unix の場合:

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --skip-final-snapshot
```

Windows の場合:

```
aws docdb delete-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --skip-final-snapshot
```

delete-db-cluster オペレーションの出力は、削除するクラスターになります。

クラスターが削除されるまでに数分かかります。クラスターのステータスをモニタリングするには、「クラスターのステータスのモニタリング」を参照してください。

Amazon DocumentDB クラスターのスケーリング

Amazon DocumentDB では、ニーズに応じてクラスターのストレージとコンピューティングをスケールできます。このセクションでは、ストレージのスケーリング、インスタンスのスケーリング、および読み取りのスケーリングを使用して、Amazon DocumentDB クラスターとインスタンスのパフォーマンスとスケーリングを管理する方法について説明します。

トピック

- ストレージのスケーリング
- インスタンスのスケーリング
- 読み取りのスケーリング
- 書き込みスケーリング

ストレージのスケーリング

Amazon DocumentDB ストレージは、クラスターボリューム内のデータに合わせて自動的にスケーリングします。データが増加すると、クラスターボリュームストレージは、10 GiB 単位で最大 128 TiB まで増加します。

インスタンスのスケーリング

DB クラスター内の各 DB インスタンスの DB インスタンスクラスを変更することで、必要に応じて Amazon DocumentDB クラスターをスケーリングできます。Amazon DocumentDB では、Amazon DocumentDB 用に最適化されたいくつかのインスタンスクラスをサポートしています。

詳細については、「Amazon DocumentDB インスタンスの変更」を参照してください。

読み取りのスケーリング

Amazon DocumentDB クラスターの読み取りのスケーリングは、最大 15 個の Amazon DocumentDB レプリカをクラスター内に作成することで実現できます。各 Amazon DocumentDB レプリカは、最小限のレプリカラグでクラスターボリュームから同じデータを返します。通常、このラグはプライマリインスタンスが更新を書き込んだ後、100 ミリ秒を大幅に下回ります。読み取りトラフィックが増えたら、追加の Amazon DocumentDB レプリカを作成し、それらに直接接続することで DB クラスターの読み取りワークロードを分散できます。Amazon DocumentDB レプリカのインスタンスクラスは、プライマリイスタンスと同じものである必要はありません。

詳細については、「クラスターへの Amazon DocumentDB インスタンスの追加」を参照してください。

Amazon DocumentDB でスケールを読み取る場合は、レプリカセットとしてクラスターに接続し、ドライバーに組み込まれている読み取り設定機能を使用してレプリカインスタンスに読み取りを分散することをお勧めします。詳細については、「レプリカセットとしての Amazon DocumentDB に接続する」を参照してください。

書き込みスケーリング

クラスターのプライマリインスタンスのサイズを増やすことで、Amazon DocumentDB クラスターの書き込み容量をスケールできます。このセクションでは、ニーズに応じてクラスターのプライマリインスタンスをスケーリングする 2 つの方法を示します。最初のオプションは、アプリケーションへの影響を最小限に抑えることを目指していますが、完了するにはより多くのステップが必要です。2 番目のオプションは、ステップ数が減るため、最適化で簡素化されますが、アプリケーションへの潜在的な影響が増えるというトレードオフを伴います。

アプリケーションに応じて、以下の最適なアプローチを選択できます。利用可能なインスタンスのサイズとコストの詳細については、「Amazon DocumentDB の料金」を参照してください。

1. 高可用性とパフォーマンスのための最適化 - レプリカセットモード でクラスターに接続している場合 (推奨) は、プライマリインスタンスのスケーリング時に、次のプロセスを使用してアプリケーションへの影響を最小限に抑えることができます。この方法では、クラスターが高可用性以上に維持され、読み取りスケーリングターゲットが (インプレースで更新されずに) インスタンスとしてクラスターに追加されるため、影響が最小限に抑えられます。

   a. より大きなインスタンスタイプの 1 つ以上のレプリカをクラスターに追加します (「???」を参照)。すべてのレプリカは、プライマリと同等以上のサイズのインスタンスタイプにすることをお勧めします。これにより、書き込みパフォーマンスが意図せずに低下して、より小さなインスタンスタイプにフェイルオーバーすることを回避できます。ほとんどのお客様にとって、これは、クラスター内のインスタンス数を一時的に 2 倍にし、スケーリングの完了後により小さなレプリカを削除することを意味します。

   b. すべての新しいレプリカのフェイルオーバー層を優先度ゼロに設定し、より小さいインスタンスタイプのレプリカのフェイルオーバー優先度が最も高くなるようにします。詳細については、「???」を参照してください。

   c. 手動フェイルオーバーを開始し、新しいレプリカの 1 つをプライマリインスタンスに昇格させます。詳細については、「???」を参照してください。

> **Note**
>
> これに伴って、クラスターのダウンタイムが最大 30 秒発生します。このダウンタイムを見越した計画を立ててください。

d. 新しいプライマリより小さいインスタンスタイプのすべてのレプリカをクラスターから削除します。

e. すべてのインスタンスのフェイルオーバー層の設定を同じ優先度に戻します (通常、これは設定を 1 に戻すことを意味します)。

たとえば、クラスターに現在 3 つの `r5.large` インスタンス (1 つのプライマリと 2 つのレプリカ) が含まれていて、`r5.xlarge` インスタンスタイプにスケールするとします。これを行うには、まず 3 つの `r5.xlarge` レプリカインスタンスをクラスターに追加し、これらの新しい `r5.xlarge` レプリカのフェイルオーバー層をゼロに設定します。次に、手動フェイルオーバーを開始します (アプリケーションのダウンタイムを最大 30秒見込みます)。フェイルオーバーが完了したら、3 つすべての `r5.large` インスタンスをクラスターから削除し、クラスターが `r5.xlarge` インスタンスにスケールされた状態にします。

コストを最適化しやすいように、Amazon DocumentDB インスタンスは秒単位で課金されます。インスタンスの作成、変更、削除などの課金対象のステータス変更に続く 10 分間の料金が最小の請求額となります。詳細については、ベストプラクティスのドキュメントの「コスト最適化」を参照してください。

2. 単純化のために最適化 — このアプローチは、シンプルさを最適化します。クラスターの拡張と縮小は行われませんが、読み取り容量が一時的に減少する可能性があります。

   レプリカのインスタンスクラスを変更すると、そのインスタンスが数秒から 30 秒未満の短い期間、リクエストを処理しなくなる可能性があります。レプリカセットモード(推奨) でクラスターに接続している場合、スケーリング操作中に読み取りキャパシティが 1 つのレプリカ (例えば、3 ノードクラスタでは 66% の容量、4 ノードクラスタで 75% の容量など) が減少します。

   a. クラスター内の 1 つのレプリカインスタンスをスケールします。詳細については、「インスタンスクラスの管理」を参照してください。

   b. インスタンスが利用可能になるまで待ちます (「Amazon DocumentDB インスタンスのステータスのモニタリング」を参照)。

> **Note**
>
> これに伴って、クラスターのダウンタイムが最大 30 秒発生します。このダウンタイムを見越した計画を立ててください。

c. すべてのレプリカインスタンスが 1 つずつスケールされるまで、手順 1 と 2 を実行します。

d. 手動フェイルオーバーを開始します。これにより、レプリカがプライマリインスタンスに昇格します。詳細については、「Amazon DocumentDB フェイルオーバー」を参照してください。

> **Note**
>
> これにより、クラスターのダウンタイムが最大 30 秒発生しますが、それよりも時間が短くなります。このダウンタイムを見越した計画を立ててください。

e. 前のプライマリ (現在はレプリカ) インスタンスをスケールします。

Amazon DocumentDB クラスターのボリュームのクローン作成

Amazon DocumentDB クローン作成を使用すると、同じ Amazon DocumentDB クラスターボリュームを使用して、元のクラスターと同じデータを持つ新しいクラスターを作成できます。このプロセスは、高速で、費用効果が高いように設計されています。関連付けられたデータボリュームを持つ新しいクラスターは、クローンと呼ばれます。クローンの作成は、スナップショットの復元など、他の手法を使用してデータを物理的にコピーするよりも、高速かつスペース効率に優れています。

Amazon DocumentDB は、プロビジョニングされた Amazon DocumentDB クラスターからの Amazon DocumentDB プロビジョニングクローンの作成をサポートしています。作成元とは異なるデプロイ設定を使用してクローンを作成すると、作成元の Amazon DocumentDB エンジンの最新バージョンを使用してクローンが作成されます。

Amazon DocumentDB クラスターからクローンを作成すると、ソース Amazon DocumentDB クラスターを所有するのと同じアカウントである AWS アカウントにクローンが作成されます。

トピック

- Amazon DocumentDB のクローン作成の概要
- Amazon DocumentDB クローン作成の制約事項

- Amazon DocumentDB のクローン作成の仕組み
- Amazon DocumentDB のクローンの作成

Amazon DocumentDB のクローン作成の概要

Amazon DocumentDB では、クローン作成に、コピーオンライトプロトコルが使用されます。このメカニズムでは、初期クローンを作成するために使用する追加領域は最小限です。クローンが初期に作成されると、Amazon DocumentDB は、作成元 DB クラスターと新しい (クローンの) Amazon DocumentDB クラスターで使用されるデータのコピーを 1 つだけ保持します。追加のストレージは、作成元の Amazon DocumentDB クラスターまたは Amazon DocumentDB クラスターのクローンが (Amazon DocumentDB ストレージボリューム上の) データに変更を加えた場合にのみ割り当てられます。コピーオンライトプロトコルの詳細については、「Amazon DocumentDB のクローン作成の仕組み」を参照してください。

Amazon DocumentDB のクローン作成は、データを破損の危険にさらすことなく、本番データを使用してテスト環境を迅速にセットアップする場合に特に役立ちます。クローンは、次のようなさまざまなタイプのアプリケーションに使用できます。

- 潜在的な変更 (スキーマの変更やパラメータグループの変更など) を試して、すべての影響を評価する。
- データのエクスポートや分析クエリの実行など、大量のワークロードを扱うオペレーションをクローン上で実行する。
- 開発、テスト、またはその他の目的のために、本番 DB クラスターのコピーを作成する。

同じ Amazon DocumentDB クラスターから複数のクローンを作成できます。また、別のクローンから複数のクローンを作成することもできます。

Amazon DocumentDB のクローンを作成したら、Amazon DocumentDB インスタンスの設定を作成元の Amazon DocumentDB クラスターとは異なる方法で行えます。例えば、開発用途のクローンは、作成元の Amazon DocumentDB クラスターと同じ高可用性要件を満たす必要がない場合があります。この場合、Amazon DocumentDB クラスターで使用される複数の DB インスタンスではなく、単一の Amazon DocumentDB インスタンスを使用するようにクローンを設定できます。

テスト、開発などの用途へのクローンの使用が終了したら、クローンを削除できます。

Amazon DocumentDB クローン作成の制約事項

Amazon DocumentDB のクローン作成には、現在、次の制約事項があります。

- AWS リージョンで許可される DB クラスターの最大数まで、必要な数のクローンを作成できます。ただし、クローンを 15 個作成すると、次のクローンはフルコピーになります。クローン作成オペレーションは、ポイントインタイムリカバリと同様に機能します。
- ソース Amazon DocumentDB クラスターとは異なる AWS リージョンにクローンを作成することはできません。
- DB インスタンスを持たない Amazon DocumentDB クラスターからクローンを作成することはできません。少なくとも 1 つの DB インスタンスを持つ Amazon DocumentDB クラスターのクローン作成のみが可能です。
- クローンは、Amazon DocumentDB クラスターとは異なる仮想プライベートクラウド (VPC) で作成できます。その場合、VPCのサブネットは同じアベイラビリティーゾーンにマッピングする必要があります。

Amazon DocumentDB のクローン作成の仕組み

Amazon DocumentDB のクローン作成は Amazon DocumentDB クラスターのストレージレイヤーで動作します。これには、コピーオンライトプロトコルが使用されます。これは、Amazon DocumentDB ストレージボリュームをサポートする基盤となる耐久性の高いメディアという点で、高速かつスペース効率に優れています。Amazon DocumentDB クラスターボリュームの詳細については、「Amazon DocumentDB クラスターの管理」を参照してください。

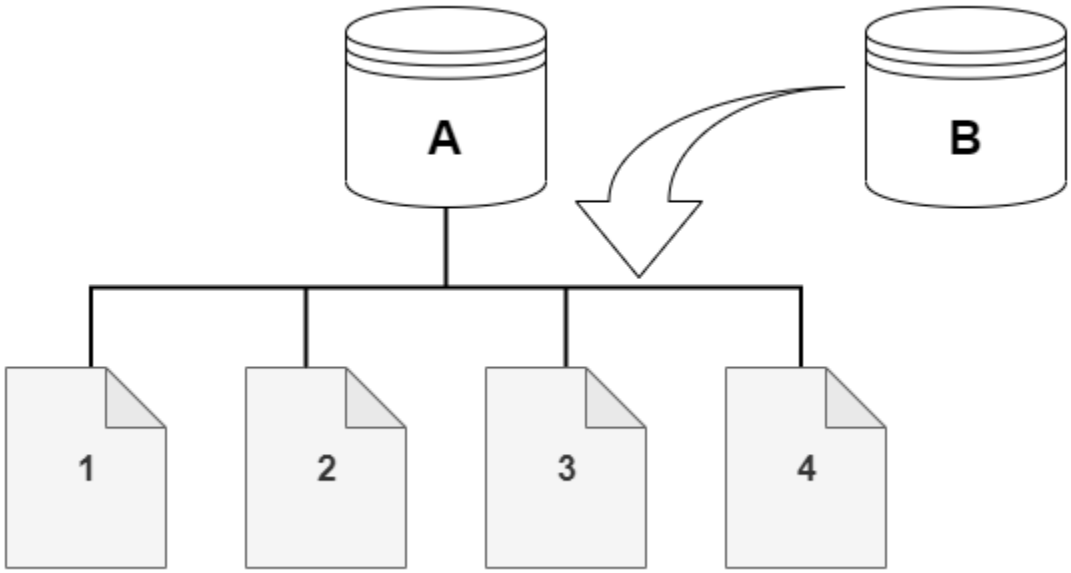
トピック

- コピーオンライトプロトコルの理解
- ソースクラスターボリュームの削除

コピーオンライトプロトコルの理解

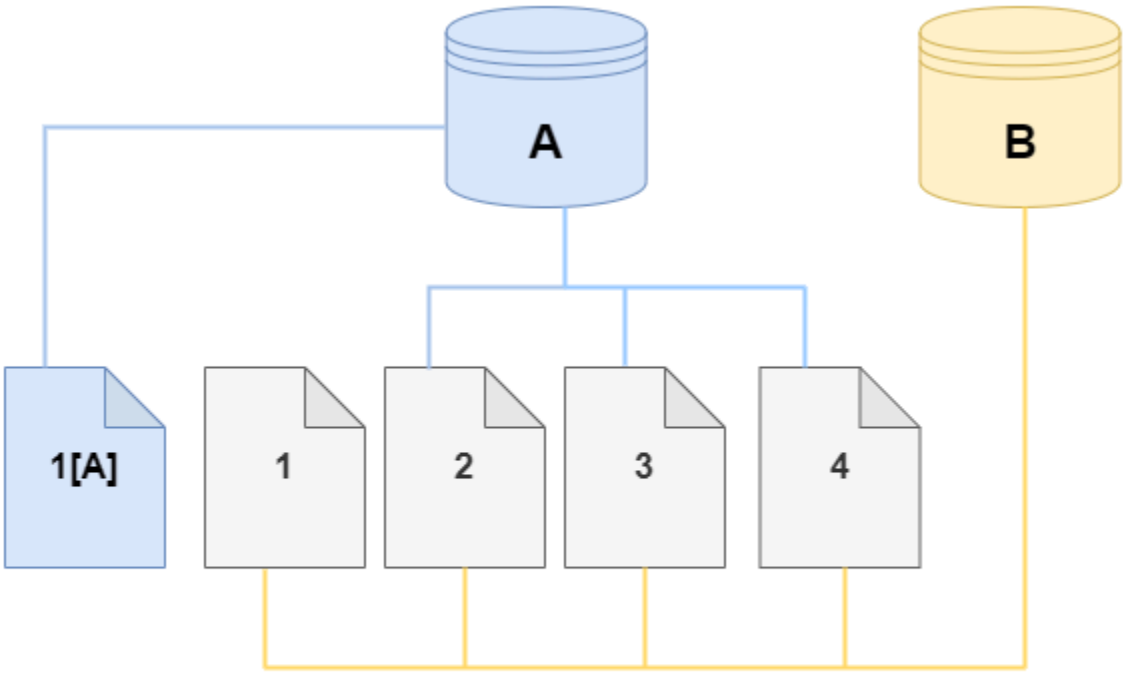
Amazon DocumentDB クラスターでは、基盤となる Amazon DocumentDB ストレージボリュームのページにデータが格納されます。

例えば、次の図には、4 つのデータページ 1、2、3、4 を持つ Amazon DocumentDB クラスター (A) があります。クローン B が Amazon DocumentDB クラスターから作成されたとします。クローンが作成されても、データはコピーされません。クローンは、作成元の Amazon DocumentDB クラスターと同じページのセットを参照しています。

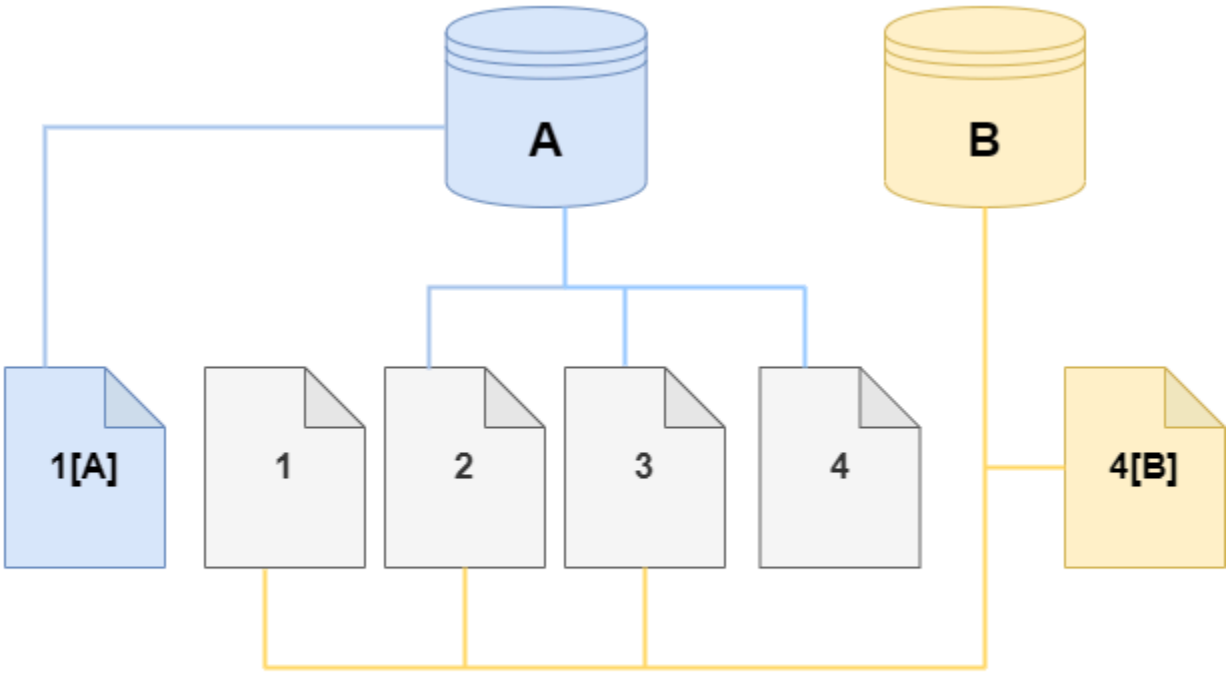


クローンが作成されたとき、通常は追加のストレージは必要ありません。コピーオンライトプロトコルでは、ソースセグメントと同じ物理ストレージメディア上のセグメントを使用します。追加のストレージが必要になるのは、ソースセグメントの容量がクローンセグメント全体に対して十分でない場合のみです。この場合、ソースセグメントは別の物理デバイスにコピーされます。

次の図に、前述と同様にクラスター A とそのクローン B を使用して動作中のコピーオンライトプロトコルの例を示します。Amazon DocumentDB クラスター (A) に変更を加えて、ページ 1 に保持されているデータが変更されたとします。元のページ 1 に書き込む代わりに、Amazon DocumentDB は新しいページ 1[A] を作成します。クラスター (A) の Amazon DocumentDB クラスターボリュームは、1[A]、2、3、4 ページを参照していますが、クローン (B) は引き続き元のページを参照しています。



クローンでは、ストレージボリュームのページ 4 に変更が加えられています。元のページ 4 に書き込む代わりに、Amazon DocumentDB は新しいページ 4[B] を作成します。クローンはページ 1、2、3、およびページ 4[B] を参照し、クラスター (A) は引き続き 1[A]、2、3、4 を参照しています。



時間が経過して作成元 Amazon DocumentDB クラスターボリュームとクローンの両方で追加の変更があると、その変更をキャプチャして保存するためにさらにストレージが必要になります。

ソースクラスターボリュームの削除

1 つ以上のクローンが関連付けられているソースクラスターボリュームを削除しても、そのクローンには影響しません。クローンは、ソースクラスターボリュームが前に所有していたページをポイントし続けます。

Amazon DocumentDB のクローンの作成

クローンは、ソース Amazon DocumentDB クラスターと同じ AWS アカウントに作成できます。そのためには、 AWS Management Console または AWS CLI と以下の手順を使用できます。

Amazon DocumentDB のクローン作成を使用すると、プロビジョニングされた Amazon DocumentDB クラスターから Amazon DocumentDB クラスターのプロビジョニングクローンを作成できます。

Using the AWS Management Console

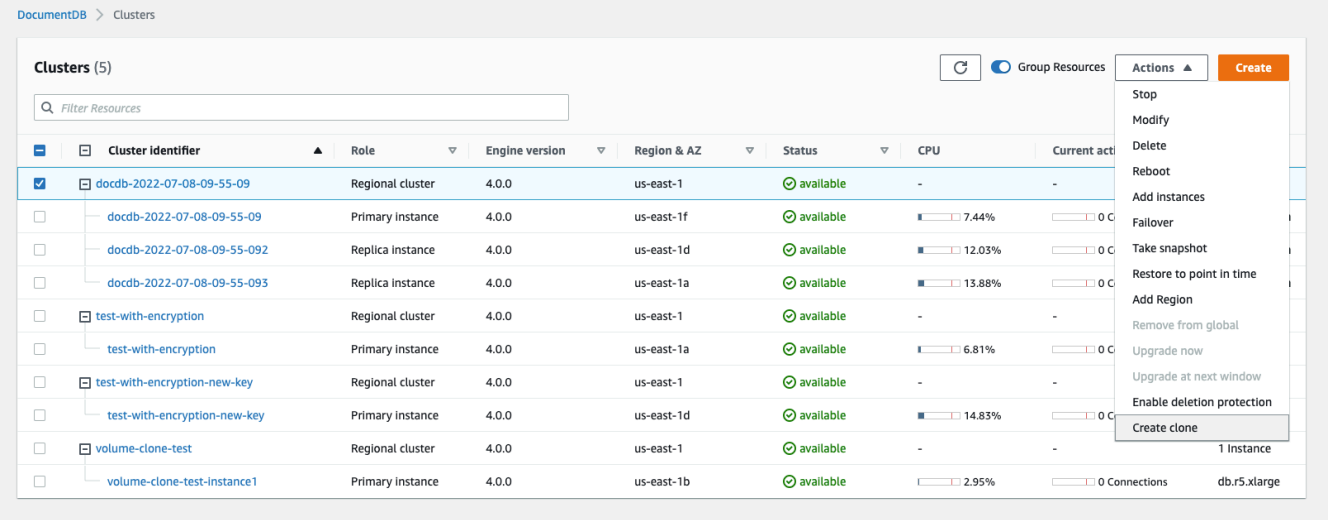
AWS Management Consoleを使用して Amazon DocumentDB クラスターのクローンを作成する手順を以下に示します。

1 つの Amazon DocumentDB インスタンスを持つ Amazon DocumentDB クラスターで AWS Management Console 結果を使用してクローンを作成します。

これらの手順は、クローンを作成しているのと同じ AWS アカウントが所有する DB クラスターに適用されます。クロス AWS アカウントのクローン作成は Amazon DocumentDB ではサポートされていないため、DB クラスターは同じアカウントによって所有されている必要があります。

を使用して、 AWS アカウントが所有する DB クラスターのクローンを作成するには AWS Management Console

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. リストから Amazon DocumentDB クラスターを選択し、[アクション] で、[クローンの作成] を選択します。

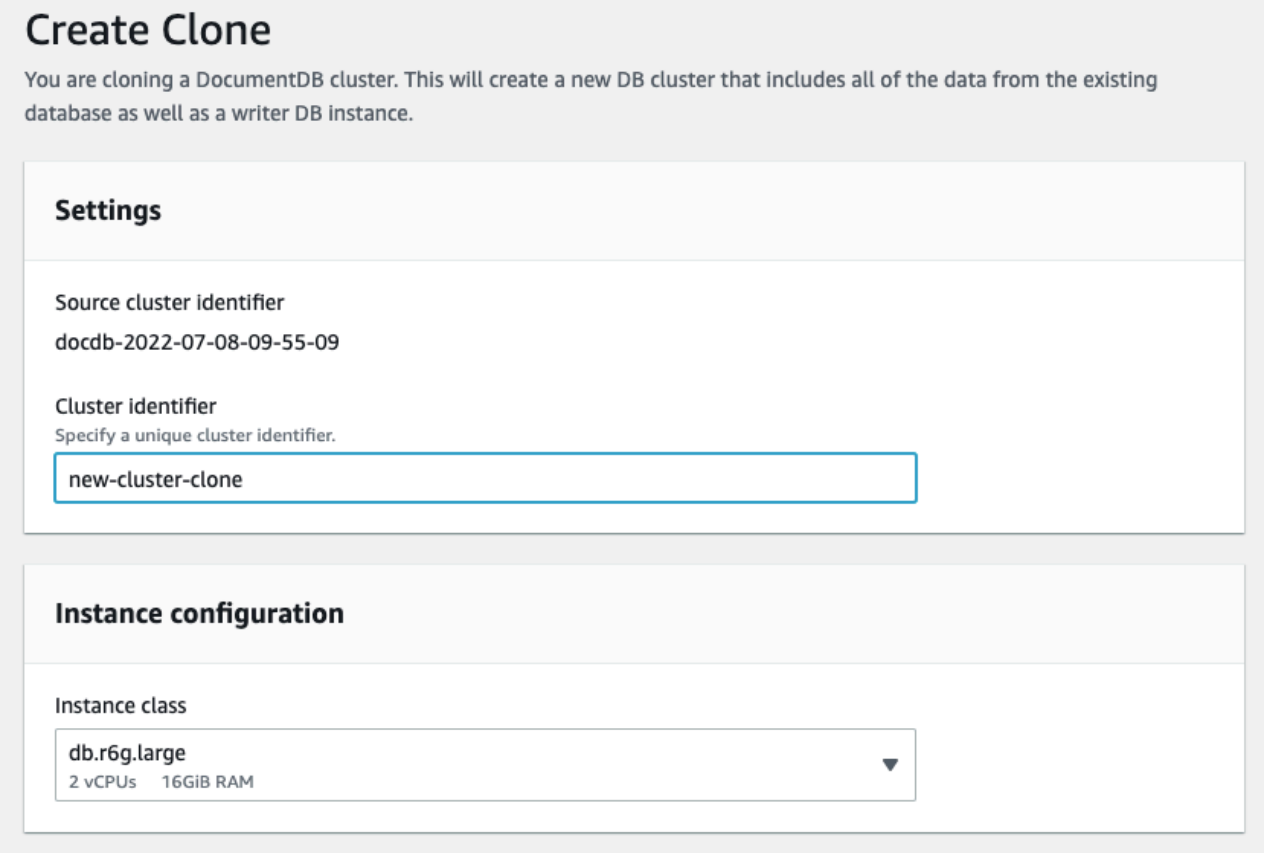


[クローンの作成] ページが開きます。そこで、[クラスター識別子]、[インスタンスクラス] などの Amazon DocumentDB クラスタークローンのオプションが設定できます。

4. [設定] セクションで、以下の手順を実行します。

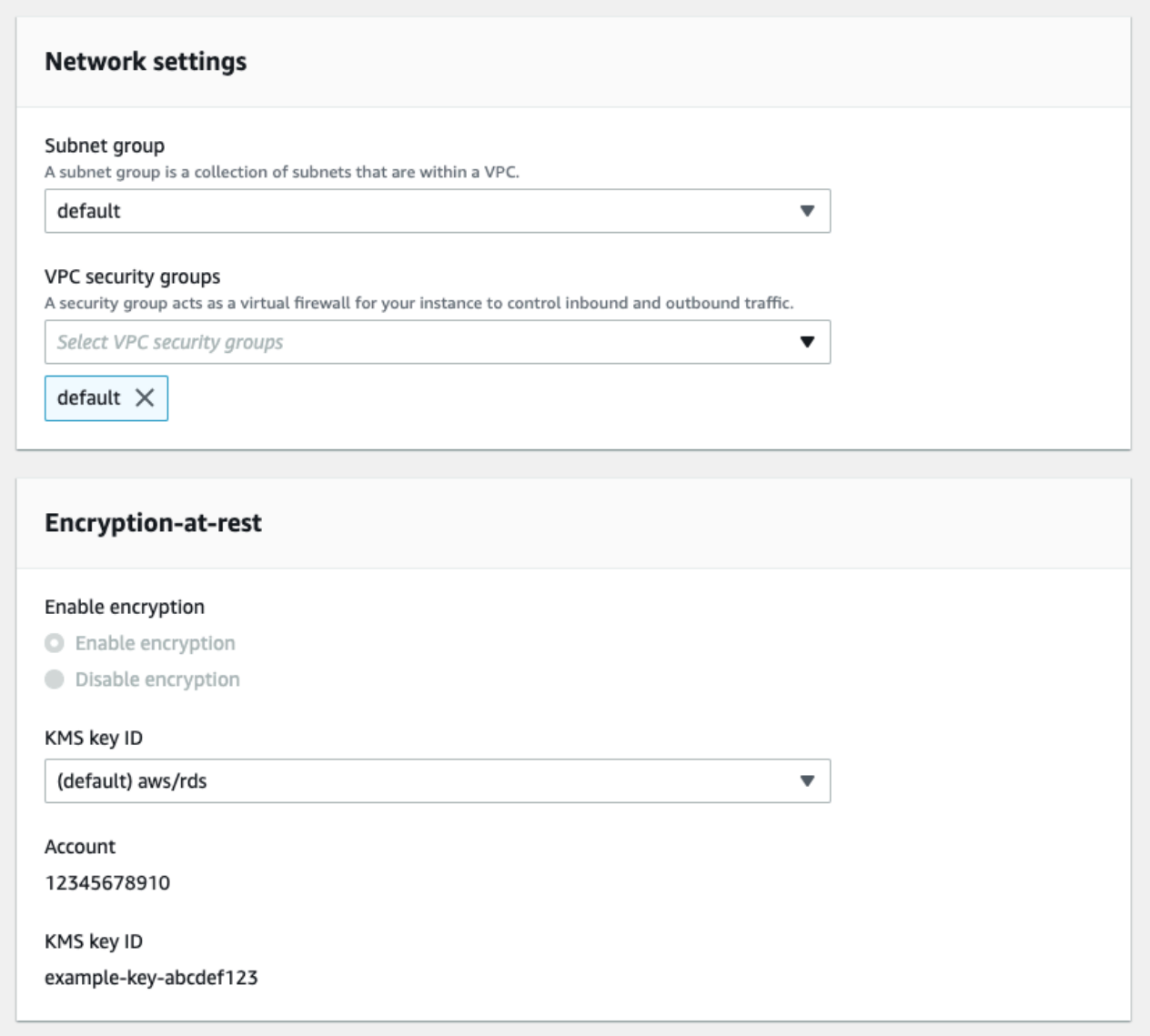
   a. [DB クラスター識別子] に、作成する Amazon DocumentDB クラスターのクローンに付ける名前を入力します。

b. [インスタンス設定] では、クローンとして作成された Amazon DocumentDB クラスターに適した [インスタンスクラス] を選択します。

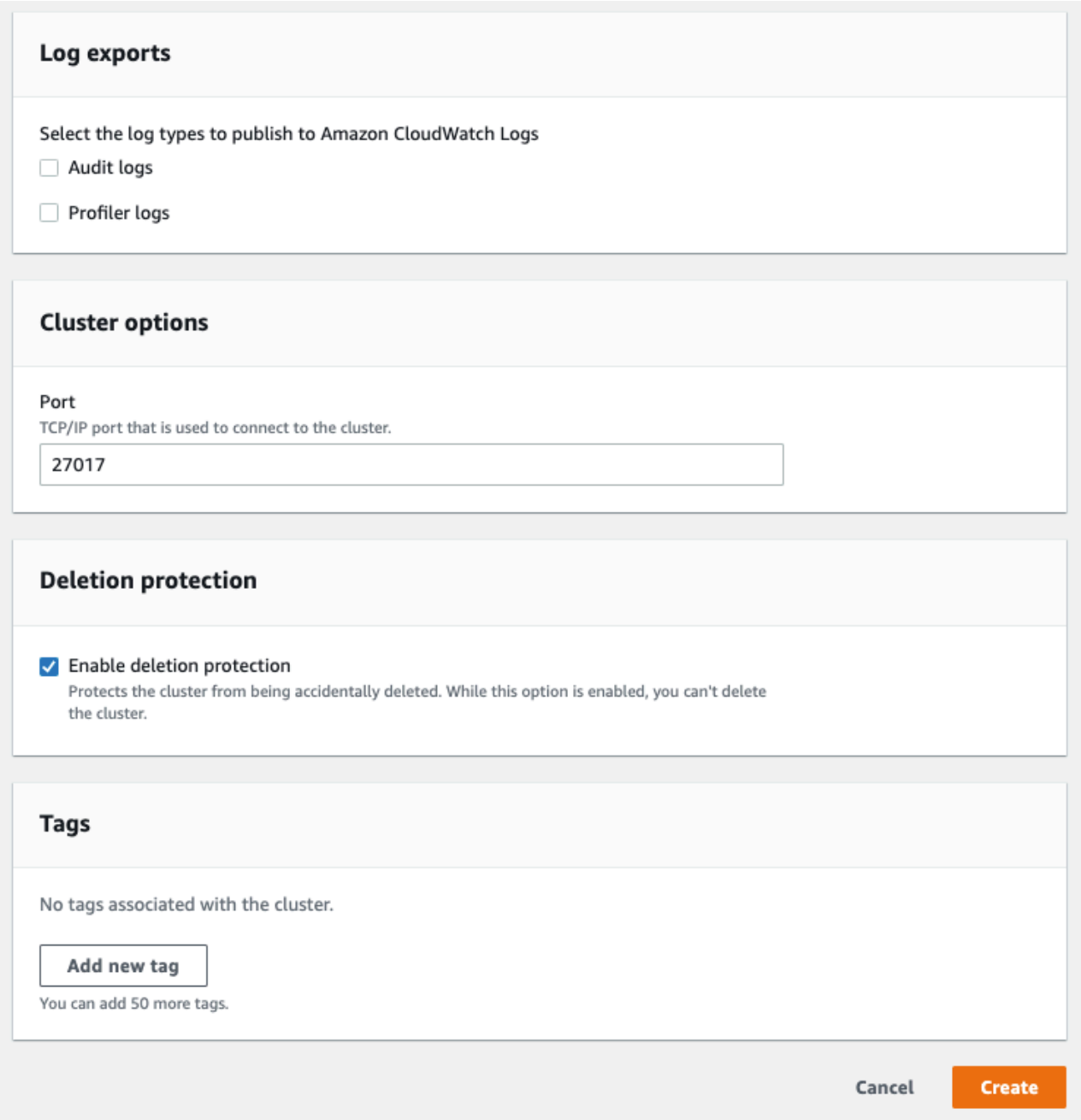


c. [ネットワーク設定] では、ユースケースに適したサブネットグループと、関連する VPC セキュリティグループを選択します。

d. [保管時の暗号化]では、作成元クラスター (クローンが作成されるクラスター) で暗号化が有効になっている場合、クローンとして作成されたクラスターでも暗号化が有効になっている必要があります。このシナリオに当てはまる場合、[暗号化を有効にする] オプションはグレー表示 (無効) になりますが、[暗号化を有効にする] オプションが選択されています。逆に、作成元クラスターで暗号化が有効になっていない場合は、[暗号化を有効にする] オプションが使用でき、暗号化を有効または無効にすることができます。



e. エクスポートするログの種類を選択し (オプション)、クラスターへの接続に使用する特定のポートを入力し、クラスターの誤削除からの保護を有効にして (デフォルトで有効) 、新しいクラスタークローンの設定を完了します。



f. Amazon DocumentDB クラスターのクローンのすべての設定の入力を完了します。Amazon DocumentDB クラスターとインスタンスの設定の詳細については、「Amazon DocumentDB クラスターの管理」を参照してください。

5. [クローンを作成] を選択して、選択した Amazon DocumentDB クラスターの Amazon DocumentDB クローンを起動します。

クローンが作成されると、コンソールの [データベース] セクションに他の Amazon DocumentDB クラスターとともに一覧表示され、現在の状態が表示されます。状態が [使用可能] の場合は、クローンはすぐに使用できます。

Using the AWS CLI

AWS CLI を使用して Amazon DocumentDB クラスターをクローンするには、いくつかのステップが必要です。

使用する`restore-db-cluster-to-point-in-time` AWS CLI コマンドは、Amazon DocumentDB インスタンスが 0 の空の Amazon DocumentDB クラスターになります。つまり、このコマンドは Amazon DocumentDB クラスターのみを復元し、クラスターの DB インスタンスは復元しません。これは、クローンが使用可能になった後に別途行います。プロセスは 2 ステップで、次のとおりです。

1. restore-db-cluster-to-point-in-time CLI コマンドを使用して、クローンを作成します。このコマンドで使用するパラメータで、作成する空の Amazon DocumentDB クラスター (クローン) の容量タイプなどの詳細が制御されます。
2. create-db-instance CLI コマンドを使用して、クローン用の Amazon DocumentDB インスタンスを作成し、復元された Amazon DocumentDB クラスターに Amazon DocumentDB インスタンスを再作成します。

次のコマンドは、 AWS CLI が AWS リージョンをデフォルトとして設定されていることを前提としています。この方法では、`--region` の名前を各コマンドに入力する手間が省けます。詳細については、「AWS CLIの設定」を参照してください。また、以下の各 CLI コマンドで `--region` を指定することもできます。

クローンの作成

`restore-db-cluster-to-point-in-time` CLI コマンドに渡す特定のパラメータは、さまざまです。渡す内容は、作成するクローンのタイプによって異なります。

次の手順に従い、プロビジョニングされた Amazon DocumentDB クラスターから Amazon DocumentDB プロビジョニングクローンを作成します。

作成元 Amazon DocumentDB クラスターと同じエンジンモードのクローンを作成するには

- `restore-db-cluster-to-point-in-time` CLI コマンドを使用して、次のパラメータに値を指定します。

- `--db-cluster-identifier` - クローン用の意味のある名前を選択します。restore-db-cluster-to-point-in-time CLI コマンド使用時に、クローンに名前を付けます。
- `--restore-type` - 出典 DB クラスターのクローンの作成に `copy-on-write` を使用します。このパラメータを指定しない場合、`restore-db-cluster-to-point-in-time` は、クローンを作成するのではなく、Amazon DocumentDB クラスターを復元します。`restore-type` のデフォルトは `full-copy` です。
- `--source-db-cluster-identifier` - クローンを作成する作成元 Amazon DocumentDB クラスターの名前を使用します。
- `--use-latest-restorable-time` - この値は、クローンの最新の復元可能なボリュームデータを指します。このパラメータは `restore-type copy-on-write` に必要ですが、一緒に `restore-to-time parameter` を使用することはできません。

次の例では、`my-source-cluster` という名前のクラスターから `my-clone` という名前のクローンを作成します。

Linux、macOS、Unix の場合:

```
aws docdb restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-source-cluster \
    --db-cluster-identifier my-clone \
    --restore-type copy-on-write \
    --use-latest-restorable-time
```

Windows の場合:

```
aws docdb restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier my-source-cluster ^
    --db-cluster-identifier my-clone ^
    --restore-type copy-on-write ^
    --use-latest-restorable-time
```

このコマンドは、クローンの詳細を含む JSON オブジェクトを返します。クローンの DB インスタンスを作成する前に、作成した DB クラスターのクローンが使用可能であることを確認します。詳細については、下記の「ステータスの確認とクローンの詳細の取得」をご参照ください。

ステータスの確認とクローンの詳細の取得

次のコマンドを使用して、新しく作成した空の DB クラスターのステータスが確認できます。

```
$ aws docdb describe-db-clusters --db-cluster-identifier my-clone --query '*[].
[Status]' --output text
```

または、次の AWS CLI クエリを使用して、クローンの DB インスタンスを作成するために必要なステータスやその他の値を取得できます。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone \
  --query '*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}'
```

Windows の場合:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone ^
  --query "*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}"
```

このクエリにより、以下のような出力が返されます。

```
[
  {
        "Status": "available",
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
    }
]
```

クローン用の Amazon DocumentDB インスタンスの作成

クローン用の DB インスタンスを作成するには、create-db-instance CLI コマンドを使用します。

`--db-instance-class` パラメータは、プロビジョニングされた Amazon DocumentDB クラスターにのみ使用されます。

Linux、macOS、Unix の場合:

```
aws docdb create-db-instance \
    --db-instance-identifier my-new-db \
    --db-cluster-identifier my-clone \
    --db-instance-class  db.r5.4xlarge \
```

```
    --engine docdb
```

Windows の場合:

```
aws docdb create-db-instance ^
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class  db.r5.4xlarge ^
    --engine docdb
```

クローン作成に使用するパラメータ

次の表は `restore-db-cluster-to-point-in-time` を使用して Amazon DocumentDB クラスターのクローンを作成する際に使用されるさまざまなパラメータをまとめたものです。

パラメータ	説明
--source-db-cluster-identifier	クローンを作成する作成元 Amazon DocumentDB クラスターの名前を使用します。
--db-cluster-identifier	クローン用に意味のある名前を選択します。 `restore-db-cluster-to-point-in-time` コマンドでクローンの名前を付けます。次に、この名前を `create-db-instance` コマンドに渡します。
--restore-type	`--restore-type` として `copy-on-write` を指定すると、作成元 Amazon DocumentDB クラスターを復元するのではなく、作成元 DB クラスターのクローンを作成します。
--use-latest-restorable-time	この値は、クローンの最新の復元可能なボリュームデータを指します。

Amazon DocumentDB クラスターの耐障害性について理解する

Amazon DocumentDB クラスターは、耐障害性を持つように設計されています。各クラスターのボリュームは 1 つの で複数のアベイラビリティーゾーンにまたがり AWS リージョン、各アベイラビリティーゾーンにはクラスターのボリュームデータのコピーが含まれます。この機能は、クラスターがデータ喪失なしでアベイラビリティーゾーンの障害に耐えることができ、発生するのはサービスの短時間の中断のみであることを意味します。

クラスターのプライマリインスタンスが失敗した場合、Amazon DocumentDB は以下のいずれかの方法で、新しいプライマリインスタンスに自動的にフェイルオーバーします。

- 既存の Amazon DocumentDB レプリカを新しいプライマリインスタンスに昇格し、昇格されたレプリカの置き換えを作成する。レプリカインスタンスへのフェイルオーバーは、通常 30 秒未満で完了します。この期間中、読み取りおよび書き込みオペレーションが短時間中断することがあります。DB クラスターの可用性を高めるために、複数のアベイラビリティーゾーン内で少なくとも 1 つ以上の Amazon DocumentDB レプリカを作成することをお勧めします。
- 新しいプライマリインスタンスを作成する。これは、クラスター内にレプリカ・インスタンスがなく、完了までに数分かかる場合のみ発生します。

クラスターに 1 つ以上の Amazon DocumentDB レプリカがある場合は、障害発生中に 1 つの Amazon DocumentDB レプリカがプライマリインスタンスに昇格されます。障害イベントによって短い中断が発生し、その間例外によって読み取りと書き込みオペレーションが失敗します。ただし、一般的なサービスの復元時間は 120 秒未満であり、多くの場合 60 秒未満で復元されます。DB クラスターの可用性を高めるために、複数のアベイラビリティーゾーン内で少なくとも 1 つ以上の Amazon DocumentDB レプリカを作成することをお勧めします。

各レプリカに優先度を割り当てることで、Amazon DocumentDB レプリカがプライマリインスタンスに昇格される順序をカスタマイズできます。優先度の範囲は、最も高い 0 から最も低い 15 までです。プライマリインスタンスが失敗した場合、最も高い優先度の Amazon DocumentDB レプリカが、新しいプライマリインスタンスに昇格します。Amazon DocumentDB レプリカの優先度はいつでも変更できます。優先度を変更しても、フェイルオーバーはトリガーされません。`modify-db-instance` オペレーションで、`--promotion-tier` パラメータを使用できます。インスタンスのフェイルオーバー優先度のカスタマイズの詳細については、「Amazon DocumentDB フェイルオーバー」を参照してください。

複数の Amazon DocumentDB レプリカで同じ優先度を共有でき、その場合は昇格階層が発生します。複数の Amazon DocumentDB レプリカで同じ優先度を共有する場合、最大サイズのレプリカがプライマリに昇格します。複数の Amazon DocumentDB レプリカで同じ優先度とサイズを共有する場合、昇格階層の任意のレプリカが昇格します。

クラスターに Amazon DocumentDB レプリカが含まれていない場合、障害イベントの発生時にプライマリインスタンスが再作成されます。障害イベントによって中断が発生し、その間例外によって読み取りと書き込みオペレーションが失敗します。新しいプライマリインスタンスが再作成されると、サービスが回復します。これは、通常は 10 分未満で行われます。Amazon DocumentDB レプリカのプライマリインスタンスへの昇格は、新しいプライマリインスタンスの作成よりもはるかに短時間で実行されます。

Amazon DocumentDB インスタンスの管理

以下のトピックでは、Amazon DocumentDB インスタンスの管理に役立つ情報を提供します。インスタンスクラスとステータスの詳細、およびインスタンスを作成、削除、変更する方法が含まれます。

トピック

- インスタンスのステータスの判断
- Amazon DocumentDB インスタンスのライフサイクル
- インスタンスクラスの管理
- NVMe-backed インスタンス

インスタンスのステータスの判断

有効なインスタンスのステータス、その意味、インスタンスのステータスの判断方法については、「Amazon DocumentDB インスタンスのステータスのモニタリング」を参照してください。

Amazon DocumentDB インスタンスのライフサイクル

Amazon DocumentDB インスタンスのライフサイクルには、インスタンスの作成、変更、保守、アップグレード、バックアップと復元の実行、再起動、削除が含まれます。このセクションでは、これらのプロセスを完了する方法について説明します。

トピック

- クラスターへの Amazon DocumentDB インスタンスの追加
- Amazon DocumentDB インスタンスの説明
- Amazon DocumentDB インスタンスの変更
- Amazon DocumentDB インスタンスの再起動
- Amazon DocumentDB インスタンスの削除

AWS Management Console または を使用して、新しい Amazon DocumentDB インスタンスを作成できます AWS CLI。クラスターにインスタンスを追加するには、クラスターが使用可能な状態になっている必要があります。停止しているクラスターにインスタンスを追加することはできません。クラスターが停止している場合は、最初にクラスターを起動し、クラスターが使用可能になるまで

待ってから、インスタンスを追加します。詳細については、「Amazon DocumentDB クラスターの停止と開始」を参照してください。

> **Note**
>
> コンソールを使用して Amazon DocumentDB クラスターを作成した場合、同時にインスタンスが自動的に作成されます。追加のインスタンスを作成する場合は、以下のいずれかの手順を使用します。

クラスターへの Amazon DocumentDB インスタンスの追加

Using the AWS Management Console

次の手順に従い、Amazon DocumentDB コンソールを使用してクラスターのインスタンスを作成します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. インスタンスの追加先のクラスターを選択するには、クラスター名の左にあるボタンを選択します。
4. [アクション] を選択し、[インスタンスの追加] を選択します。
5. [Add instance to: (インスタンスの追加先:)<cluster-name>] ページで、クラスターに追加する各インスタンスに対して以下の手順を繰り返します。最大数は 15 です。

   a. インスタンス識別子 - このインスタンスの一意の識別子を入力するか、クラスター識別子に基づき Amazon DocumentDB でインスタンスを指定できるようにします。

      インスタンスの名前付けの制約:

- 長さは [1 ～ 63] 個の英字、数字、またはハイフンです。
- 1 字目は英字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
- Amazon RDS、Neptune、Amazon DocumentDB のすべてのインスタンスで AWS アカウント、リージョンごとに一意である必要があります。

b. インスタンスクラス - ドロップダウンリストから、このインスタンスに必要なインスタンスタイプを選択します。

c. プロモーション枠 - ドロップダウンリストからインスタンスのプロモーション枠を選択するか、[指定なし] を選択して Amazon DocumentDB によってインスタンスのプロモーション枠が設定されるようにします。数値が小さいほど優先順位が高くなります。詳細については、「フェイルオーバーターゲットの制御」を参照してください。

d. インスタンスをさらに追加するには、[Add additional instances (追加のインスタンスを追加する)] を選択してステップ a、b、c を繰り返します。

6. オペレーションを完了します。

- インスタンスをクラスターに追加するには、[作成] を選択します。
- 操作をキャンセルするには、[キャンセル] を選択します。

インスタンスが作成されるまでに数分かかります。コンソールまたは を使用して、インスタンスのステータス AWS CLI を表示できます。詳細については、「インスタンスのステータスのモニタリング」を参照してください。

Using the AWS CLI

クラスターのプライマリインスタンスを作成するには、以下のパラメータを指定して `create-db-instance` AWS CLI オペレーションを使用します。

- **`--db-instance-class`** — 必須。インスタンスのコンピューティング性能とメモリ容量 (例: `db.m4.large`)。すべての AWS リージョン中で、インスタンスクラスが使用できるわけではありません。
- **`--db-instance-identifier`** — 必須。インスタンスを識別する文字列。

  インスタンスの名前付けの制約:

  - 長さは [1 ～ 63] 個の英字、数字、またはハイフンです。

- 1 字目は英字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
- Amazon RDS、Neptune、Amazon DocumentDB のすべてのインスタンスで AWS アカウント、リージョンごとに一意である必要があります。

- **--engine** — 必須。docdb を指定してください。
- **--availability-zone** - オプション。このインスタンスが作成されるアベイラビリティーゾーン このパラメータを使用して、さまざまなアベイラビリティーゾーンで耐障害性を強化するインスタンスを見つけます。詳細については、「Amazon DocumentDB 高可用性とレプリケーション」を参照してください。
- **--promotion-tier** - オプション。このインスタンスのフェイルオーバー優先度の階層。0 ~ 15 の数値である必要があります。数値が小さいほど、優先順位が高くなります。詳細については、「フェイルオーバーターゲットの制御」を参照してください。

1. 最初に、インスタンスを作成できるアベイラビリティーゾーンを決定します。

   インスタンスを作成する前にアベイラビリティーゾーンを指定する場合は、次のコマンドを実行して Amazon DocumentDB クラスターで使用できるアベイラビリティーゾーンを確認します。

   Linux、macOS、Unix の場合:

   ```
   aws docdb describe-db-clusters \
          --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
   ```

   Windows の場合:

   ```
   aws docdb describe-db-clusters ^
          --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
   ```

   このオペレーションによる出力は、次のようになります。

   ```
   [
       [
           "sample-cluster",
           [
               "us-east-1c",
               "us-east-1b",
   ```

```
            "us-east-1a"
        ]
    ]
]
```

2. 次に、リージョンで作成できるインスタンスクラスを決定します。

   リージョンで利用可能なインスタンスクラスを確認するには、次のコマンドを実行します。出力から、Amazon DocumentDB クラスターに追加したいインスタンスのためのインスタンスクラスを選択します。

   Linux、macOS、Unix の場合:

```
aws docdb describe-orderable-db-instance-options \
        --engine docdb \
        --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
```

   Windows の場合:

```
aws docdb describe-orderable-db-instance-options ^
        --engine docdb ^
        --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
```

   このオペレーションによる出力は、次のようになります。

```
[
    "db.r5.16xlarge",
    "db.r5.2xlarge",
    "db.r5.4xlarge",
    "db.r5.8xlarge",
    "db.r5.large",
    "db.r5.xlarge"
]
```

3. 最後に、Amazon DocumentDB クラスターにインスタンスを追加します。

   Amazon DocumentDB クラスターにインスタンスを追加するには、次のコマンドを実行します。

   Linux、macOS、Unix の場合:

```
aws docdb create-db-instance \
       --db-cluster-identifier sample-cluster \
       --db-instance-identifier sample-instance-2 \
       --availability-zone us-east-1b \
       --promotion-tier 2 \
       --db-instance-class db.r5.xlarge \
       --engine docdb
```

Windows の場合:

```
aws docdb create-db-instance ^
       --db-cluster-identifier sample-cluster ^
       --db-instance-identifier sample-instance-2 ^
       --availability-zone us-east-1b ^
       --promotion-tier 2 ^
       --db-instance-class db.r5.xlarge ^
       --engine docdb
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "sample-instance-2",
        "DBInstanceClass": "db.r5.xlarge",
        "Engine": "docdb",
        "DBInstanceStatus": "creating",
        "PreferredBackupWindow": "02:00-02:30",
        "BackupRetentionPeriod": 1,
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "AvailabilityZone": "us-east-1b",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-6242c31a",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
```

```
                {
                    "SubnetIdentifier": "subnet-abcd0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-wxyz0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2b"
                    },
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sun:11:35-sun:12:05",
        "PendingModifiedValues": {},
        "EngineVersion": "3.6.0",
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "DBClusterIdentifier": "sample-cluster",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "CACertificateIdentifier": "rds-ca-2019",
        "PromotionTier": 2,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-instance-2"
    }
}
```

インスタンスが作成されるまでに数分かかります。コンソールまたは を使用して、インスタンスのステータス AWS CLI を表示できます。詳細については、「Amazon DocumentDB インスタンスのステータスのモニタリング」を参照してください。

Amazon DocumentDB インスタンスの説明

Amazon DocumentDB マネジメントコンソールまたは AWS CLI を使用して、接続エンドポイント、セキュリティグループ VPC、認証機関、Amazon DocumentDB インスタンスに関連するパラメータグループなどの詳細を確認できます。

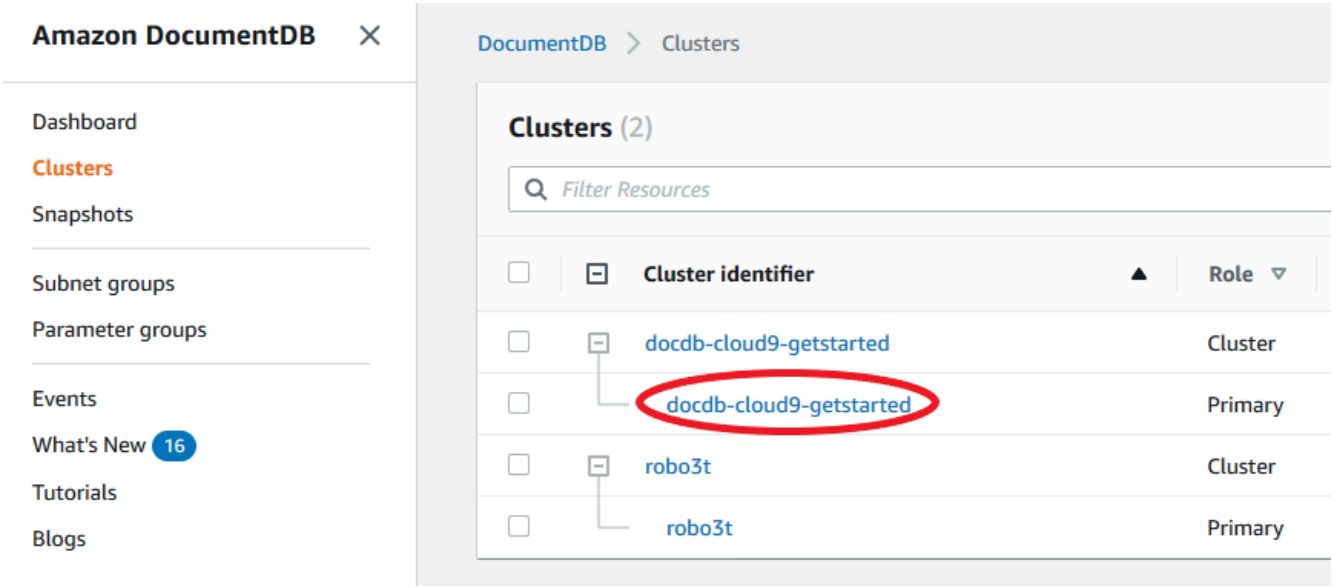
Using the AWS Management Console

AWS Management Consoleを使用してインスタンスの詳細を確認するには、次の手順に従います。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. クラスターナビゲーションボックスに、[Cluster Identifier]の列が表示されます。インスタンスは、以下のスクリーンショットのように、クラスターの下に表示されます。



4. インスタンスの一覧で、詳細を表示したいインスタンスの名前を選択します。インスタンスに関する情報は、次のグループに分けられます。

   - 概要 エンジンのバージョン、クラス、ステータス、保留中のメンテナンスなど、インスタンスに関する一般情報。
   - Connectivity & Security - Connect セクションには、[mongo] シェルまたはアプリケーションを使用してこのインスタンスに接続するための接続エンドポイントが一覧表示されま

す。[セキュリティグループ] セクションには、このインスタンスに関連付けられているセキュリティグループと、その VPC ID や説明が一覧表示されます。

- 設定 - [詳細] セクションには、インスタンスの Amazon リソースネーム (ARN)、エンドポイント、ロール、クラス、認証機関など、インスタンスの設定とステータスが一覧表示されます。また、インスタンスのセキュリティとネットワーク設定、バックアップ情報も一覧表示されます。[クラスターの詳細] セクションには、このインスタンスが属するクラスターの詳細が表示されます。[クラスターのインスタンス] セクションには、クラスターに属するすべてのインスタンスが、各インスタンスのロールとクラスターパラメータグループのステータスとともに一覧表示されます。

  > **Note**
  >
  > クラスターの詳細 ヘッダーの次に Modify を選択することにより、インスタンスに関連付けられているクラスターを変更できます。詳細については、「Amazon DocumentDB クラスターの変更」を参照してください。

- [モニタリング] — このクラスターの Amazon CloudWatch Logs メトリクス。詳細については、「Amazon DocumentDB と CloudWatch のモニタリング」を参照してください。
- [イベントとタグ] — [最近のイベント] セクションには、このインスタンスの最近のイベントの一覧が表示されます。Amazon DocumentDB は、クラスター、インスタンス、スナップショット、セキュリティグループ、およびクラスターパラメータグループに関連するイベントのレコードを保持します。この情報には、各イベントに関連する日付、時刻、およびメッセージが含まれます。[タグ] セクションには、このクラスターにアタッチされているタグが一覧表示されます。詳細については、「Amazon DocumentDB リソースへのタグ付け」を参照してください。

Using the AWS CLI

を使用して Amazon DocumentDB インスタンスの詳細を表示するには AWS CLI、以下の例に示すように `describe-db-clusters` コマンドを使用します。詳細については、「Amazon DocumentDB リソース管理 API リファレンス」の「`DescribeDBInstances`」を参照してください。

> **Note**
>
> クラスターやインスタンスのライフサイクル管理などの特定の管理機能において、Amazon DocumentDB は Amazon RDS と共有される運用テクノロジーを使用し

ます。filterName=engine,Values=docdb のフィルターパラメータは Amazon DocumentDB クラスターのみを返します。

1. List all Amazon DocumentDB instances.

   次の AWS CLI コードは、リージョン内のすべての Amazon DocumentDB インスタンスの詳細を一覧表示します。

   Linux、macOS、Unix の場合:

   ```
   aws docdb describe-db-instances \
       --filter Name=engine,Values=docdb
   ```

   Windows の場合:

   ```
   aws docdb describe-db-instances \
       --filter Name=engine,Values=docdb
   ```

2. 指定した Amazon DocumentDB インスタンスのすべての詳細を一覧表示する

   次のコードは sample-cluster-instance の詳細を一覧します。--db-instance-identifier パラメータにインスタンスの名前を含めると、出力はその特定のインスタンスの情報に制限されます。

   Linux、macOS、Unix の場合:

   ```
   aws docdb describe-db-instances \
       --db-instance-identifier sample-cluster-instance
   ```

   Windows の場合:

   ```
   aws docdb describe-db-instances \
       --db-instance-identifier sample-cluster-instance
   ```

   このオペレーションの出力は、次のようになります。

   ```
   {
       "DBInstances": [
           {
   ```

```
            "DbiResourceId": "db-BJKKB54PIDV5QFKGVRX5T3S6GM",
            "DBInstanceArn": "arn:aws:rds:us-east-1:012345678901:db:sample-
cluster-instance-00",
            "VpcSecurityGroups": [
                {
                    "VpcSecurityGroupId": "sg-77186e0d",
                    "Status": "active"
                }
            ],
            "DBInstanceClass": "db.r5.large",
            "DBInstanceStatus": "creating",
            "AutoMinorVersionUpgrade": true,
            "PreferredMaintenanceWindow": "fri:09:32-fri:10:02",
            "BackupRetentionPeriod": 1,
            "StorageEncrypted": true,
            "DBClusterIdentifier": "sample-cluster",
            "EngineVersion": "3.6.0",
            "AvailabilityZone": "us-east-1a",
            "Engine": "docdb",
            "PromotionTier": 2,
            "DBInstanceIdentifier": "sample-cluster-instance",
            "PreferredBackupWindow": "00:00-00:30",
            "PubliclyAccessible": false,
            "DBSubnetGroup": {
                "DBSubnetGroupName": "default",
                "Subnets": [
                    {
                        "SubnetIdentifier": "subnet-4e26d263",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1a"
                        },
                        "SubnetStatus": "Active"
                    },
                    {
                        "SubnetIdentifier": "subnet-afc329f4",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1c"
                        },
                        "SubnetStatus": "Active"
                    },
                    {
                        "SubnetIdentifier": "subnet-b3806e8f",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1e"
```

```
                        },
                        "SubnetStatus": "Active"
                    },
                    {
                        "SubnetIdentifier": "subnet-53ab3636",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1d"
                        },
                        "SubnetStatus": "Active"
                    },
                    {
                        "SubnetIdentifier": "subnet-991cb8d0",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1b"
                        },
                        "SubnetStatus": "Active"
                    },
                    {
                        "SubnetIdentifier": "subnet-29ab1025",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1f"
                        },
                        "SubnetStatus": "Active"
                    }
                ],
                "VpcId": "vpc-91280df6",
                "DBSubnetGroupDescription": "default",
                "SubnetGroupStatus": "Complete"
            },
            "PendingModifiedValues": {},
            "KmsKeyId": "arn:aws:kms:us-east-1:012345678901:key/0961325d-
a50b-44d4-b6a0-a177d5ff730b"
        }
    ]
}
```

Amazon DocumentDB インスタンスの変更

Amazon DocumentDB インスタンスは、 AWS Management Console または を使用して変更できます AWS CLI。インスタンスを変更するには、インスタンスが使用可能な状態である必要があります。停止しているインスタンスを変更することはできません。クラスターが停止している場合は、最

初にクラスターを起動し、インスタンスが使用可能になるまで待ってから、必要な変更を加えます。詳細については、「Amazon DocumentDB クラスターの停止と開始」を参照してください。

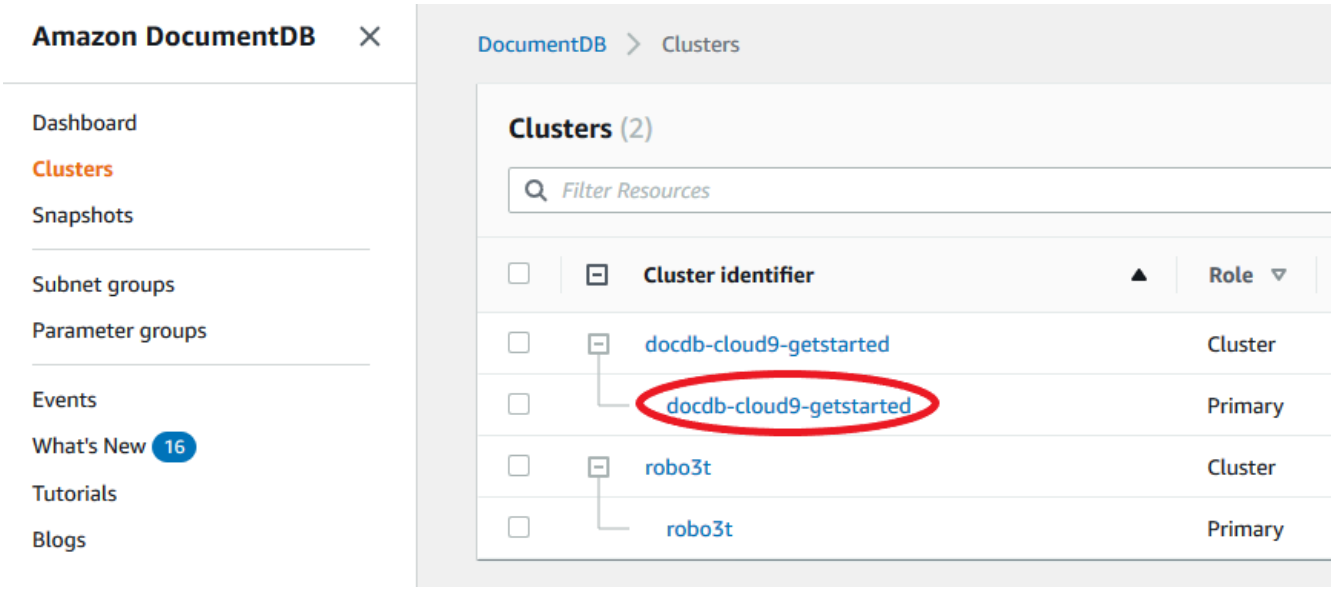
Using the AWS Management Console

コンソールを使用して特定の Amazon DocumentDB インスタンスを変更するには、以下の手順を実行します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. クラスターナビゲーションボックスに、[Cluster Identifier]の列が表示されます。インスタンスは、以下のスクリーンショットのように、クラスターの下に表示されます。



4. 変更するインスタンスの左側にあるボックスをチェックします。
5. [アクション] を選択してから、[変更] を選択します。
6. [インスタンスの変更: <instance-name>] ペインで、必要な変更を行います。以下の変更を行うことができます。

- インスタンス仕様 - インスタンス識別子とクラス。インスタンス識別子の命名制約:
  - インスタンス識別子 — 現在のリージョンで が所有するすべてのインスタンス AWS アカウント に固有の名前を入力します。インスタンス識別子は [1 — 63] 英数字またはハイフンを使用し、最初の文字に文字を使用します。最後の文字をハイフンにすることはできず、ハイフンを 2 つ続けて使用することもできません。
  - インスタンスクラス — ドロップダウンメニューから、Amazon DocumentDB インスタンスのためのインスタンスクラスを選択します。詳細については、「インスタンスクラスの管理」を参照してください。
- 認証機関 - このインスタンスのサーバー証明書。詳細については、「Amazon DocumentDB TLS 証明書の更新」を参照してください。
- フェイルオーバー - フェイルオーバーにより、昇格階層が最も高いインスタンスがプライマリに昇格されます 詳細については、「Amazon DocumentDB フェイルオーバー」を参照してください。
- メンテナンス - 保留中の変更またはパッチをクラスター内のインスタンスに適用するメンテナンスウィンドウ。

7. 完了したら、[続行] を選択して変更の概要を表示します。
8. 変更を確認したら、直ちに、あるいは、[変更のスケジューリング] の下にある次のメンテナンスウィンドウ中に適用することができます。[インスタンスの変更] を選択して、変更を保存します。または、[キャンセル] を選択して変更を破棄することもできます。

変更が適用されるまでに数分かかります。インスタンスを使用できるのは、そのステータスが [available] である場合のみです。コンソールまたは AWS CLIを使用して、インスタンスのステータスをモニタリングできます。詳細については、「Amazon DocumentDB インスタンスのステータスのモニタリング」を参照してください。

Using the AWS CLI

を使用して特定の Amazon DocumentDB インスタンスを変更するには AWS CLI、次のパラメータを指定`modify-db-instance`して を使用します。詳細については、「ModifyDBInstance」を参照してください。次のコードは、`sample-instance` インスタンスのインスタンスクラスを `db.r5.large` に変更します。

パラメータ

- **`--db-instance-identifier`** — 必須。変更するインスタンスの識別子。

- **--db-instance-class** - オプション。インスタンスの新しいコンピューティング能力とメモリ容量 (例: db.r5.large)。すべてのインスタンスクラスがすべての AWS リージョンで利用できるわけではありません。インスタンスクラスを変更すると、変更中に機能停止が発生します。ApplyImmediately がこのリクエストの [true] として指定されない限り、変更は次のメンテナンスウィンドウ中に適用されます。
- **--apply-immediately** または **--no-apply-immediately** - オプション。この修正をすぐに適用するか、次のメンテナンスウィンドウまで待機するか指定します。このパラメータを省略した場合、次のメンテナンスウィンドウ中に変更が実行されます。

Example

Linux、macOS、Unix の場合:

```
aws docdb modify-db-instance \
       --db-instance-identifier sample-instance \
       --db-instance-class db.r5.large \
       --apply-immediately
```

Windows の場合:

```
aws docdb modify-db-instance ^
       --db-instance-identifier sample-instance ^
       --db-instance-class db.r5.large ^
       --apply-immediately
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBInstances": [
        {
            "DBInstanceIdentifier": "sample-instance-1",
            "DBInstanceClass": "db.r5.large",
            "Engine": "docdb",
            "DBInstanceStatus": "modifying",
            "Endpoint": {
                "Address": "sample-instance-1.node.us-east-1.docdb.amazonaws.com",
                "Port": 27017,
                "HostedZoneId": "ABCDEFGHIJKLM"
            },
            "InstanceCreateTime": "2020-01-10T22:18:55.921Z",
```

```
            "PreferredBackupWindow": "02:00-02:30",
            "BackupRetentionPeriod": 1,
            "VpcSecurityGroups": [
                {
                    "VpcSecurityGroupId": "sg-abcd0123",
                    "Status": "active"
                }
            ],
            "AvailabilityZone": "us-east-1a",
            "DBSubnetGroup": {
                "DBSubnetGroupName": "default",
                "DBSubnetGroupDescription": "default",
                "VpcId": "vpc-abcd0123",
                "SubnetGroupStatus": "Complete",
                "Subnets": [
                    {
                        "SubnetIdentifier": "subnet-abcd0123",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1a"
                        },
                        "SubnetStatus": "Active"
                    },
                    {
                        "SubnetIdentifier": "subnet-abcd0123",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1b"
                        },
                        "SubnetStatus": "Active"
                    }
                ]
            },
            "PreferredMaintenanceWindow": "sun:10:57-sun:11:27",
            "PendingModifiedValues": {
                "DBInstanceClass": "db.r5.large"
            },
            "EngineVersion": "3.6.0",
            "AutoMinorVersionUpgrade": true,
            "PubliclyAccessible": false,
            "DBClusterIdentifier": "sample-cluster",
            "StorageEncrypted": true,
            "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/wJalrXUtnFEMI/
K7MDENG/bPxRfiCYEXAMPLEKEY",
            "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
            "CACertificateIdentifier": "rds-ca-2019",
```

```
            "PromotionTier": 1,
            "DBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:sample-
instance-1",
            "EnabledCloudwatchLogsExports": [
                "profiler"
            ]
        }
    ]
}
```

変更が適用されるまでに数分かかります。インスタンスを使用できるのは、そのステータスが [available] である場合のみです。インスタンスのステータスは、 AWS Management Console または を使用してモニタリングできます AWS CLI。詳細については、「Amazon DocumentDB インスタンスのステータスのモニタリング」を参照してください。

Amazon DocumentDB インスタンスの再起動

ときどき、通常はメンテナンスのために、Amazon DocumentDB インスタンスを再起動する必要があります。クラスターに関連付けられたクラスターパラメータグループを変更するなど、特定の変更を行う場合は、変更を有効にするためにクラスター内のインスタンスを再起動する必要があります。AWS Management Console または を使用して、指定されたインスタンスを再起動できます AWS CLI。

インスタンスを再起動すると、データベースエンジンサービスが再起動されます。再起動により一時的に機能停止になります。その間、インスタンスのステータスは [rebooting] に設定されます。再起動が完了すると、Amazon DocumentDB イベントが作成されます。

インスタンスを再起動しても、フェイルオーバーは発生しません。Amazon DocumentDB クラスターをフェイルオーバーするには、 AWS Management Console または AWS CLI オペレーションを使用しますfailover-db-cluster。詳細については、「Amazon DocumentDB フェイルオーバー」を参照してください。

available (利用可能) 状態でないインスタンスを再起動することはできません。データベースは、以前の要求による変更、またはメンテナンスウィンドウのアクションなど、いくつかの理由で使用できない場合があります。インスタンスステータスの詳細については、「Amazon DocumentDB インスタンスのステータスのモニタリング」を参照してください。

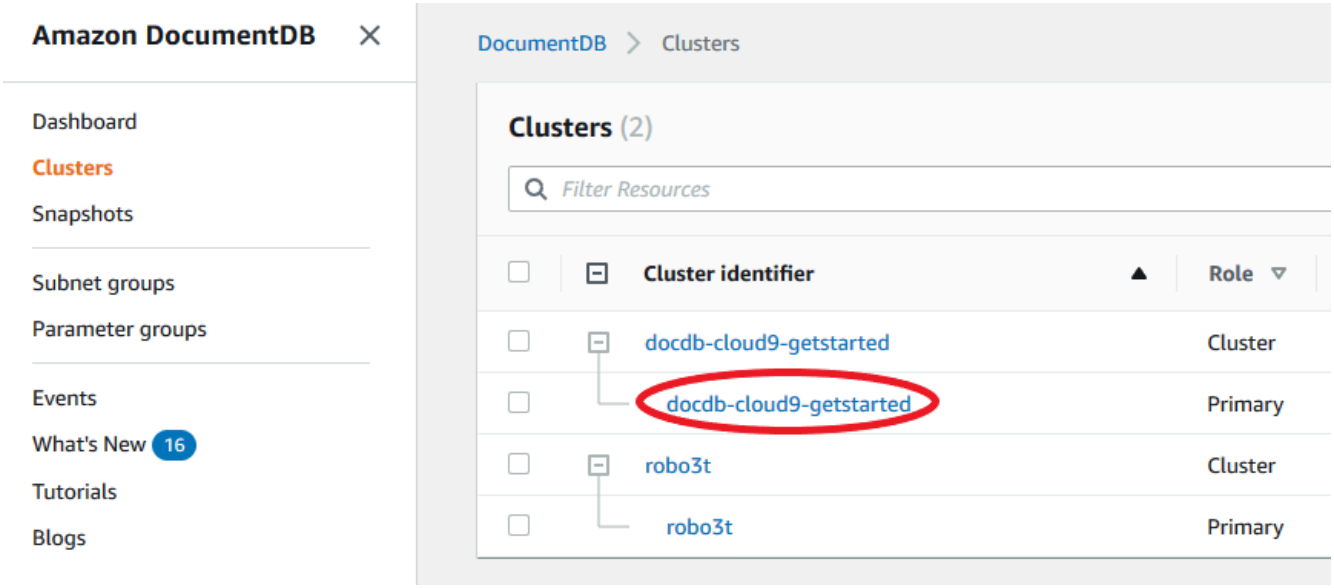
Using the AWS Management Console

次の手順では、コンソールを使用して指定するインスタンスを再起動します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. クラスターナビゲーションボックスに、[Cluster Identifier]の列が表示されます。インスタンスは、以下のスクリーンショットのように、クラスターの下に表示されます。



4. 再起動するインスタンスの左側にあるボックスをチェックします。
5. [Actions (アクション)]、[Reboot (再起動)] の順に選択し、[Reboot (再起動)] を選択して再起動を確認します。

インスタンスが再起動するまでには数分かかります。インスタンスを使用できるのは、そのステータスが [available] である場合のみです。コンソールまたは AWS CLIを使用して、インスタンスのステータスをモニタリングできます。詳細については、「Amazon DocumentDB インスタンスのステータスのモニタリング」を参照してください。

Using the AWS CLI

Amazon DocumentDB インスタンスを再起動するには、reboot-db-instance のパラメータとともに --db-instance-identifier のオペレーションを使用します。このパラメータは、再起動するインスタンスの識別子を指定します。

次のコードは、インスタンス sample-instance を再起動します。

Example

Linux、macOS、Unix の場合:

```
aws docdb reboot-db-instance \
       --db-instance-identifier sample-instance
```

Windows の場合:

```
aws docdb reboot-db-instance ^
       --db-instance-identifier sample-instance
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "sample-instance",
        "DBInstanceClass": "db.r5.large",
        "Engine": "docdb",
        "DBInstanceStatus": "rebooting",
        "Endpoint": {
            "Address": "sample-instance.node.us-east-1.docdb.amazonaws.com",
            "Port": 27017,
            "HostedZoneId": "ABCDEFGHIJKLM"
        },
        "InstanceCreateTime": "2020-03-27T08:05:56.314Z",
        "PreferredBackupWindow": "02:00-02:30",
        "BackupRetentionPeriod": 1,
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "AvailabilityZone": "us-east-1c",
```

```
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-abcd0123",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-abcd0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-wxyz0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1b"
                    },
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sun:06:53-sun:07:23",
        "PendingModifiedValues": {},
        "EngineVersion": "3.6.0",
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "DBClusterIdentifier": "sample-cluster",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "CACertificateIdentifier": "rds-ca-2019",
        "PromotionTier": 1,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-instance",
        "EnabledCloudwatchLogsExports": [
            "profiler"
        ]
    }
}
```

インスタンスが再起動するまでには数分かかります。インスタンスを使用できるのは、そのステータスが [available] である場合のみです。コンソールまたは AWS CLIを使用して、インスタン

スのステータスをモニタリングできます。詳細については、「Amazon DocumentDB インスタンスのステータスのモニタリング」を参照してください。

Amazon DocumentDB インスタンスの削除

Amazon DocumentDB インスタンスは、 AWS Management Console または を使用して削除できます AWS CLI。インスタンスを削除するには、インスタンスが使用可能な状態である必要があります。停止しているインスタンスを削除することはできません。インスタンスを含む Amazon DocumentDB クラスターが停止している場合は、最初にクラスターを起動し、インスタンスが 使用可能 になるまで待ってから、インスタンスを削除します。詳細については、「Amazon DocumentDB クラスターの停止と開始」を参照してください。

> **Note**
>
> Amazon DocumentDB は、クラスターボリューム内のすべてのデータを保存します。クラスターからすべてのインスタンスを削除しても、そのクラスターボリュームのデータは保持されます。データに再アクセスする必要がある場合は、いつでもクラスターにインスタンスを追加し、中断した個所から再開できます。

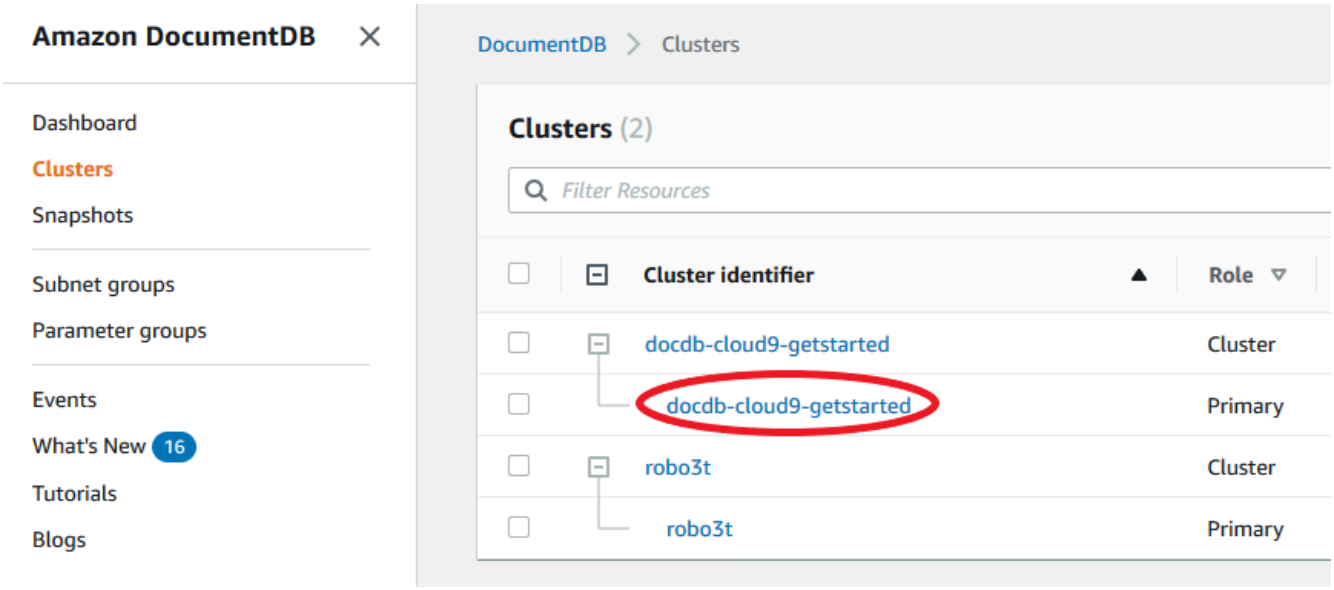
Using the AWS Management Console

次の手順では、コンソールを使用して指定した Amazon DocumentDB インスタンスを削除します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. クラスターナビゲーションボックスに、[Cluster Identifier]の列が表示されます。インスタンスは、以下のスクリーンショットのように、クラスターの下に表示されます。



4. 削除するインスタンスの左側にあるボックスをチェックします。
5. [Actions (アクション)] を選択して、[Delete (削除)] を選択します。

   1. クラスターの最後のインスタンスを削除する場合:

      - Create final cluster snapshot? (最終クラスタースナップショットを作成しますか?) - 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場合は、[はい] を選択します。それ以外の場合は [いいえ] を選択します。
      - Final snapshot name (最終スナップショット名) - 最終スナップショットを作成する場合は、作成した新しいクラスタースナップショットのクラスタースナップショットの識別子を入力します。
      - Delete <instance-name> instance? (<instance-name> インスタンスを削除しますか?) - 削除を確定するには、[delete entire cluster (クラスター全体を削除)] というフレーズをフィールドに入力します。

   2. クラスターの最後のインスタンスを削除しない場合:

      - Delete <instance-name> instance? (<instance-name> インスタンスを削除しますか?) - 削除を確認するフレーズ [delete me (これを削除する)] をフィールドに入力します。

6. [Delete (削除)] を選択して、インスタンスを削除します。

インスタンスが削除されるまでに数分かかります。インスタンスのステータスをモニタリングするには、「Amazon DocumentDB インスタンスのステータスのモニタリング」を参照してください。

Using the AWS CLI

次の手順では、 AWS CLIを使用して Amazon DocumentDB インスタンスを削除します。

1. 最初に、Amazon DocumentDB クラスター内のインスタンスの数を決定します。

   クラスター内のインスタンスの数を判断するには、以下のように `describe-db-clusters` コマンドを実行します。

   ```
   aws docdb describe-db-clusters \
       --db-cluster-identifier sample-cluster \
       --query 'DBClusters[*].
   [DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
   ```

   このオペレーションによる出力は、次のようになります。

   ```
   [
       [
           "sample-cluster",
           [
               "sample-instance-1",
               "sample-instance-2"
           ]
       ]
   ]
   ```

2. Amazon DocumentDB クラスターに複数のインスタンスがある場合。

   指定された Amazon DocumentDB インスタンスを削除するには、次に示すように `delete-db-instance` のパラメータを用いて `--db-instance-identifier` のコマンドを使用します。インスタンスが削除されるまでに数分かかります。インスタンスのステータスをモニタリングするには、「Amazon DocumentDB インスタンスのステータスのモニタリング」を参照してください。

   ```
   aws docdb delete-db-instance \
         --db-instance-identifier sample-instance-2
   ```

   このオペレーションによる出力は、次のようになります。

   ```
   {
       "DBInstance": {
   ```

```
        "DBInstanceIdentifier": "sample-instance-2",
        "DBInstanceClass": "db.r5.large",
        "Engine": "docdb",
        "DBInstanceStatus": "deleting",
        "Endpoint": {
            "Address": "sample-instance-2.node.us-east-1.docdb.amazonaws.com",
            "Port": 27017,
            "HostedZoneId": "ABCDEFGHIJKLM"
        },
        "InstanceCreateTime": "2020-03-27T08:05:56.314Z",
        "PreferredBackupWindow": "02:00-02:30",
        "BackupRetentionPeriod": 1,
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "AvailabilityZone": "us-east-1c",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-6242c31a",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-abcd0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-wxyz0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1b"
                    },
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sun:06:53-sun:07:23",
        "PendingModifiedValues": {},
        "EngineVersion": "3.6.0",
```

```
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "DBClusterIdentifier": "sample-cluster",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "CACertificateIdentifier": "rds-ca-2019",
        "PromotionTier": 1,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-instance-2",
        "EnabledCloudwatchLogsExports": [
            "profiler"
        ]
    }
}
```

3. 削除するインスタンスが Amazon DocumentDB クラスターの最後のインスタンスである場合。

   Amazon DocumentDB クラスター内の最後のインスタンスを削除すると、そのクラスターと、そのクラスターに関連付けられた自動スナップショットと継続バックアップも削除されます。

   クラスターの最後のインスタンスを削除するには、クラスターを削除し、オプションで最終スナップショットを作成します。詳細については、「Amazon DocumentDB クラスターの削除」を参照してください。

削除保護

Amazon DocumentDB クラスターの最後のインスタンスを削除すると、クラスターと、そのクラスターに関連付けられた自動スナップショットと継続バックアップも削除されます。Amazon DocumentDB は、 AWS Management Console または を使用して削除オペレーションを実行するかどうかにかかわらず、クラスターの削除保護を適用します AWS CLI。削除保護が有効になっている場合、クラスターを削除することはできません。

削除に対する保護が有効化されているクラスターを削除するには、まずクラスターを変更して削除保護を無効化する必要があります。詳細については、「Amazon DocumentDB クラスターの削除」を参照してください。

インスタンスクラスの管理

インスタンスクラスは、Amazon DocumentDB (MongoDB 互換) インスタンスの計算とメモリの容量を決定します。必要なインスタンスクラスは、処理能力とメモリの要件によって異なります。

Amazon DocumentDB は、インスタンスクラスの R4、R5、R6G、T3、および T4G ファミリーをサポートしています。これらのクラスは、メモリ消費の高いアプリケーション用に最適化された現行世代のインスタンスクラスです。これらのクラスの仕様については、インスタンスクラスの仕様を参照してください。

トピック

- インスタンスクラスの決定
- インスタンスクラスの変更
- リージョン別のサポートされるインスタンスクラス
- インスタンスクラスの仕様

インスタンスクラスの決定

インスタンスのクラスを決定するには、 AWS Management Console または `describe-db-instances` AWS CLI オペレーションを使用できます。

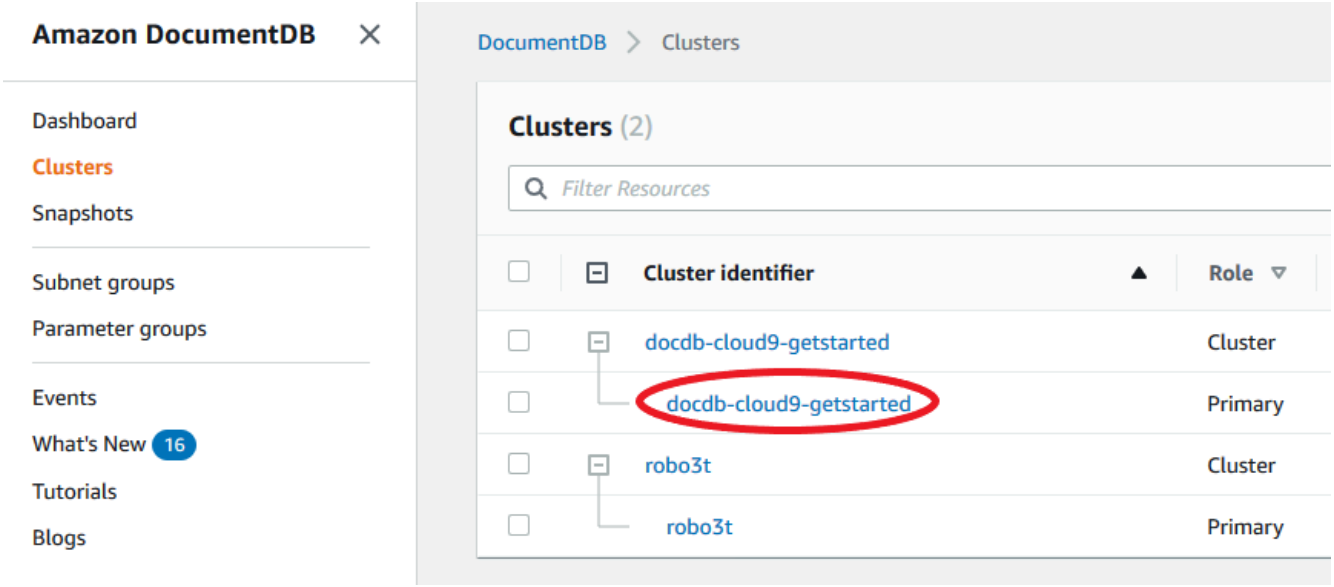
Using the AWS Management Console

クラスターのインスタンスのインスタンスクラスを決定するには、コンソールで次の手順を実行します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、対象とするインスタンスを見つけるために、クラスター を選択します。

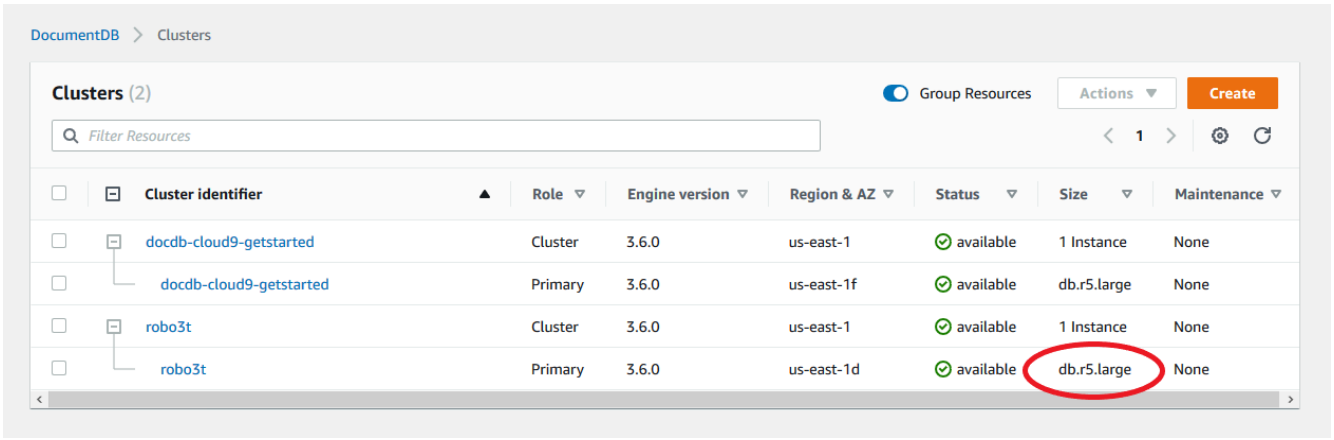
> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. クラスターナビゲーションボックスに、[Cluster Identifier]の列が表示されます。インスタンスは、以下のスクリーンショットのように、クラスターの下に表示されます。



4. インスタンスの一覧で、クラスターを展開して、関心のあるインスタンスを見つけます。目的のインスタンスを見つけます。次に、インスタンスの行の Size (サイズ) 列を確認し、インスタンスクラスを参照します。

   次の図では、インスタンス `robo3t` のインスタンスクラスは `db.r5.4xlarge` です。



Using the AWS CLI

を使用してインスタンスのクラスを確認するには AWS CLI、以下のパラメータを指定して `describe-db-instances`オペレーションを使用します。

- **--db-instance-identifier** - オプション。インスタンスクラスを検索するインスタンスを指定します。このパラメータを省略した場合、describe-db-instances は最大 100 個のインスタンスの説明を返します。
- **--query** - オプション。結果に含めるインスタンスのメンバーを指定します。このパラメータを省略すると、すべてのインスタンスメンバーが返されます。

Example

次の例では、インスタンス sample-instance-1 のインスタンス名とクラスを検索します。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
    --db-instance-identifier sample-instance-1
```

Windows の場合:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
    --db-instance-identifier sample-instance-1
```

このオペレーションによる出力は、次のようになります。

```
[
    [
        "sample-instance-1",
        "db.r5.large"
    ]
```

Example

次の例では、最大 100 個の Amazon DocumentDB インスタンスのインスタンス名とクラスを検索します。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
```

```
    --filter Name=engine,Values=docdb
```

Windows の場合:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
    --filter Name=engine,Values=docdb
```

このオペレーションによる出力は、次のようになります。

```
[
    [
        "sample-instance-1",
        "db.r5.large"
    ],
    [
        "sample-instance-2",
        "db.r5.large"
    ],
    [
        "sample-instance-3",
        "db.r5.4xlarge"
    ],
    [
        "sample-instance-4",
        "db.r5.4xlarge"
    ]
]
```

詳細については、「Amazon DocumentDB インスタンスの説明」を参照してください。

インスタンスクラスの変更

インスタンスのインスタンスクラスは、 AWS Management Console または を使用して変更できます AWS CLI。詳細については、「Amazon DocumentDB インスタンスの変更」を参照してください。

リージョン別のサポートされるインスタンスクラス

Amazon DocumentDB は、次の インスタンスクラスをサポートします。

- R6G— Arm ベースの AWS Graviton2 プロセッサを搭載したメモリ最適化インスタンスの最新世代。R5 インスタンスよりも最大 30% 優れたパフォーマンスを提供し、コストを 5% 削減します。
- R6GD— エフェメラルデータ用のローカル不揮発性メモリエクスプレス (NVMe) ベースのソリッドステートドライブ (SSD) ストレージを備えたメモリ最適化 R6G インスタンス。
- R5 - 同じインスタンスコストでR4インスタンスよりも最大 100% 優れたパフォーマンスを提供するメモリ最適化インスタンス。
- R4 - 前世代のメモリ最適化インスタンス。
- T4G— Arm ベースの AWS Graviton2 プロセッサを搭載した最新世代の低コストバースト可能な汎用インスタンスタイプ。ベースラインレベルの CPU パフォーマンスを提供し、T3 インスタンスよりも最大 35% 優れた価格パフォーマンスを実現し、CPU 使用率が中程度で使用量が一時的に急増するアプリケーションの実行に最適です。
- T3 - CPUパフォーマンスのベースラインレベルを提供し、必要な限りいつでもCPU使用率をバーストできる、低コストのバースト可能な汎用インスタンスタイプ。

インスタンスクラスの詳細な仕様については、「インスタンスクラスの仕様」を参照してください。

特定のインスタンスクラスは、特定のリージョンでサポートされる場合とサポートされない場合があります。次の表に、各リージョンで Amazon DocumentDB によってサポートされるインスタンスクラスを示してあります。

リージョン別のサポートされるインスタンスクラス

リージョン	R6GD	R6G	R5	R4	T4G	T3
米国東部 (オハイオ)	サポート	サポート対象	サポート対象	サポート対象	サポート対象	サポート
米国東部 (バージニア北部)	サポート	サポート対象	サポート対象	サポート対象	サポート対象	サポート

リージョン	R6GD	R6G	R5	R4	T4G	T3
米国西部 (オレゴン)	サポート	サポート対象	サポート対象	サポート対象	サポート対象	サポート
アフリカ (ケープタウン)		サポート	サポート対象		サポート対象	サポート
南米 (サンパウロ)	サポート	サポート対象	サポート対象		サポート対象	サポート
アジアパシフィック (香港)		サポート	サポート対象		サポート対象	サポート
アジアパシフィック (ハイデラバード)			サポート			サポート
アジアパシフィック (ムンバイ)	サポート	サポート対象	サポート対象		サポート対象	サポート
アジアパシフィック (ソウル)	サポート	サポート対象	サポート対象		サポート対象	サポート

リージョン	R6GD	R6G	R5	R4	T4G	T3
アジアパシフィック (シドニー)	サポート	サポート対象	サポート対象		サポート対象	サポート
アジアパシフィック (シンガポール)	サポート	サポート対象	サポート対象		サポート対象	サポート
アジアパシフィック (東京)	サポート	サポート対象	サポート対象		サポート対象	サポート
カナダ (中部)	サポート	サポート対象	サポート対象		サポート対象	サポート
欧州 (フランクフルト)	サポート	サポート対象	サポート対象		サポート対象	サポート
欧州 (アイルランド)	サポート	サポート対象	サポート対象	サポート対象	サポート対象	サポート
欧州 (ロンドン)	サポート	サポート対象	サポート対象		サポート対象	サポート

リージョン	R6GD	R6G	R5	R4	T4G	T3
欧州 (ミラノ)		サポート	サポート対象		サポート対象	サポート
欧州 (パリ)	サポート	サポート対象	サポート対象		サポート対象	サポート
欧州 (スペイン)		サポート	サポート対象		サポート対象	サポート
欧州 (ストックホルム)		サポート	サポート対象		サポート対象	サポート
中東 (アラブ首長国連邦)		サポート	サポート対象		サポート対象	サポート
中国 (北京)	サポート	サポート対象	サポート対象		サポート対象	サポート
中国 (寧夏)		サポート	サポート対象		サポート対象	サポート

リージョン	R6GD	R6G	R5	R4	T4G	T3
イスラエル (テルアビブ)		サポート	サポート対象		サポート対象	サポート
AWS GovCloud (米国西部)	サポート	サポート対象	サポート対象		サポート対象	サポート
AWS GovCloud (米国東部)	サポート	サポート対象	サポート対象		サポート対象	サポート

インスタンスクラスの仕様

次の表は、各クラスでサポートされているインスタンスタイプなど、Amazon DocumentDB インスタンスクラスの詳細を示しています。各テーブル列の説明については、表の下部を参照してください。

インスタンスクラス	vCPU[1]	メモリ (GiB)[2]	NVMe SSD 階層型キャッシュ (GiB)[3]	最大一時ストレージ (GiB)[4]	ベースライン/バースト帯域幅 (Gbps)[5]	使用エンジン[6]
R6G – Current Generation Memory-Optimized Instance Class based on Graviton2						
db.r6g.large	2	16	-	32	0.75 / 10	4.0.0、5.0.0
db.r6g.xlarge	4	32	-	63	1.25 / 10	4.0.0、5.0.0

インスタンスクラス	vCPU[1]	メモリ (GiB)[2]	NVMe SSD 階層型キャッシュ (GiB)[3]	最大一時ストレージ (GiB)[4]	ベースライン/バースト帯域幅 (Gbps)[5]	使用エンジン[6]
db.r6g.2xlarge	8	64	-	126	2.5 / 10	4.0.0、5.0.0
db.r6g.4xlarge	16	128	-	252	5.0 / 10	4.0.0、5.0.0
db.r6g.8xlarge	32	256	-	504	12	4.0.0、5.0.0
db.r6g.12xlarge	48	384	-	756	20	4.0.0、5.0.0
db.r6g.16xlarge	64	512	-	1008	25	4.0.0、5.0.0
R6GD – Graviton2 に基づく現行世代の NVMe-backed インスタンスクラス						
db.r6gd.xlarge	4	32	173	64	1.25 / 10	5.0.0 のみ
db.r6gd.2xlarge	8	64	346	128	2.5 / 10	5.0.0 のみ
db.r6gd.4xlarge	16	128	694	256	5.0 / 10	5.0.0 のみ
db.r6gd.8xlarge	32	256	1388	512	12	5.0.0 のみ
db.r6gd.12xlarge	48	384	2082	768	20	5.0.0 のみ
db.r6gd.16xlarge	64	512	2776	1024	25	5.0.0 のみ

インスタンスクラス	vCPU[1]	メモリ (GiB)[2]	NVMe SSD 階層型キャッシュ (GiB)[3]	最大一時ストレージ (GiB)[4]	ベースライン/バースト帯域幅 (Gbps)[5]	使用エンジン[6]
R5 – 前の世代のメモリ最適化インスタンスクラス						
db.r5.large	2	16	-	31	0.75 / 10	3.6.0、4.0.0、5.0.0
db.r5.xlarge	4	32	-	62	1.25 / 10	3.6.0、4.0.0、5.0.0
db.r5.2xlarge	8	64	-	124	2.5 / 10	3.6.0、4.0.0、5.0.0
db.r5.4xlarge	16	128	-	249	5.0 / 10	3.6.0、4.0.0、5.0.0
db.r5.8xlarge	32	256	-	504	10	3.6.0、4.0.0、5.0.0
db.r5.12xlarge	48	384	-	748	12	3.6.0、4.0.0、5.0.0
db.r5.16xlarge	64	512	-	1008	20	3.6.0、4.0.0、5.0.0
db.r5.24xlarge	96	768	-	1500	25	3.6.0、4.0.0、5.0.0
R4 – 前の世代のメモリ最適化インスタンスクラス						
db.r4.large	2	15.25	-	30	0.75 / 10	3.6.0 のみ
db.r4.xlarge	4	30.5	-	60	1.25 / 10	3.6.0 のみ

インスタンスクラス	vCPU[1]	メモリ (GiB)[2]	NVMe SSD 階層型キャッシュ (GiB)[3]	最大一時ストレージ (GiB)[4]	ベースライン/バースト帯域幅 (Gbps)[5]	使用エンジン[6]
db.r4.2xlarge	8	61	-	120	2.5 / 10	3.6.0 のみ
db.r4.4xlarge	16	122	-	240	5.0 /10	3.6.0 のみ
db.r4.8xlarge	32	244	-	480	10	3.6.0 のみ
db.r4.16xlarge	64	488	-	960	25	3.6.0 のみ
T4G – Latest Generation Burstable Performance Instance Classes based on Graviton2						
db.t4g.medium	2	4	-	8.13	0.256 / 5	4.0.0、5.0.0
T3 – 最新世代のバースト可能なパフォーマンスインスタンスクラス						
db.t3.medium	2	4	-	7.5	0.256 / 5	3.6.0、4.0.0、5.0.0

インスタンスクラス	vCPU[1]	メモリ (GiB)[2]	NVMe SSD 階層型キャッシュ (GiB)[3]	最大一時ストレージ (GiB)[4]	ベースライン/バースト帯域幅 (Gbps)[5]	使用エンジン[6]

1. vCPU - 仮想中央演算装置 (CPU) の数。仮想 CPU は、インスタンスクラスの比較に使用できる容量の単位です。特定のプロセッサを購入またはリースして数か月から数年間使用する代わりに、時間単位で処理能力をレンタルすることができます。当社の目的は、実際に利用しているハードウェアがどのようなものであろうと、一定量の CPU 能力を安定したかたちで提供することです。
2. メモリ (GiB) - インスタンスに割り当てられた RAM (ギガバイト単位)。通常、メモリと vCPU の比率は一定です。
3. NVMe SSD 階層型キャッシュ — エフェメラルデータを保存するための拡張キャッシュとして割り当てられた、ギガバイト単位で測定される SSD ボリューム上のスペース。このキャッシュは NVMe-backed インスタンスでのみ使用できます。
4. 最大一時ストレージ (GiB) — 非永続的な一時ファイルストレージ用にインスタンスに割り当てられる、ギガバイト単位で測定されるスペース。NVMe-backed インスタンスの場合、このストレージは NVMe ベースの SSD ボリュームでホストされます。他のすべてのインスタンスでは、Amazon Elastic Block Store (EBS) でホストされます。
5. ベースライン/バースト帯域幅 (Gbps) — バースト帯域幅は、1 秒あたりの最大帯域幅をギガビット単位で表します。8 で除算して、1 秒あたりの予想スループットをギガバイト単位で取得します。サイズ 4xlarge 以下のインスタンスにはベースライン帯域幅があります。追加需要を満たすために、ネットワーク I/O クレジットメカニズムを使用して、ベースライン帯域幅を超えてバーストすることができます。インスタンスはインスタンスのサイズに応じて、バースト帯域幅を限られた期間 (通常は 5～60 分) 使用できます。
6. 使用エンジン - インスタンスクラスをサポートする Amazon DocumentDB エンジン。

NVMe-backed インスタンス

NVMe-backed インスタンスは、通常のインスタンスのメモリを超える大規模なデータセットを持つアプリケーションに対して、クエリパフォーマンスを最大 7 倍高速化します。これらのインスタンスは、r6gd インスタンスで利用可能なローカルの不揮発性メモリエクスプレス (NVMe) ベースのソリッドステートドライブ (SSD) ストレージを活用してエフェメラルデータを保存し、ネットワークベースのストレージアクセスを減らし、読み取りレイテンシーとスループットを向上させます。

ローカル SSD スペースは 2 つのセクションに分かれています。

- 階層型キャッシュ — ローカル SSD の約 73% がデータベースキャッシュとして割り当てられるため、システムはメインメモリのみの最大 5 倍のデータベースページを保存できます。ローカル SSD は 2 層キャッシュとして機能し、既存のメモリ内バッファキャッシュは 1 層キャッシュのままです。クエリは、バッファキャッシュと SSD キャッシュの両方でミスが発生した場合にのみ外部ストレージにアクセスします。
- 一時ストレージ — 残りの 27% は非永続的な一時ファイルストレージ用に予約されており、ソートやインデックス構築などのリソース集約型オペレーションを含む複雑なクエリに使用されます。通常のインスタンスでは、一時スペースは Amazon Elastic Block Store (EBS) ボリュームにあります。SSD 上のローカルにホストされた一時ストレージは、ソートを伴うクエリレイテンシーを最大 2 倍短縮し、インデックス構築などのリソース集約型のオペレーションを高速化します。

サポートされる NVMe-backed インスタンスのタイプとその仕様の詳細については、「」を参照してくださいインスタンスクラスの仕様。

トピック

- NVMe-backed インスタンスの推奨ユースケース
- Amazon DocumentDB での NVMe-backed インスタンスの使用
- NVMe-backed インスタンスのモニタリング

NVMe-backed インスタンスの推奨ユースケース

次のシナリオでは、NVMe-backed インスタンスを使用することをお勧めします。

- 読み取り負荷の高いワークロード — ワークロードが読み取り負荷が高く、データセットがバッファキャッシュよりも大きく、`ReadIOPS`メトリクスが低`BufferCacheHitRatio`および高である場合、NVMe-backed インスタンスはパフォーマンス上の利点を提供します。
- 更新負荷の高いワークロード — ワークロードが更新負荷が高く、ネットワークストレージの読み取りレイテンシーが原因でガベージコレクションが追いつくことができない場合、NVMe-backed インスタンスが問題の軽減に役立ちます。

NVMe-backed インスタンスは、次のようなさまざまなユースケースにメリットがあります。

- インターネット規模のアプリケーション — 厳格なパフォーマンスのサービスレベルアグリーメント (SLAs) による支払い処理、請求、e コマースなどのアプリケーションは、NVMe-backed インスタンスのパフォーマンス上の利点を活用できます。
- リアルタイムレポートダッシュボード — メトリクス/データ収集のために数百のクエリを実行するダッシュボードは、NVMe-backed インスタンスの低レイテンシーと高スループットからメリットを得られます。
- 生成 AI アプリケーション — ベクトル検索を使用して数百万のベクトル埋め込みの完全または最も近い近傍を検索するアプリケーションは、NVMe-backed インスタンスのハイパフォーマンスを活用できます。

Amazon DocumentDB での NVMe-backed インスタンスの使用

Amazon DocumentDB の NVMe-backed インスタンスを使用するには：

- Amazon DocumentDB クラスターを作成し、NVMe-backed インスタンスクラスの 1 つを追加します。詳細については、「Amazon DocumentDB クラスターの作成」を参照してください。
- または、NVMe-backed インスタンスクラスの 1 つを使用するように既存の Amazon DocumentDB クラスターを変更します。詳細については、「Amazon DocumentDB クラスターの変更」を参照してください。

異なる AWS リージョン間で NVMe-backed インスタンスの可用性を確認するには、「」を参照してくださいリージョン別のサポートされるインスタンスクラス。

NVMe-backed インスタンスから通常のインスタンスに戻す場合は、Amazon DocumentDB インスタンスのデータベースインスタンスクラスを、NVMe ストレージのない同様のインスタンスクラスに変更します。例えば、現在のインスタンスクラスが「db.r6gd.4xlarge」の場合、「db.r6g.4xlarge」を選択して元に戻します。詳細については、「Amazon DocumentDB クラスターの変更」を参照してください。

NVMe-backed インスタンスのモニタリング

Amazon CloudWatch で利用可能な通常のインスタンスメトリクスに加えて、NVMe-backed インスタンスは NVMe ベースの SSD ストレージ、IOPS、スループットに固有の追加のメトリクスを出力します。

```
NVMeStorageCacheHitRatio
FreeNVMeStorage
ReadIOPSNVMeStorage
```

```
ReadLatencyNVMeStorage
ReadThroughputNVMeStorage
WriteIOPSNVMeStorage
WriteLatencyNVMeStorage
WriteThroughputNVMeStorage
```

これらのメトリクスの詳細については、「」を参照してください。 NVMe-backed インスタンスメトリクス

Amazon DocumentDB サブネットグループの管理

Virtual Private Cloud (VPC) は、 専用の仮想ネットワークです AWS アカウント。クラウド内の他の仮想ネットワークから論理的に分離されています AWS 。Amazon DocumentDB クラスターなどの AWS リソースを Amazon VPC で起動できます。VPC の IP アドレス範囲を指定して、サブネットを追加し、セキュリティグループを関連付けて、ルートテーブルを設定できます。

サブネットは、Amazon VPC の IP アドレスの範囲です。 AWS リソースは、指定されたサブネットで起動できます。インターネットに接続する必要があるリソースには、パブリックサブネットを使用します。インターネットに接続する必要がないリソースには、プライベートサブネットを使用します。公開サブネットとプライベートサブネットの詳細については、Amazon Virtual PrivateCloud ユーザーガイド の VPCとサブネットの基本 を参照してください。

DB サブネットグループは VPC に作成するサブネットのコレクションで、クラスター用に指定します。サブネットグループでは、クラスターの作成時に特定の VPC を指定することができます。`default` サブネットグループを使用すると、VPC 内のすべてのサブネットにまたがります。

各 DB サブネットグループには、特定の リージョン内の少なくとも 2 つのアベイラビリティーゾーンにサブネットが必要です。VPC に DB インスタンスを作成するときに、DB サブネットグループを選択する必要があります。Amazon DocumentDB は、その DB サブネットグループと優先アベイラビリティーゾーンを使用して、クラスターに関連付けるサブネットとそのサブネット内の IP アドレスを選択します。プライマリインスタンスに障害が発生した場合、Amazon DocumentDB は対応するレプリカインスタンスを新しいプライマリに昇格させることができます。その後、前のプライマリが配置されていたサブネットの IP アドレスを使用して、新しいレプリカインスタンスを作成します。

Amazon DocumentDB が VPC にインスタンスを作成すると、DB サブネットグループから選択された IP アドレスを使用して、クラスターにネットワークインターフェースが割り当てられます。基になる IP アドレスはフェイルオーバー中に変わる可能性があるため、DNS名 を使用することを強くお勧めします。詳細については、「Amazon DocumentDB エンドポイント」を参照してください。

独自の VPC とサブネットの作成については、Amazon Virtual PrivateCloud ユーザーガイド の VPC とサブネットの操作 を参照してください。

トピック

- Amazon DocumentDB サブネットグループの作成
- Amazon DocumentDB サブネットグループの説明
- Amazon DocumentDB サブネットグループの変更
- Amazon DocumentDB サブネットグループの削除

Amazon DocumentDB サブネットグループの作成

Amazon DocumentDB クラスターを作成する場合、クラスターを起動するには、Amazon VPC とその Amazon VPC 内の対応するサブネットグループを選択する必要があります。サブネットは、インスタンスの起動に使用するアベイラビリティーゾーンとアベイラビリティーゾーン内の IP 範囲を決定します。

サブネットグループは、名前付きのサブネット (またはAZ) のセットであり、Amazon DocumentDB インスタンスの起動に使用するアベイラビリティーゾーンを指定できます。たとえば、3 つのインスタンスを含むクラスターでは、各インスタンスを別々の AZ にプロビジョニングして高可用性を実現することをお勧めします。これにより、1 つの AZ に障害が発生した場合、1 つのインスタンスにのみ影響します。

Amazon DocumentDB インスタンスは現在、最大 3 つの AZ にプロビジョニングできます。サブネットグループに 3 つ以上のサブネットがある場合でも、それらのサブネットのうちの 3 つのみを使用して Amazon DocumentDB クラスターを作成できます。したがって、サブネットグループを作成するときは、インスタンスをデプロイする 3 つのサブネットのみを選択することをお勧めします。

例：クラスターが作成され、Amazon DocumentDB は AZ {1A、1B、および1C} を選択します。この場合、AZ 1D でインスタンスを作成しようとすると、その API コールは失敗します。ただし、特定の AZ を指定せずにインスタンスを作成した場合、Amazon DocumentDB がお客様に代わって AZ を選択します。Amazon DocumentDB は、アルゴリズムを使用して AZ 間でインスタンスの負荷を分散し、高可用性を実現できるようにします。たとえば、3 つのインスタンスがプロビジョニングされる場合、デフォルトでは 3 つの AZ にプロビジョニングされ、1 つの AZ にまとめてプロビジョニングされません。

ベストプラクティス

- 特に理由がない限り、常に 3 つのサブネットを含むサブネットグループを作成します。これにより、3 つ以上のインスタンスを持つクラスターでは、インスタンスが 3 つの AZ にまたがってプロビジョニングされるため、より高い可用性を実現できます。
- 高可用性を実現するには、常に複数の AZ にインスタンスを分散させます。クラスターのすべてのインスタンスを 1 つの AZ にまとめて配置しないでください。
- フェイルオーバーイベントはいつでも発生する可能性があるため、プライマリインスタンスまたはレプリカインスタンスが常に特定の AZ に存在すると想定しないでください。

サブネットグループの作成方法

AWS Management Console または を使用して AWS CLI 、Amazon DocumentDB サブネットグループを作成できます。

Using the AWS Management Console

Amazon DocumentDB サブネットグループを作成するには、次のステップを使用します。

Amazon DocumentDB サブネットグループの作成方法

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[Subnet groups (サブネットグループ)]、[Create (作成)] の順に選択します。

   > **Tip**
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. [Create subnet group (サブネットグループを作成する)] ページで:

   a. [Subnet group details (サブネットグループの詳細)] セクションで:

      i. 名前 - サブネットグループ用の意味のある名前を入力します。

      ii. [Description] (説明) - サブネットグループの説明を入力します。

   b. [Add subnets (サブネットを追加する)] セクションで:

i. VPC — リストからこのサブネットグループの VPC を選択します。

ii. 次のいずれかを行います：

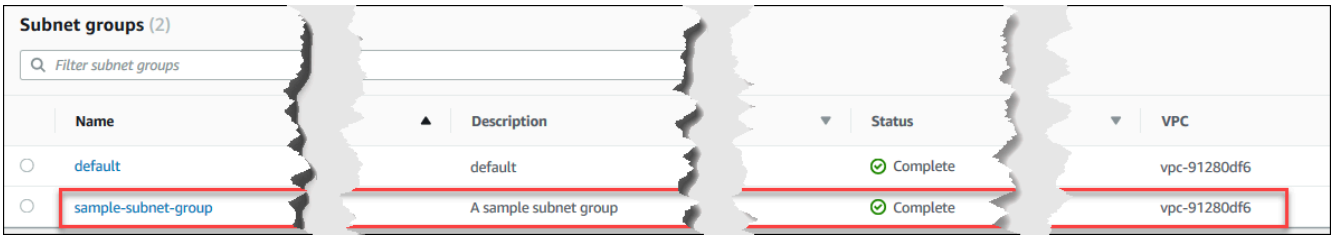
- 選択した VPC にすべてのサブネットを含めるには、[Add all the subnets related to this VPC (この VPC に関連するすべてのサブネットを追加)] を選択します。
- このサブネットグループのサブネットを指定するには、サブネットを含める各アベイラビリティーゾーンに対して次の手順を実行します。少なくとも 2 つのアベイラビリティーゾーンを含める必要があります。

A. アベイラビリティーゾーン - リストからアベイラビリティーゾーンを選択します。

B. サブネット — リストで、このサブネットグループに選択したアベイラビリティーゾーンからサブネットを選択します。

C. [Add subnet (サブネットを追加する)] を選択します。

4. [Create] (作成) を選択します。サブネットグループが作成されると、他のサブネットグループとともに表示されます。



Using the AWS CLI

を使用してサブネットグループを作成する前に AWS CLI、最初に使用可能なサブネットを決定する必要があります。次の AWS CLI オペレーションを実行して、アベイラビリティーゾーンとそのサブネットを一覧表示します。

[パラメータ:]

- **`--db-subnet-group`** - オプションです。特定のサブネットグループを指定すると、そのグループのアベイラビリティーゾーンとサブネットが一覧表示されます。このパラメータを省略すると、すべてのサブネットグループのアベイラビリティーゾーンとサブネットが一覧表示されます。`default` サブネットグループを指定すると、すべての VPC のサブネットが一覧表示されます。

Example

Linux、macOS、Unix の場合:

```
aws docdb describe-db-subnet-groups \
    --db-subnet-group-name default \
    --query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

Windows の場合:

```
aws docdb describe-db-subnet-groups ^
    --db-subnet-group-name default ^
    --query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
[
    [
        "default",
        [
            [
                "us-east-1a",
                "subnet-4e26d263"
            ],
            [
                "us-east-1c",
                "subnet-afc329f4"
            ],
            [
                "us-east-1e",
                "subnet-b3806e8f"
            ],
            [
                "us-east-1d",
                "subnet-53ab3636"
            ],
            [
                "us-east-1b",
                "subnet-991cb8d0"
            ],
            [
```

```
                "us-east-1f",
                "subnet-29ab1025"
            ]
        ]
    ]
]
```

前のオペレーションからの出力を使用して、新しいサブネットグループを作成できます。新しいサブネットグループは、少なくとも 2 つのアベイラビリティーゾーンからのサブネットを含める必要があります。

[パラメータ:]

- **--db-subnet-group-name** — 必須 このサブネットグループの名前。
- **--db-subnet-group-description** — 必須 このサブネットグループの説明。
- **--subnet-ids** — 必須 このサブネットグループに含めるサブネットのリスト。例えば、subnet-53ab3636 などです。
- --タグ - オプションです。このサブネットグループにアタッチするタグ (キーと値のペア) のリスト。

次のコードは、subnet-4e26d263、subnet-afc329f4、および subnet-b3806e8f の 3 つのサブネットを持つサブネットグループ sample-subnet-group を作成します。

Linux、macOS、Unix の場合:

```
aws docdb create-db-subnet-group \
    --db-subnet-group-name sample-subnet-group \
    --db-subnet-group-description "A sample subnet group" \
    --subnet-ids subnet-4e26d263 subnet-afc329f4 subnet-b3806e8f \
    --tags Key=tag1,Value=One Key=tag2,Value=2
```

Windows の場合:

```
aws docdb create-db-subnet-group ^
    --db-subnet-group-name sample-subnet-group ^
    --db-subnet-group-description "A sample subnet group" ^
    --subnet-ids subnet-4e26d263 subnet-afc329f4 subnet-b3806e8f ^
    --tags Key=tag1,Value=One Key=tag2,Value=2
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "DBSubnetGroup": {
        "DBSubnetGroupDescription": "A sample subnet group",
        "DBSubnetGroupName": "sample-subnet-group",
        "Subnets": [
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1a"
                },
                "SubnetIdentifier": "subnet-4e26d263",
                "SubnetStatus": "Active"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1c"
                },
                "SubnetIdentifier": "subnet-afc329f4",
                "SubnetStatus": "Active"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1e"
                },
                "SubnetIdentifier": "subnet-b3806e8f",
                "SubnetStatus": "Active"
            }
        ],
        "VpcId": "vpc-91280df6",
        "DBSubnetGroupArn": "arn:aws:rds:us-east-1:123SAMPLE012:subgrp:sample-
subnet-group",
        "SubnetGroupStatus": "Complete"
    }
}
```

Amazon DocumentDB サブネットグループの説明

AWS Management Console または を使用して AWS CLI 、Amazon DocumentDB サブネットグループの詳細を取得できます。

Using the AWS Management Console

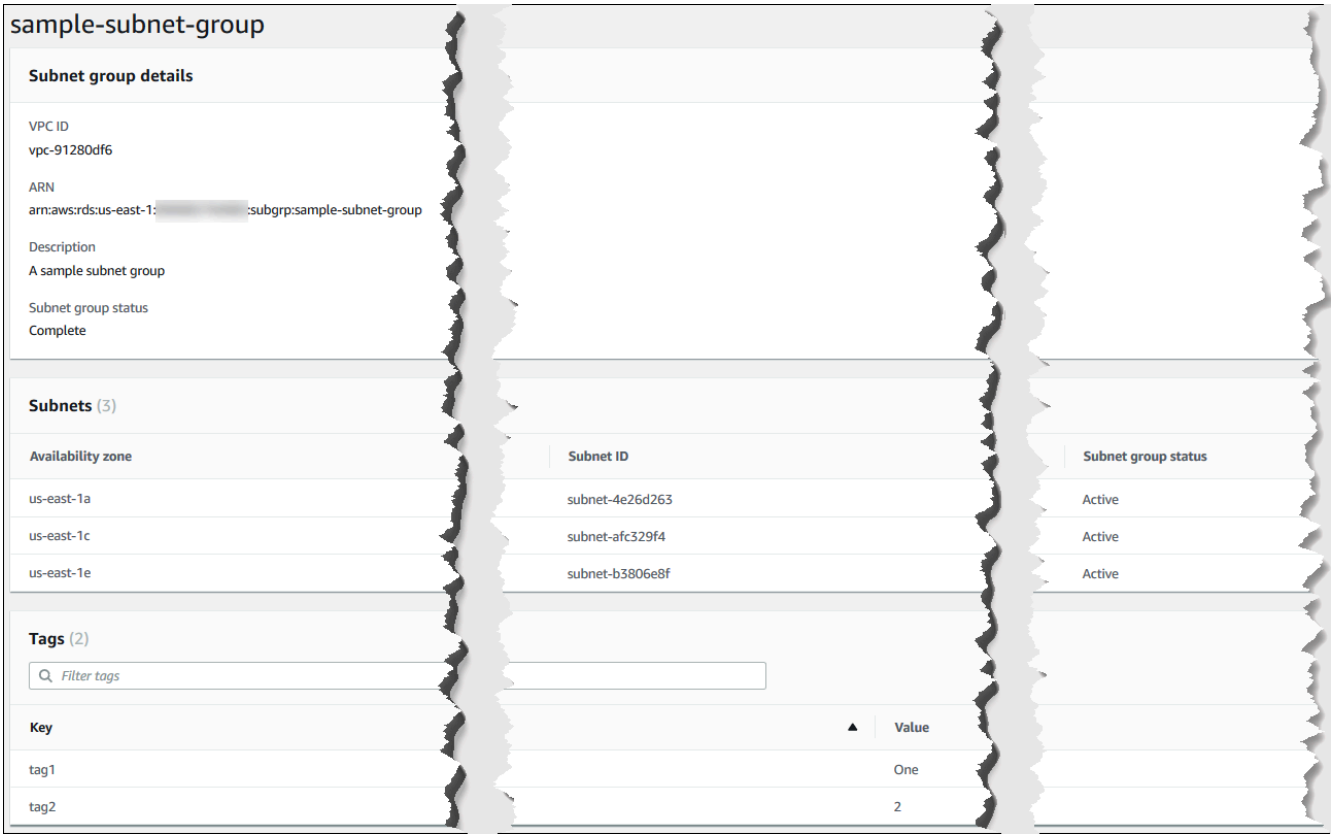
次の手順では、Amazon DocumentDB サブネットグループの詳細を取得する方法を説明します。

サブネットグループの詳細を確認するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [サブネットグループ] を選択します。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. サブネットグループの詳細を表示するには、そのサブネットグループの名前を選択します。



Using the AWS CLI

Amazon DocumentDB サブネットグループの詳細を見つけるには、次のパラメーターを指定した `describe-db-subnet-groups` オペレーションを使用します。

パラメータ

- `--db-subnet=group-name` - オプションです。含まれている場合は、名前付きサブネットグループの詳細が一覧表示されます。省略すると、最大 100 個のサブネットグループの詳細が一覧表示されます。

Example

次のコードは、Amazon DocumentDB サブネットグループの作成 セクションで作成した `sample-subnet-group` サブネットグループの詳細を示しています。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-subnet-groups \
    --db-subnet-group-name sample-subnet-group
```

Windows の場合:

```
aws docdb describe-db-subnet-groups ^
    --db-subnet-group-name sample-subnet-group
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "DBSubnetGroup": {
        "DBSubnetGroupArn": "arn:aws:rds:us-east-1:123SAMPLE012:subgrp:sample-
subnet-group",
        "VpcId": "vpc-91280df6",
        "SubnetGroupStatus": "Complete",
        "DBSubnetGroupName": "sample-subnet-group",
        "Subnets": [
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1a"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-4e26d263"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1c"
```

```
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-afc329f4"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1e"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-b3806e8f"
            }
        ],
        "DBSubnetGroupDescription": "A sample subnet group"
    }
}
```

Amazon DocumentDB サブネットグループの変更

AWS Management Console または を使用して AWS CLI 、サブネットグループの説明を変更したり、Amazon DocumentDB サブネットグループからサブネットを追加または削除したりできます。ただし、default サブネットグループは変更できません。

Using the AWS Management Console

を使用して AWS Management Console 、サブネットグループの説明を変更したり、サブネットを追加または削除したりできます。完了したら、サブネットグループに少なくとも 2 つのアベイラビリティーゾーンが関連付けられている必要があります。

サブネットグループを変更するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [サブネットグループ] を選択します。次に、サブネットグループ名の左側にあるボタンを選択します。default サブネットグループは変更できないことに注意してください。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン

(≡)
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. 説明 — サブネットグループの説明を変更するには、新しい説明を入力します。
5. サブネットグループに関連付けられているサブネットを変更するには、[Add subnets (サブネットの追加)] セクションで、次のいずれかを実行します。
   - このサブネットグループからサブネットをすべて削除するには、[Remove all (すべてを削除)] を選択します。
   - このサブネットグループから特定のサブネットを削除するには、削除するサブネットごとに [Remove (削除) を選択します。
   - この VPC に関連付けられているすべてのサブネットを追加するには、[Add all the subnets related to this VPC (このVPCに関連付けられているすべてのサブネットを追加])] を選択します。
   - このサブネットグループに特定のサブネットを追加するには、サブネットを追加する各アベイラビリティーゾーンに対して次の手順を実行します。
     a. アベイラビリティーゾーン - リストから新しいアベイラビリティーゾーンを選択します。
     b. サブネット — リストで、このサブネットグループに選択したアベイラビリティーゾーンからサブネットを選択します。
     c. [Add subnet (サブネットを追加する)] を選択します。
6. 確認ダイアログボックスでは:
   - サブネットグループにこのような変更を行うには、[Modify (変更)] を選択します。
   - サブネットグループを変更しない場合は、[キャンセル] を選択してください。

Using the AWS CLI

を使用して AWS CLI 、サブネットグループの説明を変更したり、サブネットを追加または削除したりできます。完了したら、サブネットグループに少なくとも 2 つのアベイラビリティーゾーンが関連付けられている必要があります。`default` サブネットグループを変更することはできません。

[パラメータ:]

- `--db-subnet-group-name` — 必須 変更している Amazon DocumentDB サブネットグループの名前。
- `--subnet-ids` — 必須 この変更が完了した後でサブネットグループに必要となるすべてのサブネットの一覧。

> **Important**
>
> このリストに含まれていない、現在サブネットグループにあるサブネットはすべて、サブネットグループから削除されます。現在サブネットグループにあるサブネットのいずれかを保持したい場合は、それらをこのリストに含める必要があります。

- `--db-subnet-group-description` - オプションです。サブネットグループの説明。

Example

次のコードは説明を変更し、既存のサブネットをサブネット `subnet-991cb8d0`、`subnet-53ab3636`、および `subnet-29ab1025` に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb modify-db-subnet-group \
    --db-subnet-group-name sample-subnet-group \
    --subnet-ids subnet-991cb8d0 subnet-53ab3636 subnet-29ab1025 \
    --db-subnet-group-description "Modified subnet group"
```

Windows の場合:

```
aws docdb modify-db-subnet-group ^
    --db-subnet-group-name sample-subnet-group ^
    --subnet-ids subnet-991cb8d0 subnet-53ab3636 subnet-29ab1025 ^
    --db-subnet-group-description "Modified subnet group"
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。これは、「Amazon DocumentDB サブネットグループの作成」セクションで作成したものと同じサブネットグループであることに注目します。ただし、サブネットグループ内のサブネットは、`modify-db-subnet-group` オペレーションで一覧表示されるものに置き換えられます。

```
{
    "DBSubnetGroup": {
        "DBSubnetGroupArn": "arn:aws:rds:us-east-1:123SAMPLE012:subgrp:sample-
subnet-group",
        "DBSubnetGroupDescription": "Modified subnet group",
        "SubnetGroupStatus": "Complete",
        "Subnets": [
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1d"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-53ab3636"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1b"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-991cb8d0"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1f"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-29ab1025"
            }
        ],
        "VpcId": "vpc-91280df6",
        "DBSubnetGroupName": "sample-subnet-group"
    }
}
```

Amazon DocumentDB サブネットグループの削除

AWS Management Console または を使用して AWS CLI 、Amazon DocumentDB サブネットグループを削除できます。ただし、`default` サブネットグループは削除できません。

Using the AWS Management Console

を使用してサブネットグループ AWS Management Console を削除できます。ただし、 `default` サブネットグループを削除することはできません。

サブネットグループを削除するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [サブネットグループ] を選択します。次に、サブネットグループ名の左側にあるボタンを選択します。`default` サブネットグループは削除できないことに注意してください。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. [アクション] を選択し、[削除] を選択します。
4. 確認ダイアログボックスでは:
   - サブネットグループを削除するには、[削除] を選択します。
   - 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。

Using the AWS CLI

を使用して Amazon DocumentDB サブネットグループを削除するには AWS CLI、次のパラメータを指定して `delete-db-subnet-group`オペレーションを使用します。

パラメータ

- `--db-subnet-group-name` — 必須 削除する Amazon DocumentDB サブネットグループの名前。`default` サブネットグループは削除できないことに注意してください。

Example

次のコードは、`sample-subnet-group` を削除します。

Linux、macOS、Unix の場合:

```
aws docdb delete-db-subnet-group \
    --db-subnet-group-name sample-subnet-group
```

Windows の場合:

```
aws docdb delete-db-subnet-group ^
    --db-subnet-group-name sample-subnet-group
```

このオペレーションでは何も出力されません。

Amazon DocumentDB 高可用性とレプリケーション

レプリカインスタンスを使用すると、Amazon DocumentDB で高可用性と読み取りスケールインを実現できます。1 つの Amazon DocumentDB クラスターは、1 つのプライマリインスタンスと 15 までのレプリカインスタンスをサポートします。これらのインスタンスは、クラスターのリージョン内のアベイラビリティーゾーンに分散することができます。プライマリインスタンスでは、読み込みと書き込みトラフィックを受け入れ、レプリカインスタンスは読み込みリクエストのみを受け入れます。

クラスターボリュームはクラスターのデータの複数のコピーで構成されます。ただし、クラスターボリューム内のデータは、プライマリインスタンスおよびクラスター内の Amazon DocumentDB レプリカに対する単一の論理ボリュームとして表されます。レプリカインスタンスには、結果整合性があります。これによって、最短のレプリカラグでクエリ結果を返します。通常の場合、プライマリインスタンスが更新を書き込みしてから 100 ミリ秒未満になります。レプリカラグは、データベースの変更レートによって異なります。つまり、データベースに対して大量の書き込みオペレーションが発生している間、レプリカラグが増加することがあります。

読み取りのスケーリング

Amazon DocumentDB プリカは、クラスターボリュームでの読み取りオペレーションに特化しているため、読み取りのスケーリングに最適です。書き込みオペレーションはプライマリインスタンスによって管理されます。クラスターボリュームは、クラスター内のすべてのインスタンス間で共有されます。したがって、各 Amazon DocumentDB レプリカごとにデータのコピーを複製して維持する必要はありません。

高可用性

Amazon DocumentDB クラスターを作成するときは、サブネットグループ内のアベイラビリティーゾーン の数に応じて (少なくとも 2 つ存在する必要があります)、Amazon DocumentDB は、アベイラビリティゾーン間でインスタンスをプロビジョニングします。クラスター内でインスタンスを作成する場合、Amazon DocumentDB はサブネットグループ内のアベイラビリティゾーン間でインスタンスを自動的に配信して、クラスターのバランスを取ります。また、このアクションは、すべてのインスタンスが同じアベイラビリティーゾーンに配置されることを回避します。

例

この点を説明するために、3 つのアベイラビリティゾーン (AZ1、AZ2、および AZ3) を持つサブネットグループを持つクラスターを作成する例を考えます。

クラスター内の最初のインスタンスが作成されると、それがプライマリインスタンスとなり、いずれかのアベイラビリティゾーンに配置されます。この例では、これが AZ1 です。作成される 2 番目のインスタンスはレプリカインスタンスで、他の 2 つのアベイラビリティゾーン (AZ2 とします) のいずれかにあります。作成される 3 番目のインスタンスはレプリカインスタンスで、残りのアベイラビリティゾーン (AZ3) にあります。さらにインスタンスを作成する場合、クラスター内でバランスが取れるようにアベイラビリティゾーン間で分散します。

プライマリインスタンス (AZ1) で障害が発生すると、フェイルオーバーがトリガーされ、既存のレプリカの 1 つがプライマリに昇格します。古いプライマリティが復旧すると、プロビジョニングされていたアベイラビリティゾーン (AZ1) と同じ場所でレプリカとなります。3 つのインスタンスクラスターをプロビジョニングすると、Amazon DocumentDB はその 3 つのインスタンスクラスターを引き続き保持します。Amazon DocumentDB は、手動による介入なしに、インスタンス障害の検出、フェイルオーバー、および回復を自動的に処理します。

Amazon DocumentDB がフェイルオーバーを実行してインスタンスを復旧すると、復旧したインスタンスはプロビジョニングされていたアベイラビリティーゾーンに残ります。ただし、インスタンスのロールはプライマリからレプリカに変更される場合があります。これを実行することで、一連のフェイルオーバーによってすべてのインスタンスが結果として同じアベイラビリティゾーンになるというシナリオを回避することができます。

フェイルオーバーターゲットとして Amazon DocumentDB レプリカを指定できます。つまり、プライマリインスタンスが失敗した場合、指定された Amazon DocumentDB プリカまたは層からのレプリカがプライマリインスタンスに昇格します。短い中断があり、その間はプライマリインスタンスに対して行われた読み取りおよび書き込みリクエストは、例外により失敗します。Amazon DocumentDB クラスターに Amazon DocumentDB レプリカが含まれていない場合は、プライマリ

インスタンスに障害が発生すると再作成されます。Amazon DocumentDB レプリカを昇格するほうが、プライマリインスタンスの再作成よりも大幅に短時間で行えます。

高可用性のシナリオでは、1 つ以上の Amazon DocumentDB レプリカを作成することをお勧めします。これらのレプリカは、プライマリインスタンスと同じインスタンスクラスとし、Amazon DocumentDB クラスターの異なるアベイラビリティーゾーンに配置します。

詳細については次を参照してください:

- Amazon DocumentDB クラスターの耐障害性について理解する
- Amazon DocumentDB フェイルオーバー
  - フェイルオーバーターゲットの制御

グローバルクラスターによる高可用性

複数の で高可用性を実現するために AWS リージョン、Amazon DocumentDB グローバルクラスターを設定できます。各グローバルクラスターは複数のリージョンにまたがり、低レイテンシーのグローバル読み取りと、 全体の停止からのディザスタリカバリを可能にします AWS リージョン。Amazon DocumentDB は、プライマリリージョンから各セカンダリリージョンへのすべてのデータと更新の複製を自動的に処理します。

レプリカの追加

クラスターに追加される最初のインスタンスは、プライマリインスタンスです。最初のインスタンスの後に追加されるすべてのインスタンスはレプリカインスタンスです。クラスターは、プライマリに加えて 15 個までのレプリカインスタンスを持つことができます。

を使用してクラスターを作成すると AWS Management Console、プライマリインスタンスが自動的に同時に作成されます。クラスターとプライマリインスタンスを作成すると同時にレプリカを作成するには、[別のゾーンにレプリカを作成します] を選択します。詳細については、「Amazon DocumentDB クラスターの作成」のステップ 4.d を参照してください。Amazon DocumentDB クラスターにさらにレプリカを追加するには、クラスターへの Amazon DocumentDB インスタンスの追加 を参照してください。

を使用してクラスター AWS CLI を作成する場合は、プライマリインスタンスとレプリカインスタンスを明示的に作成する必要があります。詳細については、以下のトピックの「 の使用 AWS CLI」セクションを参照してください。

- Amazon DocumentDB クラスターの作成

- クラスターへの Amazon DocumentDB インスタンスの追加

Amazon DocumentDB フェイルオーバー

特定のタイプの計画的メンテナンスなどの特定の場合、またはプライマリノードまたはアベイラビリティーゾーンの障害が発生する可能性が低い場合、Amazon DocumentDB (MongoDB 互換) が障害を検出し、プライマリノードを置き換えます。フェイルオーバー中、書き込みのダウンタイムは最小限になります。これは、プライマリノードのロールが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代わりに、リードレプリカのいずれかにフェイルオーバーするためです。この障害検出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。

フェイルオーバーが機能するためには、クラスターには少なくとも 2 つのインスタンス、プライマリおよび少なくとも 1 つのレプリカインスタンスが必要です。

> **Note**
>
> このトピックは、元の Amazon DocumentDB インスタンスベースのクラスターにのみ適用されます。Elastic クラスターまたはグローバルクラスターには適用されません。

フェイルオーバーターゲットの制御

Amazon DocumentDB は、フェイルオーバーが発生した場合にプライマリに昇格されるレプリカインスタンスを制御するための手段として、フェイルオーバー階層を提供します。

フェイルオーバー階層

各レプリカインスタンスは、フェイルオーバー階層 (0 ～ 15) と関連付けられます。メンテナンスのためフェイルオーバーが発生するか、予期しないハードウェア障害が発生した場合、プライマリインスタンスは優先度の最も高い (番号が最も小さい層) レプリカにフェイルオーバーします。優先度が同じ階層を持つ複数のレプリカがある場合、プライマリは、プライマリに最も近いサイズのその階層のレプリカにフェイルオーバーします。

選択したレプリカのグループのフェイルオーバー階層を 0 (最大の優先度) に設定することで、フェイルオーバーでそのグループ内のいずれかのレプリカを昇格することができます。フェイルオーバーが発生した場合に、優先度の低い階層 (高い番号) をこれらのレプリカに割り当てることで、特定のレプリカのプライマリへの昇格を実質的に防ぐことができます。これは、特定のレプリカがアプリケーションによって多く使用され、それらの 1 つへのフェイルオーバーによって重要なアプリケーションに悪影響を与える可能性がある状況で役立ちます。

インスタンスのフェイルオーバー階層は、インスタンスの作成時に設定するか、後でインスタンスを変更して設定できます。インスタンスを変更してインスタンスのフェイルオーバーを設定しても、フェイルオーバーはトリガーされません。詳細については、以下のトピックを参照してください。

- クラスターへの Amazon DocumentDB インスタンスの追加
- Amazon DocumentDB インスタンスの変更

手動でフェイルオーバーを開始するときは、プライマリに昇格するレプリカインスタンスを制御するための 2 つの方法として、先ほど説明したフェイルオーバー階層と `--target-db-instance-identifier` パラメータがあります。

`--target-db-instance-identifier`

テストでは、`failover-db-cluster` オペレーションを使用してフェイルオーバーイベントを強制できます。`--target-db-instance-identifier` パラメータを使用して、プライマリに昇格させるレプリカを指定します。`--target-db-instance-identifier` パラメータの使用は、フェイルオーバー優先度階層よりも優先されます。`--target-db-instance-identifier` パラメータを指定しなかった場合、プライマリフェイルオーバーはフェイルオーバー優先度階層に従います。

フェイルオーバー中の処理

フェイルオーバーは Amazon DocumentDB によって自動的に処理されるため、アプリケーションは管理上の介入なく、可能な限り迅速にデータベースオペレーションを再開することができます。

- Amazon DocumentDB プリカが同じアベイラビリティーゾーンや別のアベイラビリティーゾーンにあると、フェイルオーバーが発生した場合、Amazon DocumentDB はインスタンスの正規名レコード (CNAME) を切り替えて正常なレプリカを指定します。これに伴い、この正常なレプリカが新しいプライマリに昇格されます。通常、フェイルオーバーは開始から終了まで 30 秒以内に完了します。
- Amazon DocumentDB レプリカインスタンス (単一インスタンスのクラスターなど) がない場合、Amazon DocumentDB は元のインスタンスと同じアベイラビリティーゾーンに新しいインスタンスを作成しようとします。このような元のインスタンスの置換処理はベストエフォート方式で行われるため、アベイラビリティーゾーンの広範囲に影響する問題がある場合などは失敗する可能性があります。

接続が切断された場合、アプリケーションはデータベースへの接続を再試行する必要があります。

フェイルオーバーのテスト

クラスターのフェイルオーバーにより、クラスター内の Amazon DocumentDB レプリカの 1 つ (読み取り専用インスタンス) が、プライマリインスタンス (クラスターライター) に昇格されます。

Amazon DocumentDB は、プライマリインスタンスが失敗した場合に、存在する場合、自動的に Amazon DocumentDB レプリカにフェイルオーバーします。テストのため、プライマリインスタンスの失敗をシミュレートする場合は、フェイルオーバーを強制できます。クラスター内の各インスタンスには、独自のエンドポイントアドレスがあります。したがって、フェイルオーバーの完了時にこれらのエンドポイントアドレスを使用するすべての既存の接続をクリーンアップして再度確立する必要があります。

フェイルオーバーを強制するには、これらのパラメータを指定して、`failover-db-cluster` オペレーションを使用します。

- `--db-cluster-identifier` — 必須 フェイルオーバーするクラスターの名前。
- `--target-db-instance-identifier` - オプションです。プライマリインスタンスに昇格するインスタンスの名前。

Example

以下のオペレーションでは、`sample-cluster` クラスターのフェイルオーバーを強制します。新しいプライマリインスタンスを作成するインスタンスが指定されないため、Amazon DocumentDB はフェイルオーバー階層の優先度に従ってインスタンスを選択します。

Linux、macOS、Unix の場合:

```
aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster
```

Windows の場合:

```
aws docdb failover-db-cluster ^
   --db-cluster-identifier sample-cluster
```

以下のオペレーションでは、`sample-cluster` クラスターのフェイルオーバーを強制し、`sample-cluster-instance` がプライマリロールに昇格することを指定します。(出力の `"IsClusterWriter": true` に注目してください。)

Linux、macOS、Unix の場合:

```
aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster \
   --target-db-instance-identifier sample-cluster-instance
```

Windows の場合:

```
aws docdb failover-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --target-db-instance-identifier sample-cluster-instance
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "DBCluster": {
        "HostedZoneId": "Z2SUY0A1719RZT",
        "Port": 27017,
        "EngineVersion": "3.6.0",
        "PreferredMaintenanceWindow": "thu:04:05-thu:04:35",
        "BackupRetentionPeriod": 1,
        "ClusterCreateTime": "2018-06-28T18:53:29.455Z",
        "AssociatedRoles": [],
        "DBSubnetGroup": "default",
        "MasterUsername": "master-user",
        "Engine": "docdb",
        "ReadReplicaIdentifiers": [],
        "EarliestRestorableTime": "2018-08-21T00:04:10.546Z",
        "DBClusterIdentifier": "sample-cluster",
        "ReaderEndpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "DBClusterMembers": [
            {
                "DBInstanceIdentifier": "sample-cluster-instance",
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1,
                "IsClusterWriter": true
            },
            {
                "DBInstanceIdentifier": "sample-cluster-instance-00",
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1,
                "IsClusterWriter": false
```

```
            },
            {
                "DBInstanceIdentifier": "sample-cluster-instance-01",
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1,
                "IsClusterWriter": false
            }
        ],
        "AvailabilityZones": [
            "us-east-1b",
            "us-east-1c",
            "us-east-1a"
        ],
        "DBClusterParameterGroup": "default.docdb3.6",
        "Endpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "IAMDatabaseAuthenticationEnabled": false,
        "AllocatedStorage": 1,
        "LatestRestorableTime": "2018-08-22T21:57:33.904Z",
        "PreferredBackupWindow": "00:00-00:30",
        "StorageEncrypted": false,
        "MultiAZ": true,
        "Status": "available",
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-12345678"
            }
        ],
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    }
}
```

レプリケーションの遅延

レプリケーションラグは通常 50 ms 以下です。レプリカのラグが上がる最も一般的な理由は次のとおりです。

- リードレプリカがプライマリより遅くなる原因として、プライマリの書き込みレートが高いです。
- 長時間実行されるクエリ (大規模なシーケンシャルスキャン、集約クエリなど) と着信書き込みレプリケーションの間のリードレプリカの競合。
- リードレプリカの同時クエリが非常に多いです。

レプリケーションの遅延を最小限に抑えるには、次のトラブルシューティング方法を試してください。

- 書き込みレートが高いか CPU 使用率が高い場合は、クラスター内のインスタンスをスケールアップすることをお勧めします。
- リードレプリカで長時間実行されるクエリがあり、クエリ対象のドキュメントが非常に頻繁に更新される場合は、リードレプリカでの競合を回避するために、長時間実行されるクエリを変更するか、プライマリ/書き込みレプリカに対して実行することを検討してください。
- 同時クエリが非常に多い場合や、リードレプリカでのみ高い CPU 使用率がある場合、別のオプションは、リードレプリカの数をスケールアウトしてワークロードを分散することです。
- レプリケーションラグは、高い書き込みスループットと長時間実行されるクエリの結果であるため、DBClusterReplicalagMaximum CW メトリクスを低速クエリロガーと `WriteThroughput`/`WriteIOPS` メトリクスと組み合わせて、レプリケーションラグをトラブルシューティングすることをお勧めします。

一般に、クラスターのフェイルオーバーによってパフォーマンスが低下しないように、すべてのレプリカが同じインスタンスタイプであることを推奨します。

スケールアップとスケールアウト (例：6 つの小さいインスタンスと 3 つの大きなインスタンス) を選択する場合は、DB インスタンスごとに大きなバッファーキャッシュが得られるため、スケールアウト前に最初に (より大きなインスタンス) にスケールアップすることをお勧めします。

プロアクティブに、レプリケーション・ラグ・アラームを設定し、アプリケーションの機能に影響を及ぼす前に、しきい値をレプリカ・インスタンス上のデータがどの程度遅れる (または「古い」) 上限であると感じる値に設定する必要があります。一般に、一時的なワークロードが原因で、アラームが発生する前に、複数のデータポイントについてレプリケーションラグのしきい値を超えることをお勧めします。

> **Note**
>
> さらに、10 秒を超えるレプリケーションラグに対して別のアラームを設定することをお勧めします。複数のデータポイントでこのしきい値を超える場合は、インスタンスをスケールアップするか、プライマリインスタンスの書き込みスループットを削減することをお勧めします。

Amazon DocumentDB インデックスの管理

トピック

- Amazon DocumentDB インデックスの作成
- Amazon DocumentDB インデックスのメンテナンス

Amazon DocumentDB インデックスの作成

Amazon DocumentDB でインデックスを作成するには、いくつかの決定を行う必要があります。

- どのくらいの速さで完了する必要がありますか?
- ビルドの実行中にコレクションにアクセスできなくなることはありますか?
- インスタンスのコンピューティングパワーのうち、ビルドに割り当てることができるのはどれぐらいですか?
- どのタイプのインデックスを作成すべきですか?

本セクションでは、これらの疑問に答え、インスタンスベースのクラスターコレクションで Amazon DocumentDB インデックスを作成するためのコマンドやモニタリング例を提供します。

ガイドライン

以下のガイドラインには、新しいインデックスを作成する際の基本的な制限と設定のトレードオフが含まれています。

- Amazon DocumentDB バージョンのサポート: 単一ワーカーインデックス作成はすべての Amazon DocumentDB バージョンでサポートされていますが、マルチワーカーインデックスは Amazon DocumentDB バージョン 4.0 と 5.0 でのみサポートされています。
- パフォーマンスのトレードオフ: インデックス作成プロセスのワーカー数を増やすと、Amazon DocumentDB データベースのプライマリインスタンスの CPU 使用率と読み取り IO が増加します。新しいインデックスを作成するのに必要なリソースは、実行中のワークロードでは使用できなくなります。
- Elastic クラスター: Amazon DocumentDB Elastic クラスターでは並列インデックスはサポートされていません。
- 最大ワーカー: 設定できるワーカーの最大数は、データベースクラスター内のプライマリインスタンスのサイズによって異なります。これは、データベースクラスターのプライマリインスタンスに

ある vCPU の総数の半分です。たとえば、64 個の vCPU を備えた db.r6g.16xlarge インスタンスでは、最大 32 ワーカーを実行できます。

> **Note**
>
> 並列ワーカーは 2xlarge 以下のインスタンスクラスではサポートされていません。

- 最小ワーカー: 設定できるワーカーの最小数は 1 です。インスタンスベースのクラスターでのインデックス作成のデフォルト設定は 2 ワーカーです。ただし、「ワーカースレッド」オプションを使用すると、ワーカーの数を 1 つに減らすことができます。これはプロセスが 1 つのワーカーで実行されます。
- インデックス圧縮: Amazon DocumentDB はインデックス圧縮をサポートしていません。インデックスのデータサイズは、他のオプションを使用したときより大きくなる可能性があります。
- 複数コレクションのインデックス作成: データベースクラスターのプライマリインスタンスにある vCPU の半分を、複数のコレクションでインデックス作成を実行する設定済みワーカーに使用できます。
- インデックスタイプ: Amazon DocumentDB でサポートされているインデックスタイプの詳細については、「このブログ記事」を参照してください。

入門

特定のコレクションでインデックスの作成を開始するには、`createIndexes` コマンドを使用します。デフォルトでは、このコマンドは 2 つの並列ワーカーを実行するため、インデックス作成プロセスの速度が 2 倍向上します。

たとえば、次のコマンドプロセスは、ドキュメント内の「user_name」フィールドのインデックスを作成し、インデックス作成プロセスの速度を 4 ワーカーに上げる方法を示しています。

1. クラスター上の 2 つの並列ワーカーを使用してインデックスを作成します。

   ```
   db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
    "name":"username_idx"}]})
   ```

2. インデックス作成プロセスの速度を最適化するため、`db.runCommand createIndexes` コマンドの「ワーカースレッド」オプション (`"workers":<number>`) を使用してワーカー数を指定できます。

   プロセスの速度を 4 つの並列ワーカーに上げます。

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "workers":4}]})
```

> **Note**
>
> ワーカーの数が増えるほど、インデックスの作成が速くなります。ただし、ワーカーの数が増えるほど、プライマリインスタンスの vCPU と読み取り IO のロードも大きくなります。他のワークロードを低下させることなく、増加したロードを処理できるよう、クラスターが十分にプロビジョニングされていることを確認してください。

インデックスの進捗状況

インデックス作成プロセスは、初期化、コレクションのスキャン、キーのソート、そして最後にインデックスビルダーによるキーの挿入によって行われます。このプロセスは、フォアグラウンドで実行する場合は最大 6 段階、バックグラウンドで実行する場合は最大 9 段階に分かれます。完了率、スキャンされたストレージブロックの総数、ソートされたキー、挿入されたキーなどのステータスメトリックをステージごとに表示できます。

mongo シェルの `db.currentOp()` コマンドを使用して、インデックス作成プロセスの進行状況をモニタリングします。最後の段階が 100% 完了すると、すべてのインデックスが正常に作成されたことがわかります。

```
db.currentOp({"command.createIndexes": { $exists : true } })
```

> **Note**
>
> インデックス作成の進行状況の表示は、Amazon DocumentDB 5.0 でのみサポートされています。

インデックスビルドのタイプ

インデックスビルドには次の 4 つのタイプがあります。

- フォアグラウンド — フォアグラウンドインデックスビルドは、インデックスが作成されるまで、他のすべてのデータベースオペレーションをブロックします。Amazon DocumentDB フォアグラウンドビルドは 5 つのステージで構成されています。

- フォアグラウンド (一意) — 単一ドキュメント (一意) フォアグラウンドインデックスビルドは、通常のフォアグラウンドビルドなどの他のデータベースオペレーションをブロックします。基本的なフォアグラウンドビルドとは異なり、単一ビルドでは追加のステージ (ソートキー 2) を使用して重複キーを探します。フォアグラウンド (単一) ビルドは 6 つのステージで構成されます。
- バックグラウンド — バックグラウンドインデックスビルドでは、インデックスの作成中に他のデータベースオペレーションをフォアグラウンドで実行できます。Amazon DocumentDB バックグラウンドビルドは 8 つのステージで構成されています。
- 背景 (一意) — 単一ドキュメント (一意) の背景インデックスビルドでは、インデックスの作成中に他のデータベースオペレーションをフォアグラウンドで実行できます。基本的なバックグラウンドビルドとは異なり、単一ビルドでは追加のステージ (ソートキー 2) を使用して重複キーを探します。バックグラウンド (単一) ビルドは 9 つのステージで構成されます。

インデックスビルドステージ

ステージ	フォアグラウンド	フォアグラウンド (単一)	背景	バックグラウンド (単一)
初期化	1	1	1	1
インデックスの構築: 初期化	2	2	2	2
インデックスの構築: コレクションのスキャン	3	3	3	3
インデックスの構築: ソートキー 1	4	4	4	4
インデックスの構築: ソートキー 2		5		5
インデックスの構築: キーの挿入	5	6	5	6

ステージ	フォアグラウンド	フォアグラウンド (単一)	背景	バックグラウンド (単一)
検証: インデックスのスキャン			6	7
検証: タプルのソート			7	8
検証: コレクションのスキャン			8	9

- 初期化: createIndex はインデックスビルダーを準備しています。このフェーズは非常に短いはずです。
- インデックスの構築: 初期化 - インデックスビルダーはインデックスを作成する準備をしています。このフェーズは非常に短いはずです。
- インデックスの構築: コレクションのスキャン - インデックスビルダーはコレクションスキャンを実行してインデックスキーを収集しています。測定単位は「ブロック」です。

> **Note**
>
> インデックス構築用に複数のワーカーが設定されている場合は、この段階で表示されます。「コレクションのスキャン」ステージは、インデックス構築プロセス中に複数のワーカーを使用する唯一のステージです。その他のすべてのステージには 1 人のワーカーが表示されます。

- インデックスの構築: ソートキー 1 - インデックスビルダーは収集されたインデックスキーをソートしています。測定単位は「キー」です。
- インデックスの構築: ソートキー 2 - インデックスビルダーは、収集されたデッドタプルに対応するインデックスキーをソートします。このフェーズは単一インデックスの構築にのみ適用されます。測定単位は「キー」です。
- インデックスの構築: キーの挿入 - インデックスビルダーは、新しいインデックスにインデックスキーを挿入しています。測定単位は「キー」です。
- 検証: インデックスのスキャン - createIndex はインデックスをスキャンして、検証が必要なキーを見つけます。測定単位は「ブロック」です。
- 検証: タプルのソート - createIndex はインデックススキャンフェーズの出力をソートしています。

- 検証: コレクションのスキャン - createIndex はコレクションをスキャンして、前の 2 つのフェーズで見つかったインデックスキーを検証しています。測定単位は「ブロック」です。

インデックスビルドの出力例

以下の出力例 (フォアグラウンドインデックスビルド) では、インデックス作成のステータスが表示されています。「msg」フィールドには、ビルドのステージと完了率を示すことで、ビルドの進捗状況がまとめられています。「workers」フィールドは、インデックスビルドの各段階で使用されたワーカーの数を示します。「progress」フィールドには、完了率の計算に使用された実際の数値が表示されます。

> **Note**
>
> 「currentIndexBuildName」、「msg」、および「progress」フィールドは Amazon DocumentDB バージョン 4.0 ではサポートされていません。

```
{
    "inprog" : [{
    …
        "command": {
            "createIndexes": "test",
            "indexes": [{
                "v": 2,
                "key": {
                    "user_name": 1
                },
                "name": "user_name_1"
            }],
            "lsid": {
                "id": UUID(“094d0fba-8f41-4373-82c3-7c4c7b5ff13b”)
            },
            "$db": "test"
        },
        "currentIndexBuildName": user_name_1,
        "msg": "Index Build: building index number_1, stage 6/6 building index:
 656860/1003520 (keys) 65%",
        "workers": 1,
        "progress": {
            "done": 656861,
            "total": 1003520
```

```
        },
    …
    ],

    "ok" : 1
}
```

Amazon DocumentDB インデックスのメンテナンス

を使用したインデックスのメンテナンス **reIndex**

reIndex は、インデックスの再構築に使用されるコマンドです。通常、インデックスが破損したり、または非効率的になった場合に使用されます。時間の経過とともに、インデックスには多くの更新、挿入、削除に伴い未使用のスペースが蓄積され、パフォーマンスが低下する恐れがあります。インデックスの再構築は、このような未使用のスペースを削除し、インデックスの効率を回復するのに役立ちます。

reIndex ガイドライン

- reIndex は現在、Amazon DocumentDB 5.0 でのみサポートされています。
- Amazon DocumentDB は、バックグラウンドでの 1 つのインデックスに対する reindex をサポートし、複数のワーカーの関与を可能にします。reIndex プロセスの実行中も、古いインデックスを用いたクエリの実行が可能です。
- Amazon DocumentDB は、currentOp によるインデックス構築の進行状況レポートをサポートしています。レポートでは、インデックスの作成中に表示される インデックスビルドステージ と同様のインデックス構築ステージを確認できます。唯一の違いは、一意であるかどうかにかかわらず、reIndex のステージ数は常に 8 ステージであることです。「インデックスのビルド中: キーのソート中 #2」のようなステージはありません。
- reIndex は、同じコレクション上で実行されるインデックス関連のコマンド (createIndexes、dropIndexes、collMod、renameCollection) を除いて、あらゆるコマンドと同時に実行できます。
- reIndex は現在、テキスト、地理空間、ベクトル、部分インデックスについてはサポートされていません。

reIndex の構築

次のコマンドを使用してインデックスを再構築します。

```
db.runCommand({ reIndex: "collection-name", index: "index-name"})
```

オプションで、再構築プロセスに割り当てるワーカーの数をコントロールすることもできます。

```
db.runCommand({ reIndex: "collection-name", index: "index-name", workers: number })
```

コレクションレベルのドキュメント圧縮の管理

Amazon DocumentDB のコレクションレベルのドキュメント圧縮では、コレクション内のドキュメントを圧縮することで、ストレージと IO のコストを削減できます。コレクションレベルでのドキュメント圧縮を有効にし、圧縮ドキュメントのストレージサイズや圧縮ステータスなどの圧縮メトリクスに基づいて得られたストレージを測定することで、必要に応じて圧縮メトリクスを確認できます。Amazon DocumentDB は LZ4 圧縮アルゴリズムを使用してドキュメントを圧縮します。

Amazon DocumentDB は、バージョン 5.0 以降でドキュメント圧縮をサポートしています。以下は、コレクションレベルのドキュメント圧縮機能です。

- デフォルトの動作 — クラスター上の新しいコレクションのデフォルトの圧縮設定は、クラスターパラメータ `default_collection_compression` によって決まります。このパラメータは、デフォルトでは「無効」に設定されています。
- 既存のコレクションの圧縮 — 既存のコレクションの圧縮設定は、`collMod` コマンドを使用して変更できます。
- 圧縮しきい値の変更 — デフォルトの圧縮しきい値は 2KB です。`createCollection` コマンドを使用して新しいコレクション向けに圧縮しきい値を指定したり、`collMod` コマンドを使用して既存のコレクション向けに値を変更することができます。

> **Note**
>
> Amazon DocumentDB ドキュメント圧縮は Amazon DocumentDB バージョン 3.6 および 4.0 ではサポートされていません。

トピック

- ドキュメント圧縮の管理
- ドキュメント圧縮のモニタリング

ドキュメント圧縮の管理

コレクション内でのドキュメント圧縮の有効化

Amazon DocumentDB でコレクションを作成する際に、db.createCollection() の方法でドキュメント圧縮を有効にすることができます。

```
db.createCollection( sample_collection,{
    storageEngine : {
        documentDB: {
            compression:{enable: <true | false>}
        }
    }
})
```

クラスター内でのドキュメント圧縮の有効化

クラスターパラメータ default_collection_compression を「有効」に設定することで、クラスターレベルのすべての新しいコレクションでドキュメント圧縮をデフォルトで有効にできます。このパラメータを「有効」に設定すると、クラスターで新しく作成されるコレクションではデフォルトで圧縮が有効になり、圧縮しきい値は 2 KB になります。

既存のコレクションの圧縮

collMod オペレーションの使用により、以下の storageEngine 設定を指定して既存のコレクションの圧縮設定を変更することもできます。なお、このコマンドを使用して行われる変更は、新たに挿入/更新されるドキュメントにのみ適用され、過去に挿入されたドキュメントに対する圧縮は変更されないことに注意してください。

```
db.runCommand({
    collMod: "orders",
    storageEngine: {
        documentDB: {compression: {enable: <true | false>} }
    }
})
```

圧縮しきい値の設定

デフォルトでは、圧縮されたコレクションの圧縮しきい値は 2032 バイトです。このしきい値は、圧縮を有効にして新しいコレクションを作成するときに createCollection コマンドで設定できます。

```
db.createCollection( sample_collection, {
    storageEngine : {
        documentDB: {
            compression: {
                enable: true,
                threshold: <128 - 8000>
            }
        }
    }
})
```

collMod オペレーションの使用により、次の storageEngine 設定を指定して既存の圧縮コレクションの圧縮しきい値を変更することもできます。

```
db.runCommand({
    collMod: "orders",
    storageEngine: {
        documentDB: {
            compression: {
                enable: true,
                threshold: <128 - 8000>
            }
        }
    }
})
```

圧縮しきい値は 128～8000 バイトの範囲でのみ設定できることに留意してください。さらに、圧縮しきい値を指定するときは、enable オプションを「true」に設定する必要があります。

ドキュメント圧縮のモニタリング

コレクションが圧縮されているかどうかを確認し、圧縮率を求める方法は次の通りです。

mongo シェルから db.printCollectionStats() または db.collection.stats() コマンドを実行して圧縮統計を表示します。出力に元のサイズと圧縮サイズが表示されるため、両者を比較してドキュメント圧縮によって得られたストレージを分析できます。この例では、「sample_collection」という名前のコレクションの統計が下に表示されます。以下のように、スケーリング係数 1024*1024 を使用して、size と storageSize の値を MB 単位で出力します。

```
db.sample_collection.stats(1024*1024)
```

上記のコマンドを使った出力例を以下にご紹介します。

```
{
    "ns" : "test.sample_collection",
    "count" : 1000000,
    "size" : 3906.3,
    "avgObjSize" : 4096,
    "storageSize" : 1953.1,
    compression:{"enabled" : true,"threshold" : 2032},
    ...
}
```

- size: ドキュメントコレクションの元のサイズ。
- avgObjSize: 圧縮前のドキュメントサイズ平均 (小数第一位まで四捨五入)。測定単位はバイト。
- storageSize: 圧縮後のコレクションのストレージサイズ。測定単位はバイト。
- enabled: 圧縮の有効/無効を示します。

実際の圧縮率を計算するには、コレクションサイズをストレージサイズで割ります (size/storageSize)。上記の例の場合、計算は 3906.3/1953.1 で、これは 2:1 の圧縮率に相当します。

Amazon DocumentDB イベントの管理

Amazon DocumentDB (MongoDB との互換性あり) は、クラスター、インスタンス、スナップショット、セキュリティグループ、およびクラスターパラメータグループに関連するイベントのレコードを保持します。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、イベントに関連付けられたメッセージなどがあります。

> **Important**
>
> 特定の管理機能のために、Amazon DocumentDB は Amazon RDS および Amazon Neptune と共有したオペレーションテクノロジーを使用します。Amazon DocumentDB、Amazon RDS、および Amazon Neptune 間でリージョンレベルで管理されている制限のあるリージョンが共有されています。詳細については、「リージョン別クォータ」を参照してください。

トピック

- Amazon DocumentDB のイベントカテゴリの表示

- Amazon DocumentDB イベントの表示

Amazon DocumentDB のイベントカテゴリの表示

各 Amazon DocumentDB リソースタイプには、関連付けることができる特定のタイプのイベントがあります。オペレーションを使用して AWS CLI `describe-event-categories`、イベントタイプと Amazon DocumentDB リソースタイプのマッピングを表示できます。

パラメータ

- **`--source-type`** - オプションです。特定のソースタイプのイベントカテゴリを表示するには、`--source-type` パラメータを使用します。使用できる値を以下に示します。
  - `db-cluster`
  - `db-instance`
  - `db-parameter-group`
  - `db-security-group`
  - `db-cluster-snapshot`
- **`--filters`** - オプションです。Amazon DocumentDB のイベントカテゴリを表示するには、フィルター `--filter Name=engine,Values=docdb` を使用します。

Example

次のコードは、クラスターに関連付けられているイベントカテゴリを一覧表示しています。

Linux、macOS、Unix の場合:

```
aws docdb describe-event-categories \
    --filter Name=engine,Values=docdb \
    --source-type db-cluster
```

Windows の場合:

```
aws docdb describe-event-categories ^
    --filter Name=engine,Values=docdb ^
    --source-type db-cluster
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "EventCategoriesMapList": [
        {
            "EventCategories": [
                "notification",
                "failure",
                "maintenance",
                "failover"
            ],
            "SourceType": "db-cluster"
        }
    ]
}
```

次のコードは、各 Amazon DocumentDB ソースタイプに関連付けられているイベントカテゴリを一覧表示しています。

```
aws docdb describe-event-categories
```

このオペレーションによる出力は、次のようになります（JSON 形式）。

```
{
    "EventCategoriesMapList": [
        {
            "SourceType": "db-instance",
            "EventCategories": [
                "notification",
                "failure",
                "creation",
                "maintenance",
                "deletion",
                "recovery",
                "restoration",
                "configuration change",
                "read replica",
                "backtrack",
                "low storage",
                "backup",
                "availability",
                "failover"
            ]
        },
```

```
        {
            "SourceType": "db-security-group",
            "EventCategories": [
                "configuration change",
                "failure"
            ]
        },
        {
            "SourceType": "db-parameter-group",
            "EventCategories": [
                "configuration change"
            ]
        },
        {
            "SourceType": "db-cluster",
            "EventCategories": [
                "notification",
                "failure",
                "maintenance",
                "failover"
            ]
        },
        {
            "SourceType": "db-cluster-snapshot",
            "EventCategories": [
                "backup"
            ]
        }
    ]
}
```

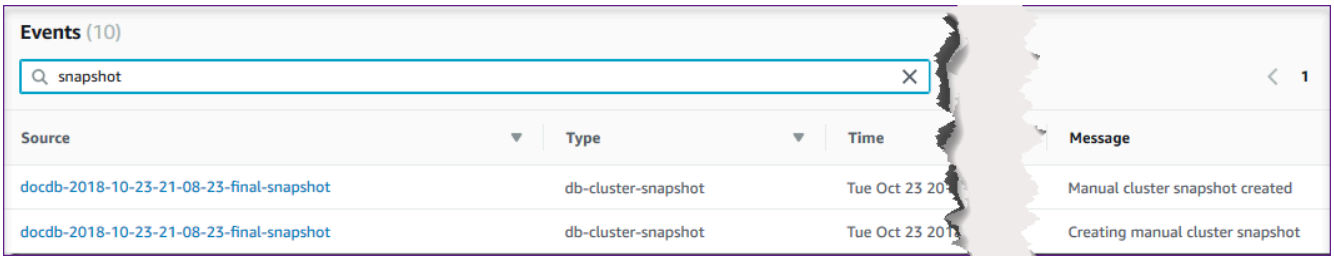
Amazon DocumentDB イベントの表示

Amazon DocumentDB コンソールを使用して Amazon DocumentDB リソースのイベントを取得し、過去 24 時間のイベントを確認できます。また、Amazon DocumentDB リソースのイベントを取得するには、describe-events AWS CLI コマンド、またはDescribeEvents Amazon DocumentDB API オペレーションを使用します。 AWS CLI または Amazon DocumentDB API を使用してイベントを表示する場合は、過去 14 日間のイベントを取得できます。

Using the AWS Management Console

過去 24 時間で発生したAmazon DocumentDB インスタンスのすべてのイベントを表示するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインの [Events] を選択してください。使用できるイベントがリストに表示されます。
3. タイプ別にイベントをフィルタリングするには、[フィルター] リストを使用します。テキストボックスにキーワードを入力して、結果をさらに絞り込むこともできます。例えば、次のスクリーンショットは、スナップショット イベントのすべての Amazon DocumentDB イベントをフィルタリングすることを示しています



Using the AWS CLI

過去 7 日間で発生したAmazon DocumentDB インスタンスのすべてのイベントを表示するには

過去 7 日間のすべての Amazon DocumentDB インスタンスイベントを表示するには、 `--duration`パラメータを `10080` (10,080 分) に設定して describe-events AWS CLI オペレーションを実行します。

```
aws docdb describe-events --duration 10080
```

Amazon DocumentDB イベントのフィルタリング

特定の Amazon DocumentDB イベントを表示するには、以下のパラメータを指定して、`describe-events` オペレーションを使用します。

パラメータ

- **`--filter`** - Amazon DocumentDB イベントに対する戻り値を制限するために必要です。**`Name=engine,Values=docdb`** を使用して、Amazon DocumentDB だけのすべてのイベントをフィルタリングします。

- **--source-identifier** - オプションです。イベントが返されるイベントソースの識別子。省略した場合、すべてのソースからのイベントが結果に含まれます。
- **--source-type** - オプションです。--source-identifier が指定されていない限り必須。--source-identifier が指定されている場合、--source-type は --source-identifier の種類と一致する必要があります。使用できる値を以下に示します。
  - db-cluster
  - db-instance
  - db-parameter-group
  - db-security-group
  - db-cluster-snapshot

次の例では、すべての Amazon DocumentDB イベントを一覧表示します。

```
aws docdb describe-events --filters Name=engine,Values=docdb
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "Events": [
        {
            "SourceArn": "arn:aws:rds:us-east-1:123SAMPLE012:db:sample-cluster-
instance3",
            "Message": "instance created",
            "SourceType": "db-instance",
            "Date": "2018-12-11T21:17:40.023Z",
            "SourceIdentifier": "sample-cluster-instance3",
            "EventCategories": [
                "creation"
            ]
        },
        {
            "SourceArn": "arn:aws:rds:us-
east-1:123SAMPLE012:db:docdb-2018-12-11-21-08-23",
            "Message": "instance shutdown",
            "SourceType": "db-instance",
            "Date": "2018-12-11T21:25:01.245Z",
            "SourceIdentifier": "docdb-2018-12-11-21-08-23",
            "EventCategories": [
                "availability"
```

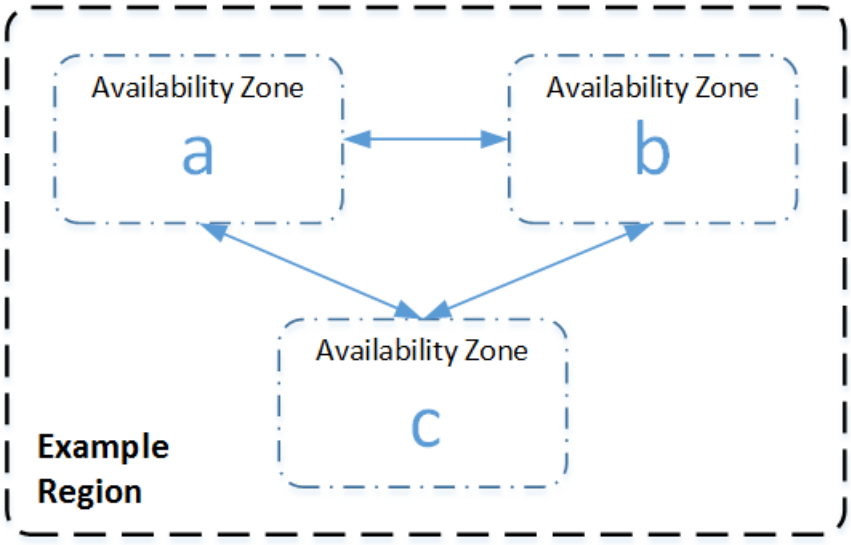
```
            ]
        },
        {
            "SourceArn": "arn:aws:rds:us-
east-1:123SAMPLE012:db:docdb-2018-12-11-21-08-23",
            "Message": "instance restarted",
            "SourceType": "db-instance",
            "Date": "2018-12-11T21:25:11.441Z",
            "SourceIdentifier": "docdb-2018-12-11-21-08-23",
            "EventCategories": [
                "availability"
            ]
        }
    ]
}
```

詳細については、「Amazon DocumentDB イベントの監査」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

Amazon クラウドコンピューティングリソースは、世界各地の多くの場所でホストされています。これらの口ケーションは、 AWS リージョン および アベイラビリティーゾーンで構成されます。各 AWS リージョン は地理的に離れた領域です。リージョンごとにアベイラビリティーゾーンと呼ばれる複数の独立した場所があります。Amazon DocumentDB では、インスタンスなどのリソースやデータを複数の口ケーションに配置することができます。特に AWS リージョン そうしない限り、リソースは にレプリケートされません。

Amazon は、可用性の高い高度なデータセンターを運用しています。しかし、非常にまれですが、同じ場所にあるインスタンスすべての可用性に影響する障害が発生することもあります。もし、すべてのインスタンスを 1 か所でホストしている場合、そのような障害が起きたとき、インスタンスがすべて利用できなくなります。次の図は、3 つのアベイラビリティーゾーン AWS リージョン を持つを示しています。



各リージョンは独立していることを覚えておくことが重要です。Amazon DocumentDB のアクティビティ (インスタンスの作成や利用可能なインスタンスのリストなど) は、現在のデフォルトの AWS リージョンでしか実行されません。EC2_REGION 環境変数を設定して、コンソールのデフォルトリージョンを変更できます。または、 AWS CLIで --region パラメータを使用して上書きすることもできます。詳細については、「 の設定 AWS Command Line Interface」、特に環境変数とコマンドラインオプションのセクションを参照してください。

Amazon DocumentDB コンソールを使用してクラスターを作成し、別のアベイラビリティーゾーンにレプリカを作成することを選択した場合、Amazon DocumentDB は 2 つのインスタンスを作成します。1 つのアベイラビリティーゾーンでプライマリインスタンスが作成され、別のアベイラビリティーゾーンでレプリカインスタンスが作成されます。クラスターボリュームは常に 3 つのアベイラビリティーゾーンにレプリケートされます。

特定の で Amazon DocumentDB インスタンスを作成または操作するには AWS リージョン、対応するリージョンサービスエンドポイントを使用します。

利用可能なリージョン

Amazon DocumentDB は、次の AWS リージョンで利用できます。

Amazon DocumentDB で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	アベイラビリティーゾーン (コンピューティング)
米国東部 (オハイオ)	us-east-2	3

リージョン名	リージョン	アベイラビリティーゾーン (コンピューティング)
米国東部 (バージニア北部)	us-east-1	6
米国西部 (オレゴン)	us-west-2	4
アフリカ (ケープタウン)	af-south-1	3
南米 (サンパウロ)	sa-east-1	3
アジアパシフィック (香港)	ap-east-1	3
アジアパシフィック (ハイデラバード)	ap-south-2	3
アジアパシフィック (ムンバイ)	ap-south-1	3
アジアパシフィック (ソウル)	ap-northeast-2	4
アジアパシフィック (シンガポール)	ap-southeast-1	3
アジアパシフィック (シドニー)	ap-southeast-2	3
アジアパシフィック (東京)	ap-northeast-1	3
カナダ (中部)	ca-central-1	3
中国 (北京) リージョン	cn-north-1	3
中国 (寧夏)	cn-northwest-1	3

リージョン名	リージョン	アベイラビリティーゾーン (コンピューティング)
欧州 (フランクフルト)	eu-central-1	3
欧州 (アイルランド)	eu-west-1	3
欧州 (ロンドン)	eu-west-2	3
欧州 (ミラノ)	eu-south-1	3
欧州 (パリ)	eu-west-3	3
欧州 (スペイン)	eu-south-2	3
欧州 (ストックホルム)	eu-north-1	3
中東 (アラブ首長国連邦)	me-central-1	3
イスラエル (テルアビブ)	il-central-1	3
AWS GovCloud (米国西部)	us-gov-west-1	3
AWS GovCloud (米国東部)	us-gov-east-1	3

デフォルトでは、Amazon DocumentDB クラスターのタイムゾーンは協定世界時 (UTC) です。

特定のリージョン内のクラスターとインスタンスの接続エンドポイントを見つける方法については、Amazon DocumentDB エンドポイントについてを参照してください。

Amazon DocumentDB クラスターパラメータグループの管理

クラスターパラメータグループのパラメータを使用して Amazon DocumentDB エンジンの設定を管理できます。クラスターパラメータグループ は、Amazon DocumentDB クラスターのパラメータ管理を容易にする Amazon DocumentDB の設定値のコレクションです。クラスターパラメータグルー

プは、クラスター内のすべてのインスタンスに適用されるエンジン設定値のコンテナとして機能します。

このセクションでは、クラスターパラメータグループを作成、表示、および変更する方法について説明します。また、任意のクラスターに関連付けられているクラスターパラメータグループを判別する方法も示します。

トピック

- Amazon DocumentDB クラスターパラメータグループについて説明します。
- Amazon DocumentDB クラスターパラメータグループを作成する
- Amazon DocumentDB クラスターパラメータグループを変更する
- カスタムクラスターパラメータグループを使用するために Amazon DocumentDB クラスターを変更する
- Amazon DocumentDB クラスターパラメータグループをコピーする
- Amazon DocumentDB クラスターパラメータグループをリセットする
- Amazon DocumentDB クラスターパラメータグループを削除する
- Amazon DocumentDB クラスターパラメータリファレンス

Amazon DocumentDB クラスターパラメータグループについて説明します。

`default` クラスターパラメータグループは、新しいリージョンで最初の Amazon DocumentDB クラスターを作成するとき、または新しいエンジンを使用するときに自動的に作成されます。同じリージョンで作成され、同じエンジンバージョンを使用する後続のクラスターは、`default` クラスターパラメータグループを使用して作成されます。

トピック

- Amazon DocumentDB クラスターパラメータグループの詳細について説明します。
- Amazon DocumentDB クラスターのパラメータグループの確認をする

Amazon DocumentDB クラスターパラメータグループの詳細について説明します。

特定のクラスターパラメータグループの詳細を記述するには、 AWS Management Console または AWS Command Line Interface () を使用して次の手順を実行しますAWS CLI。

Using the AWS Management Console

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. [クラスターパラメータグループ] ペインで、詳細情報を表示するパラメータグループの名前を選択します。
4. 表示されるページには、パラメータグループのパラメータ、最近のアクティビティ、タグが表示されます。

   - [クラスターパラメータ] には、パラメータの名前、現在の値、許容値、パラメータが変更可能かどうか、適用タイプ、データタイプ、説明が表示されます。個々のパラメータを修正するには、パラメータを選択し、クラスターパラメータ セクションの編集 を選択します。詳細については、「Amazon DocumentDB クラスターパラメータを変更する」を参照してください。
   - [最近のイベント] で、このパラメータグループの最新のイベントを確認できます。このセクションの検索バーを使用して、これらのイベントをフィルタリングできます。詳細については、「Amazon DocumentDB イベントの管理」を参照してください。
   - [タグ] で、このクラスターパラメータグループのタグを表示できます。タグを追加、または削除するには、編集 を タグ のセクションで選択します。詳細については、「Amazon DocumentDB リソースへのタグ付け」を参照してください。

Using the AWS CLI

`describe-db-cluster-parameter-groups` AWS CLI コマンドを使用して、Amazon DocumentDB 用の単一のクラスターパラメータグループまたはすべてのクラスターパラメータグループの Amazon リソースネーム (ARN)、ファミリー、説明、および名前を表示できま

す。describe-db-cluster-parameters AWS CLI コマンドを使用して、単一のクラスターパラメータグループ内のパラメータとその詳細を表示することもできます。

- **--describe-db-cluster-parameter-groups** - すべてのクラスターパラメータグループとその詳細の一覧を表示するには。
  - **--db-cluster-parameter-group-name** - オプション。説明するクラスターパラメータグループの名前。このパラメータを省略した場合、すべてのクラスターパラメータグループが説明されます。
- **--describe-db-cluster-parameters** - パラメータグループ内のすべてのパラメータとその詳細の一覧を表示するには。
  - **--db-cluster-parameter-group name** — 必須。説明するクラスターパラメータグループの名前。

Example

次のコードは、最大 100 のクラスターパラメータグループと、その ARN、ファミリー、説明、および名前を表示します。

```
aws docdb describe-db-cluster-parameter-groups
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
          "DBClusterParameterGroups": [
              {
                  "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:012345678912:cluster-pg:default.docdb4.0",
                  "DBParameterGroupFamily": "docdb4.0",
                  "Description": "Default cluster parameter group for docdb4.0",
                  "DBClusterParameterGroupName": "default.docdb4.0"
              },
              {
                  "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:012345678912:cluster-pg:sample-parameter-group",
                  "DBParameterGroupFamily": "docdb4.0",
                  "Description": "Custom docdb4.0 parameter group",
                  "DBClusterParameterGroupName": "sample-parameter-group"
              }
          ]
```

```
}
```

Example

次のコードは、sample-parameter-group の ARN、ファミリー、説明、および名前を示しています。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-cluster-parameter-groups \
          --db-cluster-parameter-group-name sample-parameter-group
```

Windows の場合:

```
aws docdb describe-db-cluster-parameter-groups ^
          --db-cluster-parameter-group-name sample-parameter-group
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
          "DBClusterParameterGroups": [
              {
                  "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:cluster-pg:sample-parameter-group",
                  "Description": "Custom docdb4.0 parameter group",
                  "DBParameterGroupFamily": "docdb4.0",
                  "DBClusterParameterGroupName": "sample-parameter-group"
              }
          ]
}
```

Example

次のコードは、sample-parameter-group のパラメータの値を表示します。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

Windows の場合:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

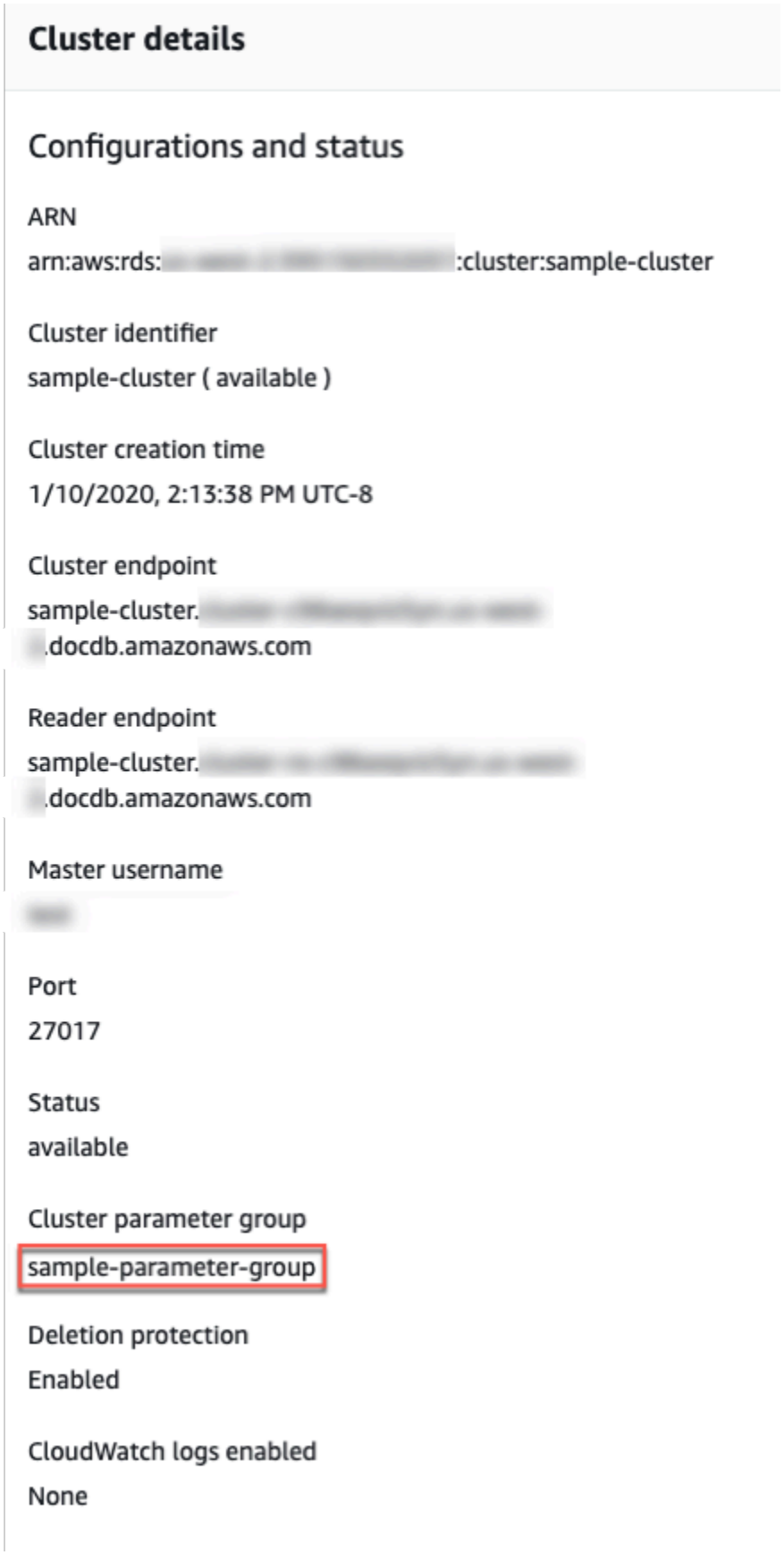
```
{
   "Parameters": [
         {
            "ParameterName": "audit_logs",
            "ParameterValue": "disabled",
            "Description": "Enables auditing on cluster.",
            "Source": "system",
            "ApplyType": "dynamic",
            "DataType": "string",
            "AllowedValues": "enabled,disabled",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
         },
         {
            "ParameterName": "change_stream_log_retention_duration",
            "ParameterValue": "17777",
            "Description": "Duration of time in seconds that the change stream log
 is retained and can be consumed.",
            "Source": "user",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "3600-86400",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
         }
   ]
}
```

Amazon DocumentDB クラスターのパラメータグループの確認をする

特定のクラスターに関連付けられているパラメータグループを確認するには、 AWS Management Console または を使用して次の手順を実行します AWS CLI。

Using the AWS Management Console

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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. クラスターの一覧で、関心のあるクラスターの名前を選択します。
4. 結果のページには、選択したクラスターの詳細が表示されます。[クラスターの詳細] まで下にスクロールします。そのセクションの下部で、[クラスターパラメータグループ] の下にパラメータグループの名前を見つけます。


Cluster details
Configurations and status
ARN
arn:aws:rds: :cluster:sample-cluster
Cluster identifier
sample-cluster (available)
Cluster creation time
1/10/2020, 2:13:38 PM UTC-8
Cluster endpoint
sample-cluster.
.docdb.amazonaws.com
Reader endpoint
sample-cluster.
.docdb.amazonaws.com
Master username
Port
27017
Status
available
Cluster parameter group
sample-parameter-group
Deletion protection
Enabled
CloudWatch logs enabled
None

Using the AWS CLI

次の AWS CLI コードは、クラスター を管理しているパラメータグループを決定しますsample-cluster。

```
aws docdb describe-db-clusters \
      --db-cluster-identifier sample-cluster \
      --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
[
       [
           "sample-cluster",
           "sample-parameter-group"
       ]
]
```

Amazon DocumentDB クラスターパラメータグループを作成する

default.docdb5.0、default.docdb4.0、default.docdb3.6 などのデフォルトのクラスターパラメータグループは、新しいエンジンバージョンを使って新しいリージョンでクラスターを作成したときに作成されます。このリージョンで同じエンジンバージョンを使用して作成された後続のクラスターは、default クラスターパラメータグループを継承します。一度作成した default パラメータグループは、削除したり名前を変更したりすることはできません。設定したパラメータ値を使用してカスタムパラメータグループを作成し、それを Amazon DocumentDB クラスターにアタッチすることで、クラスターインスタンスのエンジン動作を変更できます。

次の手順では、 カスタムクラスターパラメータグループを作成する方法について説明します。その後、そのパラメータグループ内のパラメータを変更できます。

> **Note**
>
> クラスターパラメータグループの作成後、その特定のパラメータグループを使用するまで、5 分以上お待ちください。これにより、クラスターパラメーターグループが新しいクラスターに使用される前に、Amazon DocumentDB が create アクションを完全に完了させることができます。 AWS Management Console または describe-db-cluster-parameter-groups AWS CLI オペレーションを使用して、クラスターパラメータグループ

が作成されたことを確認できます。詳細については、「Amazon DocumentDB クラスターパラメータグループについて説明します。」を参照してください。

Using the AWS Management Console

クラスターパラメータグループを作成するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. [クラスターパラメータグループ] ペインで、[作成] を選択します。
4. [クラスターパラメータグループの作成] ペインで、次のように入力します。

   a. 新しいクラスターパラメータグループ名 — クラスターパラメータグループの名前を入力します。例えば、`sample-parameter-group`。クラスターパラメータグループには、次のような命名に関する制約があります。

      - 長さは [1 ~ 255] 個の英数字です。
      - 1 字目は英字である必要があります。
      - ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

   b. ファミリー — クラスターに使用する DocumentDB バージョンを選択します。

   c. 説明 - このクラスターパラメーターグループの説明を入力します。

5. クラスターパラメータグループを作成するには、[作成] を選択します。操作をキャンセルするには、[キャンセル] を選択します。
6. [作成] を選択すると、ページの上部に次のテキストが表示され、クラスターパラメータグループが正常に作成されたことを確認できます。

```
Successfully created cluster parameter group 'sample-parameter-group'.
```

Using the AWS CLI

Amazon DocumentDB 4.0 クラスターの新しいクラスター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 オペレーションを使用します AWS CLI `create-db-cluster-parameter-group`。

- **`--db-cluster-parameter-group-name`** - カスタムクラスターパラメータグループの名前です。例えば、`sample-parameter-group`。
- **`--db-cluster-parameter-group-family`** - カスタムクラスターパラメータグループのテンプレートとして使用されるクラスターパラメーターグループファミリーです。
- **`--description`** - このクラスターパラメータグループのユーザーが提供する説明です。次の例では、「`Custom docdb4.0 parameter group`」を使用しています。

Linux、macOS、Unix の場合:

Example

```
aws docdb create-db-cluster-parameter-group \
 --db-cluster-parameter-group-name sample-parameter-group \
 --db-parameter-group-family docdb5.0 \
 --description "Custom docdb5.0 parameter group"
```

Windows の場合:

```
aws docdb create-db-cluster-parameter-group ^
 --db-cluster-parameter-group-name sample-parameter-group ^
 --db-parameter-group-family docdb5.0 ^
 --description "Custom docdb5.0 parameter group"
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "sample-parameter-group",
```

```
        "DBParameterGroupFamily": "docdb5.0",
        "Description": "Custom docdb4.0 parameter group",
        "DBClusterParameterGroupArn": "sample-parameter-group-arn"
    }
}
```

Amazon DocumentDB クラスターパラメータグループを変更する

このセクションでは、カスタム Amazon DocumentDB パラメータグループを変更する方法について説明します。Amazon DocumentDB では、新しいリージョンで新しいエンジンバージョンでクラスターを初めて作成したときに作成された `default` クラスターパラメータグループを変更することはできません。Amazon DocumentDB クラスターがデフォルトのクラスターパラメータグループを使用していて、その中の値を変更する場合は、まず 新しいパラメータグループを作成 するか、既存のパラメータグループをコピーし、それを変更してから、変更されたパラメータグループをクラスターに適用します。

カスタムクラスターパラメータグループを変更するには、以下のステップを実行します。変更アクションが反映されるまでに時間がかかる場合があります。変更したクラスターパラメータグループが使用可能になるのを待ってから、クラスターにアタッチしてください。 AWS Management Console または AWS CLI `describe-db-cluster-parameters` オペレーションを使用して、クラスターパラメータグループが変更されたことを確認できます。詳細については、「クラスターパラメータグループの説明」を参照してください。

Using the AWS Management Console

以下のステップに従って、カスタム Amazon DocumentDB パラメータグループを変更します。`default` パラメータグループを変更することはできません。`default` パラメータグループの値を変更する場合は、デフォルトのパラメータグループをコピーして、変更し、次に変更したパラメータグループをクラスターに適用します。クラスターへのパラメータグループの適用の詳細については、「Amazon DocumentDB クラスターの変更」を参照してください。

カスタムのクラスターパラメータグループを変更するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. コンソールの左側のナビゲーションペインで、[パラメータグループ] を選択します。パラメータグループのリストで、変更するパラメータグループの名前を選択します。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. 変更するパラメータグループの各パラメータに対して、以下の操作を行います。

   a. 修正したいパラメータを探し、変更可能 列の下に `true` としてリストされているかどうかを確認することで、それが変更可能であることを確認します。

   b. 変更可能な場合は、パラメータを選択し、コンソールページの右上にある [編集] を選択します。

   c. [**<parameter-name>** の変更] ダイアログボックスで、必要な変更を行います。次に、[クラスターパラメータの変更] を選択するか、[キャンセル] を選択して変更を破棄します。

Using the AWS CLI

カスタムAmazon DocumentDB クラスターパラメータグループの変更可能なパラメータの `ParameterValue`、`Description`、または `ApplyMethod` は、 AWS CLIを使用して変更することができます。デフォルトのクラスターパラメータグループに対して直接変更を加えることはできません。

カスタムクラスターパラメータグループのパラメータを変更するには、以下のパラメータを指定して `modify-db-cluster-parameter-group` オペレーションを使用します。

- **`--db-cluster-parameter-group-name`** — 必須。変更するクラスターパラメータグループの名前。
- **`--parameters`** — 必須。変更するパラメータ。Amazon DocumentDB クラスター内のすべてのインスタンスに適用されるパラメータの一覧は、Amazon DocumentDB クラスターパラメータリファレンス を参照してください。各パラメータの入力には以下を含める必要があります。
  - **`ParameterName`** - 変更するパラメータの名前です。
  - **`ParameterValue`** - このパラメータの新しい値です。
  - **`ApplyMethod`** - このパラメータの変更を適用する方法です。有効な値は、`immediate` および `pending-reboot` です。

> **Note**
>
> ApplyType が static であるパラメータでは、ApplyMethod が pending-reboot である必要があります。

Example - パラメータの値を変更する

この例では、sample-parameter-group のパラメータ値を一覧表示し、tls のパラメータを変更します。次に、5 分間待機したら、sample-parameter-group のパラメータ値もう一度一覧表示して、変更されたパラメータ値を確認します。

1. sample-parameter-group のパラメータとその値を一覧表示します。

   Linux、macOS、Unix の場合:

   ```
   aws docdb describe-db-cluster-parameters \
          --db-cluster-parameter-group-name sample-parameter-group
   ```

   Windows の場合:

   ```
   aws docdb describe-db-cluster-parameters ^
          --db-cluster-parameter-group-name sample-parameter-group
   ```

   このオペレーションによる出力は、次のようになります (JSON 形式) 。

   ```
   {
          "Parameters": [
              {
                  "Source": "system",
                  "ApplyType": "static",
                  "AllowedValues": "disabled,enabled",
                  "ParameterValue": "enabled",
                  "ApplyMethod": "pending-reboot",
                  "DataType": "string",
                  "ParameterName": "tls",
                  "IsModifiable": true,
                  "Description": "Config to enable/disable TLS"
              },
              {
   ```

```
                "Source": "user",
                "ApplyType": "dynamic",
                "AllowedValues": "disabled,enabled",
                "ParameterValue": "enabled",
                "ApplyMethod": "pending-reboot",
                "DataType": "string",
                "ParameterName": "ttl_monitor",
                "IsModifiable": true,
                "Description": "Enables TTL Monitoring"
            }
        ]
}
```

2. tls パラメータを変更し、その値が disabled になるようにします。

   ApplyType が static であるため、ApplyMethod を変更することはできません。

   Linux、macOS、Unix の場合:

```
aws docdb modify-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --parameters
 "ParameterName"=tls,"ParameterValue"=disabled,"ApplyMethod"=pending-reboot
```

   Windows の場合:

```
aws docdb modify-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
       --parameters
 "ParameterName"=tls,"ParameterValue"=disabled,"ApplyMethod"=pending-reboot
```

   このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
       "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

3. 少なくとも 5 分お待ち下さい。
4. sample-parameter-group のパラメータ値を表示して、tls パラメータが変更されたことを確認します。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-cluster-parameters \
       --db-cluster-parameter-group-name sample-parameter-group
```

Windows の場合:

```
aws docdb describe-db-cluster-parameters ^
       --db-cluster-parameter-group-name sample-parameter-group
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
       "Parameters": [
           {
               "ParameterValue": "false",
               "ParameterName": "enable_audit_logs",
               "ApplyType": "dynamic",
               "DataType": "string",
               "Description": "Enables auditing on cluster.",
               "AllowedValues": "true,false",
               "Source": "system",
               "IsModifiable": true,
               "ApplyMethod": "pending-reboot"
           },
           {
               "ParameterValue": "disabled",
               "ParameterName": "tls",
               "ApplyType": "static",
               "DataType": "string",
               "Description": "Config to enable/disable TLS",
               "AllowedValues": "disabled,enabled",
               "Source": "system",
               "IsModifiable": true,
               "ApplyMethod": "pending-reboot"
           }
       ]
}
```

カスタムクラスターパラメータグループを使用するために Amazon DocumentDB クラスターを変更する

Amazon DocumentDB クラスターを作成すると、そのクラスター用に `default.docdb4.0` パラメータグループが自動的に作成されます。`default` クラスターパラメータグループは変更できません。代わりに、Amazon DocumentDB クラスターを変更して、新しいカスタマイズされたパラメータグループをそれに関連付けることができます。

このセクションでは、 AWS Management Console と AWS Command Line Interface () を使用して、既存の Amazon DocumentDB クラスターを変更してカスタムクラスターパラメータグループを使用する方法について説明しますAWS CLI。

Using the AWS Management Console

Amazon DocumentDB クラスターを変更して、新しいデフォルトでないクラスターパラメータグループを使用するには

1. 開始する前に、Amazon DocumentDB クラスターとクラスターパラメータグループが作成されていることを確認してください。詳細については、「Amazon DocumentDB クラスターの作成」および「Amazon DocumentDB クラスターパラメータグループを作成する」を参照してください。
2. クラスターパラメータグループを作成した後、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。ナビゲーションペインで、[クラスター] を選択し、新しいパラメータグループをクラスターに追加します。
3. パラメータグループを関連付けるクラスターを選択します。[アクション] を選択し、[変更] を選択してクラスターを変更します。
4. [クラスターオプション] で、クラスターを関連付ける新しいパラメータグループを選択します。
5. [続行] を選択して、変更の概要を表示します。
6. 変更を確認したら、直ちに、あるいは、[変更のスケジューリング] の下にある次のメンテナンスウィンドウ中に適用することができます。
7. [クラスターの変更] を選択して、新しいパラメータグループでクラスターを更新します。

Using the AWS CLI

開始する前に、Amazon DocumentDB クラスターと、クラスターパラメータグループが作成されていることを確認してください。オペレーションを使用して AWS CLI `create-db-cluster`

Amazon DocumentDB クラスターを作成できます。create-db-cluster-parameter-group オペレーションを使用して AWS CLI クラスターパラメータグループを作成できます。

新しいクラスターパラメータグループをクラスターに追加するには、次のパラメータを指定して AWS CLI modify-db-clusterオペレーションを使用します。

- --db-cluster-identifier - クラスターの名前 (例えば、sample-cluster) です。
- --db-cluster-parameter-group-name - クラスターを関連付けるパラメータグループの名前 (例えば、sample-parameter-group) です。

Example

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --db-cluster-parameter-group-name sample-parameter-group
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
"DBCluster": {
    "AvailabilityZones": [
            "us-west-2c",
            "us-west-2b",
            "us-west-2a"
    ],
    "BackupRetentionPeriod": 1,
    "DBClusterIdentifier": "sample-cluster",
    "DBClusterParameterGroup": "sample-parameter-group",
    "DBSubnetGroup": "default",
   ...
}
```

Amazon DocumentDB クラスターパラメータグループをコピーする

Amazon DocumentDB のクラスターパラメータグループのコピーは、 AWS Management Console または AWS Command Line Interface () を使用して作成できますAWS CLI。

Using the AWS Management Console

以下の手順では、既存のクラスターパラメータグループのコピーを作成して、新しいクラスターパラメータグループを作成する手順について説明します。

クラスターパラメータグループをコピーするには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. [クラスターパラメータグループ] ペインで、コピーするクラスターパラメータグループの名前を選択します。
4. [アクション] を選択し、[コピー] を選択して、そのパラメータグループをコピーします。
5. [コピーのオプション] で、新しいクラスターパラメータグループの名前と説明を入力します。次に、[コピー] を選択して変更を保存します。

Using the AWS CLI

クラスターパラメータグループのコピーを作成するには、以下のパラメータを指定して `copy-db-cluster-parameter-group` オペレーションを使用します。

- **`--source-db-cluster-parameter-group-identifier`** — 必須。コピーを作成するクラスターパラメータグループの名前または Amazon リソースネーム (ARN)。

  ソースクラスターパラメータグループとターゲットクラスターパラメータグループが同じである場合 AWS リージョン、識別子は名前または ARN のいずれかになります。

  ソースクラスターパラメータグループとターゲットクラスターパラメータグループが異なる場合 AWS リージョン、識別子は ARN である必要があります。
- **`--target-db-cluster-parameter-group-identifier`** — 必須。クラスターパラメータグループコピーの名前または ARN。

  制約:

  - null、空、または空白にすることはできません。
  - 1 ~ 255 の英字、数字、またはハイフンを含む必要があります。
  - 1 字目は英字である必要があります。
  - ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。
- **`--target-db-cluster-parameter-group-description`** — 必須。クラスターパラメータグループのコピーについてユーザーが入力する説明。

Example

次のコードは、sample-parameter-group のコピーを作成し、コピーに sample-parameter-group-copy という名前を付けます。

Linux、macOS、Unix の場合:

```
aws docdb copy-db-cluster-parameter-group \
    --source-db-cluster-parameter-group-identifier sample-parameter-group \
    --target-db-cluster-parameter-group-identifier sample-parameter-group-copy \
    --target-db-cluster-parameter-group-description "Copy of sample-parameter-group"
```

Windows の場合:

```
aws docdb copy-db-cluster-parameter-group ^
    --source-db-cluster-parameter-group-identifier sample-parameter-group ^
    --target-db-cluster-parameter-group-identifier sample-parameter-group-copy ^
    --target-db-cluster-parameter-group-description "Copy of sample-parameter-group"
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:sample-parameter-group-copy",
        "DBClusterParameterGroupName": "sample-parameter-group-copy",
        "DBParameterGroupFamily": "docdb4.0",
        "Description": "Copy of sample-parameter-group"
    }
}
```

Amazon DocumentDB クラスターパラメータグループをリセットする

Amazon DocumentDB クラスターパラメータグループのパラメータ値の一部またはすべてをデフォルト値にリセットするには、 AWS Management Console または AWS Command Line Interface (AWS CLI) を使用してクラスターパラメータグループをリセットします。

Using the AWS Management Console

以下のステップに従って、クラスターパラメータグループのパラメータ値の一部または全部をデフォルト値にリセットします。

クラスターパラメータグループのパラメータ値をリセットするには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. コンソールの左側のナビゲーションペインで、[パラメータグループ] を選択します。
3. [クラスターパラメータグループ] ペインで、リセットするクラスターパラメータグループの名前を選択します。
4. [アクション] を選択し、[リセット] を選択して、そのパラメータグループをリセットします。
5. 結果の [クラスターパラメータグループのリセットの確認] ページで、そのパラメータグループのすべてのクラスターパラメータをデフォルトにリセットすることを確認します。次に、[リセット] を選択してパラメータグループをリセットします。[キャンセル] を選択して変更を破棄することもできます。

Using the AWS CLI

クラスターパラメータグループのパラメータ値の一部またはすべてをデフォルト値にリセットするには、次のパラメータを指定して `reset-db-cluster-parameter-group` オペレーションを使用します。

- **`--db-cluster-parameter-group-name`** — 必須。リセットするクラスターパラメータグループの名前。
- **`--parameters`** - オプション。デフォルト値にリセットするクラスターパラメータグループの `ParameterName` と `ApplyMethod` のリスト。次のインスタンスの再起動または `reboot-db-instance` リクエストで有効にするには、静的パラメータを `pending-reboot` に設定する必要があります。更新された静的パラメータを適用するクラスター内のすべてのインスタンスに対して、`reboot-db-instance` を呼び出す必要があります。

  このパラメータおよび `--reset-all-parameters` は相互に排他的です。いずれかを使用できますが、両方を使用することはできません。
- **`--reset-all-parameters`** または **`--no-reset-all-parameters`** - オプション。すべてのパラメータ (`--reset-all-parameters`) または一部のパラメータのみ (`--no-reset-all-parameters`) をデフォルト値にリセットするかどうかを指定します。`--reset-all-parameters` パラメータおよび `--parameters` は相互に排他的です。いずれかを使用できますが、両方を使用することはできません。

グループ全体をリセットすると、動的パラメータはただちに更新されます。次のインスタンスの再起動または reboot-db-instance リクエストで有効にするには、静的パラメータを pending-reboot に設定します。更新された静的パラメータを適用するクラスター内のすべてのインスタンスに対して、reboot-db-instance を呼び出す必要があります。

Example

例 1: すべてのパラメータをデフォルト値にリセットする

次のコードでは、クラスターパラメータグループ sample-parameter-group のすべてのパラメータをデフォルト値にリセットします。

Linux、macOS、Unix の場合:

```
aws docdb reset-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --reset-all-parameters
```

Windows の場合:

```
aws docdb reset-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
       --reset-all-parameters
```

例 2: 指定されたパラメータをデフォルト値にリセットする

次のコードでは、クラスターパラメータグループ sample-parameter-group の tls パラメータをデフォルト値にリセットします。

Linux、macOS、Unix の場合:

```
aws docdb reset-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --no-reset-all-parameters \
       --parameters ParameterName=tls,ApplyMethod=pending-reboot
```

Windows の場合:

```
aws docdb reset-db-cluster-parameter-group ^
```

```
        --db-cluster-parameter-group-name sample-parameter-group ^
        --no-reset-all-parameters ^
        --parameters ParameterName=tls,ApplyMethod=pending-reboot
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
        "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

クラスターインスタンスの再起動

静的パラメータの値が変更される前に、クラスターインスタンスを再起動する必要があります。更新された静的パラメータを適用するクラスター内の各インスタンスを再起動します。

Linux、macOS、Unix の場合:

```
aws docdb reboot-db-instance \
        --db-instance-identifier sample-cluster-instance
```

Windows の場合:

```
aws docdb reboot-db-instance ^
        --db-instance-identifier sample-cluster-instance
```

Amazon DocumentDB クラスターパラメータグループを削除する

AWS Management Console または AWS Command Line Interface () を使用して、カスタム Amazon DocumentDB クラスターパラメータグループを削除できますAWS CLI。`default.docdb4.0` クラスターパラメータグループは削除できません。

Using the AWS Management Console

クラスターパラメータグループを削除するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. [パラメータグループ] ペインで、削除するクラスターパラメータグループ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、[削除] を選択します。
5. [削除] 確認ペインで、[削除] を選択してクラスターパラメータグループを削除します。クラスターパラメータグループを維持するには、[キャンセル] を選択します。

Using the AWS CLI

クラスターパラメータグループを削除するには、以下のパラメータを指定して `delete-db-cluster-parameter-group` オペレーションを使用します。

- **`--db-cluster-parameter-group-name`** — 必須。削除するクラスターパラメータグループの名前。これは既存のクラスターパラメータグループである必要があります。*`default.docdb4.0`* クラスターパラメータグループは削除できません。

Example - クラスターパラメータグループの削除

以下の例では、クラスターパラメータグループを削除するための 3 つのステップについて説明します。

1. 削除するクラスターパラメータグループの名前を見つける。
2. 指定されたクラスターパラメータグループを削除する。
3. クラスターパラメータグループが削除されたことを確認する。

1. 削除するクラスターパラメータグループの名前を見つけます。

次のコードは、すべてのクラスターパラメータグループの名前を表示します。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-cluster-parameter-groups \
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Windows の場合:

```
aws docdb describe-db-cluster-parameter-groups ^
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

前のオペレーションの出力は、次のようなクラスターパラメータグループの名前のリストになります (JSON 形式)。

```
[
       [
           "default.docdb4.0"
       ],
       [
           "sample-parameter-group"
       ],
       [
           "sample-parameter-group-copy"
       ]
   ]
```

2. 特定のクラスターパラメータグループを削除します。

以下のコードは、クラスターパラメータグループ `sample-parameter-group-copy` を削除します。

Linux、macOS、Unix の場合:

```
aws docdb delete-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group-copy
```

Windows の場合:

```
aws docdb delete-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group-copy
```

このオペレーションからの出力はありません。

3. 指定されたクラスターパラメータグループが削除されたことを確認します。

次のコードは、残りのすべてのクラスターパラメータグループの名前を表示します。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-cluster-parameter-groups \
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Windows の場合:

```
aws docdb describe-db-cluster-parameter-groups ^
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

前のオペレーションの出力は、次のようなクラスターパラメータグループのリストになります (JSON 形式)。先ほど削除したクラスターパラメータグループはリストに含まれません。

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
[
       [
           "default.docdb4.0"
       ],
       [
           "sample-parameter-group"
       ]
   ]
```

Amazon DocumentDB クラスターパラメータリファレンス

動的なパラメータを変更し、クラスターパラメータグループを保存すると、[すぐに適用] の設定にかかわらず、変更は直ちに適用されます。静的パラメータを変更し、クラスターパラメータグループを保存すると、パラメーターの変更はインスタンスを手動で再起動した後に有効になります。Amazon DocumentDB コンソールを使用するか、`reboot-db-instance` を明示的に呼び出して、インスタンスを再起動できます。

次の表は、Amazon DocumentDB クラスターのすべてのインスタンスに適用されるパラメータを示しています。

Amazon DocumentDB クラスターレベルのパラメータ

パラメータ	デフォルト値	有効値	変更可能	適用タイプ	データタイプ	説明
audit_logs	無効	有効、無効、ddl、dml_read、dml_write、すべて、なし	Yes	動的	String	Amazon CloudWatch 監査ログを有効にするかどうかを定義します。 • **enabled** - CloudTrail の監査ログが有効になります。 • **disabled** - CloudWatch の監査ログが無効になります。 • **ddl** - DDL イベントの監査が有効になります。 • **dml_read** — DML 読み取りイベン

パラメータ	デフォルト値	有効値	変更可能	適用タイプ	データタイプ	説明
						トの監査が有効になっています。 • **dml_write** - DML 書き込みイベントの監査が有効になります。 • **all** - すべてのデータベースイベントの監査が有効になります。 • **none** - 監査が無効になります。
change_stream_log_retention_duration	10800	3600-604800	はい	動的	整数	変更ストリームログが保持され、消費できる期間 (秒単位) を定義します。

パラメータ	デフォルト値	有効値	変更可能	適用タイプ	データタイプ	説明
default_collection_compression	無効	有効、無効	はい	動的	String	クラスター内の新しいコレクションのデフォルトの圧縮設定を定義します。 • **enabled** — デフォルトでは、圧縮は有効になっています。 • **disabled** — デフォルトでは、圧縮は無効になっています。

パラメータ	デフォルト値	有効値	変更可能	適用タイプ	データタイプ	説明
profiler	無効	有効、無効	はい	動的	String	低速のオペレーションについてプロファイリングを有効にします。 • **enabled** - オペレーションが、クラスター定義しきい値 (100 ミリ秒など) よりも長い時間がかかる場合は、Amazon CloudWatch Logs に記録されます。 • **disabled** - 低速のオペレーションは CloudWatch Logs に記録

パラメータ	デフォルト値	有効値	変更可能	適用タイプ	データタイプ	説明
						されません。
profiler_sampling_rate	1.0	0.0-1.0	はい	動的	浮動小数点数	ログに記録されるオペレーションのサンプリングレートを定義します。
profiler_threshold_ms	100	50-2147483646	はい	動的	整数	profiler のしきい値を定義します。 • profiler_threshold_ms より大きいすべてのオペレーションは CloudWatch Logs に記録されます。

パラメータ	デフォルト値	有効値	変更可能	適用タイプ	データタイプ	説明
tls	有効	有効、無効、fips-140-3、tls1.2+、tls1.3+	はい	静的	String	Transport Layer Security (TLS) 接続が必要かどうかを定義します。 • **enabled** - 接続するには、TLS 接続が必要です。 • **disabled** - 接続に TLS 接続を使用することはできません。 • **fips-140-3** - 接続には、連邦情報処理標準 (FIPS) 属性をもつ TLS 接続が必要です。クラスター

パラメータ	デフォルト値	有効値	変更可能	適用タイプ	データタイプ	説明
						は、FIPS 出版物 140-3 に従って、セキュアな接続のみを受け入れます。これは、これらのリージョン (ca-central-1、us-west-2、us-east-1、us-east-2、us-gov-east-1、us-gov-west-1) で、Amazon DocumentDB 5.0 (エンジンバージョン 3.0.3727) 以降のクラスターでのみサ

パラメータ	デフォルト値	有効値	変更可能	適用タイプ	データタイプ	説明
						ポートされます。 • **tls1.2+** — 接続するには、TLS バージョン 1.2 以降を使用する TLS 接続が必要です。これは、Amazon DocumentDB 4.0 (エンジンバージョン 2.0.10980) および Amazon DocumentDB 5.0 (エンジンバージョン 3.0.11051) 以降でのみサポートされています。

パラメータ	デフォルト値	有効値	変更可能	適用タイプ	データタイプ	説明
						• **tls1.3+** — 接続するには、TLS バージョン 1.3 以降を使用する TLS 接続が必要です。これは、Amazon DocumentDB 4.0 (エンジンバージョン 2.0.10980) および Amazon DocumentDB 5.0 (エンジンバージョン 3.0.11051) 以降でのみサポートされています。

パラメータ	デフォルト値	有効値	変更可能	適用タイプ	データタイプ	説明
ttl_monitor	有効	有効、無効	はい	動的	String	クラスターに対して有効期限 (TTL) モニタリングを有効にするかどうかを定義します。 • **enabled** - TTL モニタリングが有効になります。 • **disabled** - TTL モニタリングが無効になります。

Amazon DocumentDB クラスターパラメータを変更する

Amazon DocumentDB では、クラスターパラメータグループ は、クラスターで作成するすべてのインスタンスに適用される パラメータ で構成されています。カスタムクラスターパラメータグループでは、パラメータ値をいつでも変更したり、作成したパラメータグループのすべてのパラメータ値をデフォルトにリセットできます。このセクションでは、Amazon DocumentDB クラスターパラメータグループを構成するパラメータとその値を表示する方法、およびこれらの値を変更または更新する方法について説明します。

パラメータは動的でも静的でもかまいません。動的なパラメータを変更し、クラスターパラメータグループを保存すると、`Apply Immediately` の設定にかかわらず、変更は直ちに適用されます。静

的パラメータを変更し、クラスターパラメータグループを保存すると、変更はインスタンスを手動で再起動した後にのみ有効になります。

Amazon DocumentDB クラスターパラメータグループのパラメータを表示する

Amazon DocumentDB クラスターのパラメータとその値は、 AWS Management Console または を使用して確認できます AWS CLI。

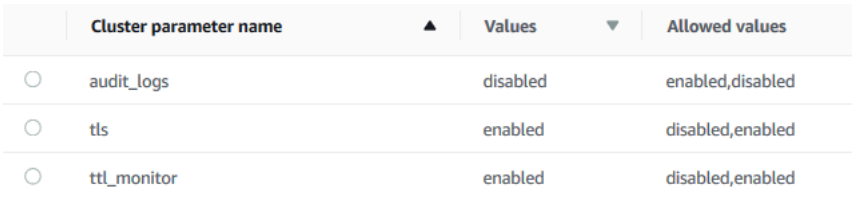
Using the AWS Management Console

クラスターパラメータグループの詳細情報を表示するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. [パラメータグループ] ペインで、詳細情報を表示するクラスターパラメータグループの名前を選択します。
4. 結果ページには、各パラメータのパラメータ名、現在の値、許可された値、パラメータが変更可能かどうか、適用タイプ、データ型および説明の値が表示されます。



Using the AWS CLI

クラスターの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、次のパラメータを指定して `describe-db-cluster-parameters` オペレーションを使用します。

- **--db-cluster-parameter-group-name** — 必須。詳細なパラメータリストを表示するクラスターパラメータグループの名前。
- **--source** - オプション。指定した場合、特定のソースのパラメータだけが返されます。パラメータのソースは engine-default、system、または user のいずれかとすることができます。

Example

次のコードは、custom3-6-param-grp パラメータグループのパラメータと値を一覧表示します。パラメータグループの詳細については、--query の行を省略してください。すべてのパラメータグループについては、--db-cluster-parameter-group-name の行を省略してください。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-cluster-parameters \
   --db-cluster-parameter-group-name custom3-6-param-grp \
   --query 'Parameters[*].[ParameterName,ParameterValue]'
```

Windows の場合:

```
aws docdb describe-db-cluster-parameters ^
   --db-cluster-parameter-group-name custom3-6-param-grp ^
   --query 'Parameters[*].[ParameterName,ParameterValue]'
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
[
    [
        "audit_logs",
        "disabled"
    ],
    [
        "tls",
        "enabled"
    ],
    [
        "ttl_monitor",
        "enabled"
```

```
    ]
]
```

Amazon DocumentDB クラスターパラメータグループのパラメータを変更する

パラメータグループのパラメータは、 AWS Management Console または を使用して変更できます AWS CLI。

Using the AWS Management Console

クラスターパラメータグループのパラメータを更新するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. [パラメータグループ] ペインで、パラメータを更新するクラスターパラメータグループを選択します。
4. 結果のページには、このクラスターパラメータグループのパラメータと対応する詳細が表示されます。更新するパラメータを選択します。
5. ページの右上にある [編集] を選択して、パラメータの値を変更します。クラスターパラメータのタイプの詳細については、「Amazon DocumentDB クラスターパラメータリファレンス」を参照してください。
6. 変更を行い、[クラスターのパラメータを変更] を選択して変更を保存します。変更を破棄するには、[キャンセル] を選択します。

Using the AWS CLI

クラスターパラメータグループのパラメータを変更するには、以下のパラメータを指定して `modify-db-cluster-parameter-group` オペレーションを使用します。

- **--db-cluster-parameter-group-name** — 必須。変更するクラスターパラメータグループの名前。
- **--parameters** — 必須。変更するパラメータ。各パラメータの入力には以下を含める必要があります。
  - **ParameterName** - 変更するパラメータの名前です。
  - **ParameterValue** - このパラメータの新しい値です。
  - **ApplyMethod** - このパラメータの変更を適用する方法です。有効な値は、immediate および pending-reboot です。

> **Note**
>
> ApplyType が static であるパラメータでは、ApplyMethod が pending-reboot である必要があります。

クラスターパラメータグループのパラメータの値を変更するには (AWS CLI)

次の例では、tls パラメータを変更します。

1. **sample-parameter-group** のパラメータとその値を一覧表示します。

   Linux、macOS、Unix の場合:

   ```
   aws docdb describe-db-cluster-parameters \
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   Windows の場合:

   ```
   aws docdb describe-db-cluster-parameters ^
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   このオペレーションによる出力は、次のようになります (JSON 形式) 。

   ```
   {
       "Parameters": [
           {
               "Source": "system",
               "ApplyType": "static",
   ```

```
            "AllowedValues": "disabled,enabled",
            "ParameterValue": "enabled",
            "ApplyMethod": "pending-reboot",
            "DataType": "string",
            "ParameterName": "tls",
            "IsModifiable": true,
            "Description": "Config to enable/disable TLS"
        },
        {
            "Source": "user",
            "ApplyType": "dynamic",
            "AllowedValues": "disabled,enabled",
            "ParameterValue": "enabled",
            "ApplyMethod": "pending-reboot",
            "DataType": "string",
            "ParameterName": "ttl_monitor",
            "IsModifiable": true,
            "Description": "Enables TTL Monitoring"
        }
    ]
}
```

2. **tls** パラメータを変更し、その値が **disabled** になるようにします。ApplyType が static であるため、ApplyMethod を変更することはできません。

   Linux、macOS、Unix の場合:

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 "ParameterName"=tls,ParameterValue=disabled,ApplyMethod=pending-reboot"
```

   Windows の場合:

```
aws docdb modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name sample-parameter-group ^
    --parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-
reboot"
```

   このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
```

```
    "DBClusterParameterGroupName": "sample-parameter-group"
}
```

3. 少なくとも 5 分お待ち下さい。
4. **sample-parameter-group** のパラメータ値を一覧表示します。

   Linux、macOS、Unix の場合:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

   Windows の場合:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

   このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "Parameters": [
        {
            "ParameterName": "audit_logs",
            "ParameterValue": "disabled",
            "Description": "Enables auditing on cluster.",
            "Source": "system",
            "ApplyType": "dynamic",
            "DataType": "string",
            "AllowedValues": "enabled,disabled",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
        },
        {
            "ParameterName": "tls",
            "ParameterValue": "disabled",
            "Description": "Config to enable/disable TLS",
            "Source": "user",
            "ApplyType": "static",
            "DataType": "string",
            "AllowedValues": "disabled,enabled",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
        }
```

```
    ]
}
```

Amazon DocumentDB エンドポイントについて

Amazon DocumentDB (MongoDB 互換) エンドポイントを使用して、クラスターまたはインスタンスに接続できます。Amazon DocumentDB には 3 つの異なる種類のエンドポイントがあり、それぞれに独自の用途があります。

トピック

- クラスターのエンドポイントの検索
- インスタンスのエンドポイントの検索
- エンドポイントへの接続

クラスターエンドポイント

クラスターエンドポイントとは、クラスターの現在のプライマリインスタンスに接続する Amazon DocumentDB のクラスターのエンドポイントです。Amazon DocumentDB クラスターごとに 1 つのクラスターエンドポイントと 1 つのプライマリインスタンスがあります。フェイルオーバーが発生すると、クラスターエンドポイントは新しいプライマリインスタンスに再マッピングされます。

リーダーエンドポイント

読み取りエンドポイントとは、そのクラスターで利用できるレプリカの 1 つに接続する、Amazon DocumentDB クラスターのエンドポイントです。各 Amazon DocumentDB クラスターに読み取りエンドポイントがあります。複数のレプリカがある場合、読み取りエンドポイントは各接続リクエストを Amazon DocumentDB レプリカのいずれかにルーティングします。

インスタンスエンドポイント

インスタンスエンドポイントは、特定のインスタンスに接続するエンドポイントです。クラスターの各インスタンスには、プライマリインスタンスかレプリカインスタンスタイプにかかわらず、独自のインスタンスエンドポイントがあります。アプリケーションでインスタンスエンドポイントを使用しないことが推奨されます。これは、フェイルオーバーの場合にインスタンスエンドポイントがロールを変更することがあり、これによってアプリケーション内のコード変更が必要となるためです。

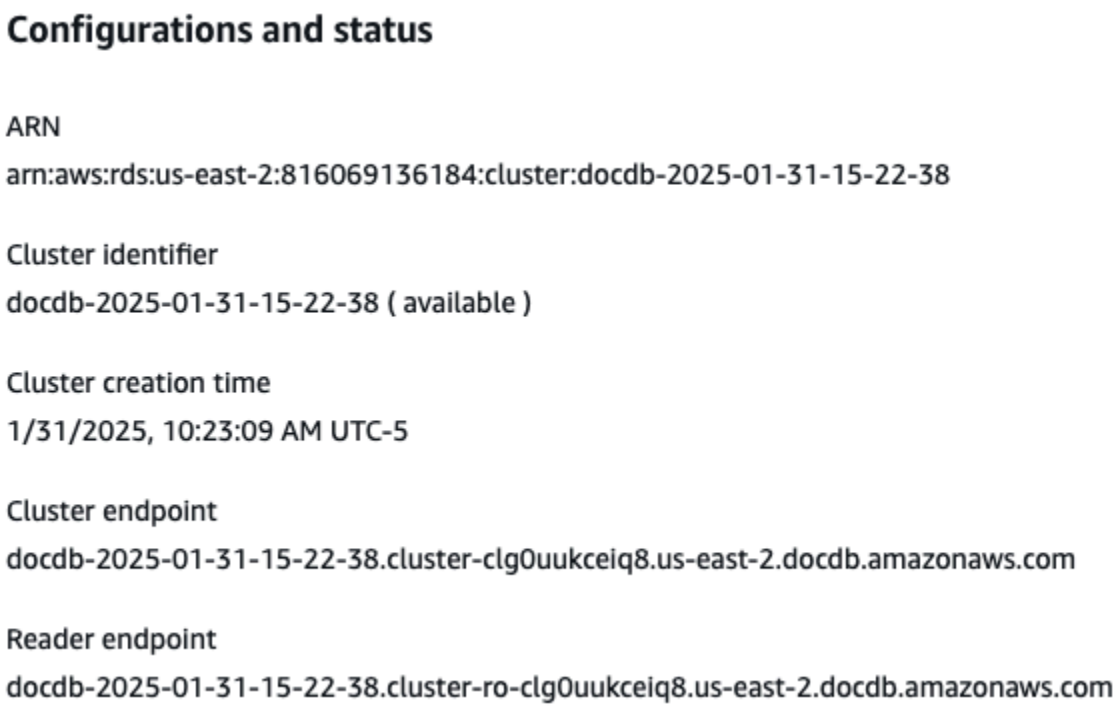
クラスターのエンドポイントの検索

Amazon DocumentDB コンソールまたは AWS CLIを使用して、クラスターのクラスターエンドポイントと読み込みエンドポイントを見つけることができます。

Using the AWS Management Console

コンソールを使用してクラスターのエンドポイントを検索するには：

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択します。
3. クラスターの一覧で、関心のあるクラスターの名前を選択します。
4. クラスターの詳細ページで、設定タブを選択します。「設定とステータス」セクションに、クラスターエンドポイントとリーダーエンドポイントがあります。



5. このクラスターに接続するには、接続とセキュリティタブを選択します。mongo シェルの接続文字列と、クラスターに接続するためにアプリケーションコードで使用できる接続文字列を見つけます。



Using the AWS CLI

を使用してクラスターのクラスターエンドポイントとリーダーエンドポイントを検索するには AWS CLI、これらのパラメータを指定して `describe-db-clusters` コマンドを実行します。

パラメータ

- **`--db-cluster-identifier`** - オプションです。エンドポイントを返すクラスターを指定します。省略した場合、最大 100 個のクラスターのエンドポイントを返します。
- **`--query`** - オプションです。表示するフィールドを指定します。エンドポイントを見つけるために表示する必要があるデータの量を減らすと役立ちます。省略した場合、すべてのクラスターについての情報が返されます。
- **`--region`** - オプションです。`--region` パラメータを使用して、コマンドを適用するリージョンを指定します。省略すると、デフォルトのリージョンが使用されます。

Example

次の例では、`DBClusterIdentifier`、エンドポイント (クラスターエンドポイント)、および `sample-cluster` の `ReaderEndpoint` を返します。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-clusters \
   --region us-east-1 \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

Windows の場合:

```
aws docdb describe-db-clusters ^
```

```
    --region us-east-1 ^
    --db-cluster-identifier sample-cluster ^
    --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

このオペレーションによる出力は、次のようになります（JSON 形式）。

```
[
  [
     "sample-cluster",
     27017,
     "sample-cluster.cluster-corlsfccjozr.us-east-1.docdb.amazonaws.com",
     "sample-cluster.cluster-ro-corlsfccjozr.us-east-1.docdb.amazonaws.com"
  ]
]
```

これでクラスターエンドポイントが作成したので、mongo または mongodb を使用してクラスターに接続できます。詳細については、「エンドポイントへの接続」を参照してください。

インスタンスのエンドポイントの検索

Amazon DocumentDB コンソールまたは AWS CLIを使用して、インスタンスのためのエンドポイントを見つけることができます。

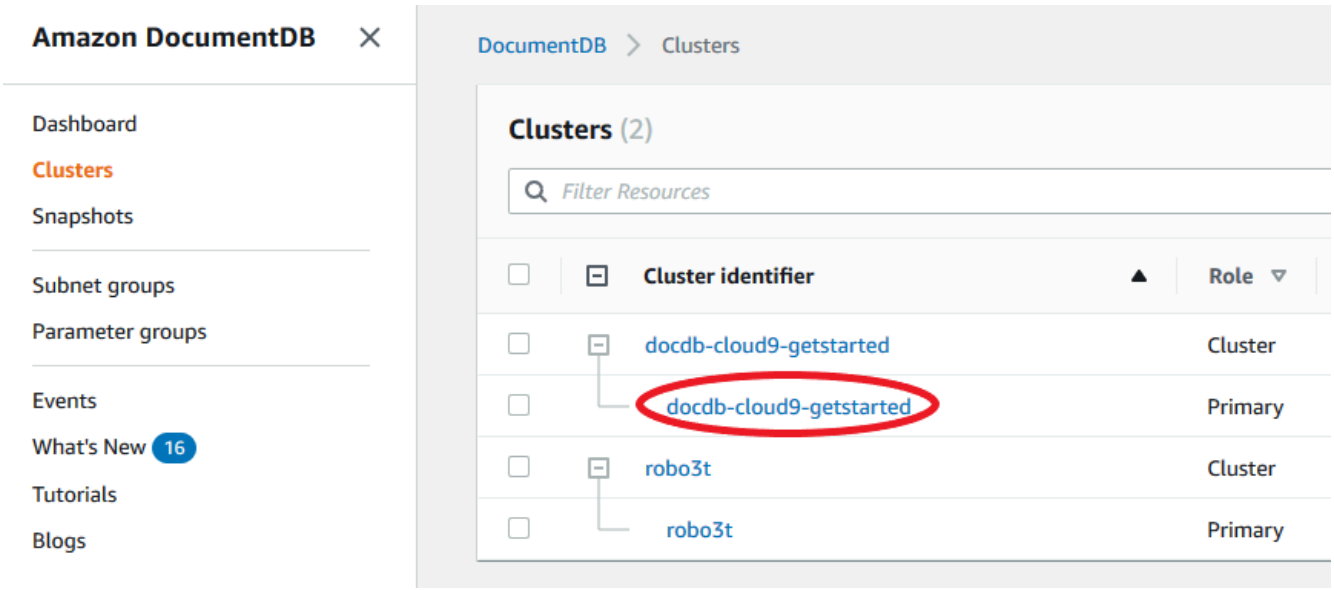
Using the AWS Management Console

コンソールを使用してインスタンスのエンドポイントを見つけるには

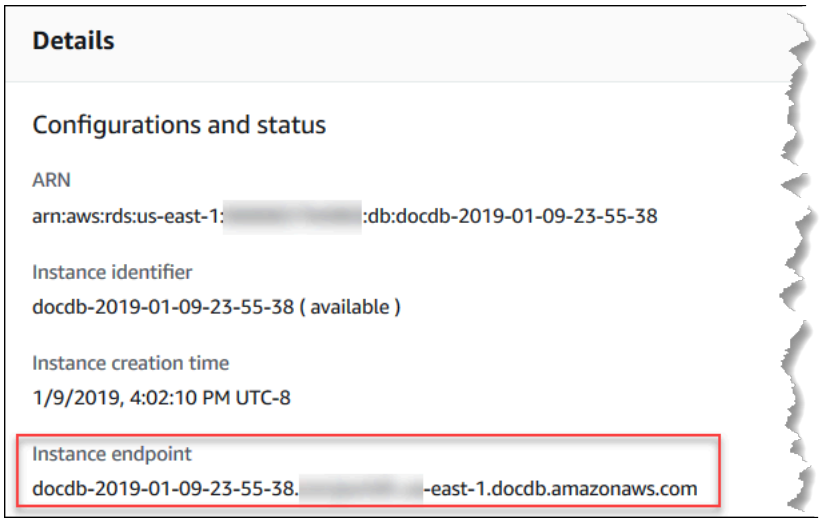
1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで クラスター を選択します。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. クラスターナビゲーションボックスに、[Cluster Identifier]の列が表示されます。インスタンスは、以下のスクリーンショットのように、クラスターの下に表示されます。



4. 興味があるインスタンスの左側にあるチェックボックスをチェックします。
5. [詳細] セクションまで下にスクロールし、インスタンスのエンドポイントを見つけます。



6. このインスタンスに接続するには、[接続] セクションまで上にスクロールします。mongo シェルの接続文字列と、アプリケーションコードで使用してインスタンスに接続できる接続文字列を見つけます。



Using the AWS CLI

を使用してインスタンスエンドポイントを検索するには AWS CLI、以下の引数を指定して次のコマンドを実行します。

引数

- **--db-instance-identifier** - オプションです。エンドポイントを返すインスタンスを指定します。省略した場合、最大 100 個のインスタンスのエンドポイントを返します。
- **--query** - オプションです。表示するフィールドを指定します。エンドポイントを見つけるために表示する必要があるデータの量を減らすと役立ちます。省略した場合、インスタンスのすべての情報が返されます。この Endpoint フィールドには 3 つのメンバーがあるため、次の例に示すようにクエリでリストすると、3 つすべてのメンバーが返されます。一部の Endpoint メンバーのみに関心がある場合は、2 番目の例のように、クエリで Endpoint を関心のあるメンバーに置き換えます。
- **--region** - オプションです。--region パラメータを使用して、コマンドを適用するリージョンを指定します。省略すると、デフォルトのリージョンが使用されます。

Example

Linux、macOS、Unix の場合:

```
aws docdb describe-db-instances \
    --region us-east-1 \
    --db-instance-identifier sample-cluster-instance \
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

Windows の場合:

```
aws docdb describe-db-instances ^
    --region us-east-1 ^
    --db-instance-identifier sample-cluster-instance ^
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
[
    [
```

```
        "sample-cluster-instance",
        {
            "Port": 27017,
            "Address": "sample-cluster-instance.corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
            "HostedZoneId": "Z2R2ITUGPM61AM"
        }
    ]
]
```

出力を減らしてエンドポイントの HostedZoneId を排除すると、Endpoint.Port と Endpoint.Address を使用してクエリを変更することができます。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-instances \
    --region us-east-1 \
    --db-instance-identifier sample-cluster-instance \
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint.Port,Endpoint.Address]'
```

Windows の場合:

```
aws docdb describe-db-instances ^
    --region us-east-1 ^
    --db-instance-identifier sample-cluster-instance ^
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint.Port,Endpoint.Address]'
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
[
    [
        "sample-cluster-instance",
        27017,
        "sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com"
    ]
]
```

これでインスタンスエンドポイントを作成したので、mongo または mongodb を使用してインスタンスに接続できます。詳細については、「エンドポイントへの接続」を参照してください。

エンドポイントへの接続

エンドポイント (クラスターまたはインスタンス) が作成されたら、mongo シェルまたは接続文字列を使用して接続できます。

mongo シェルを使用した接続

以下の構造を使用して、mongo シェルを使ってクラスターまたはインスタンスに接続する必要がある文字列を作成します。

```
mongo \
    --ssl \
    --host Endpoint:Port \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

mongo シェルの例

クラスターへの接続:

```
mongo \
    --ssl \
    --host sample-cluster.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

インスタンスへの接続:

```
mongo \
    --ssl \
    --host sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

接続文字列を使用した接続

以下の構造を使用して、クラスターまたはインスタンスに接続する必要がある接続文字列を作成します。

```
mongodb://UserName:Password@endpoint:port?replicaSet=rs0&ssl_ca_certs=global-
bundle.pem
```

接続文字列の例

クラスターへの接続:

```
mongodb://UserName:Password@sample-cluster.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

インスタンスへの接続:

```
mongodb://UserName:Password@sample-cluster-instance.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

Amazon DocumentDB Amazon リソースネーム (ARN) の理解

で作成したリソース AWS は、それぞれ Amazon リソースネーム (ARN) で一意に識別されます。特定の Amazon DocumentDB (MongoDB 互換) オペレーションでは、ARN を指定して、Amazon DocumentDB リソースを一意に識別する必要があります。たとえば、リソースにタグを追加するときは、リソースの ARN を指定する必要があります。

トピック

- Amazon DocumentDB リソースの ARN の設定
- Amazon DocumentDB リソースの ARN を検索する

Amazon DocumentDB リソースの ARN の設定

次の構文を使用して Amazon DocumentDB リソースの ARN を構築できます。Amazon DocumentDB は、Amazon Relational Database Service (Amazon RDS) ARNs。Amazon DocumentDB ARN には `rds` を含み、`docdb` は含みません。

`arn:aws:rds:region:account_number:resource_type:resource_id`

リージョン名	リージョン	アベイラビリティーゾーン (コンピューティング)
米国東部 (オハイオ)	us-east-2	3
米国東部 (バージニア北部)	us-east-1	6
米国西部 (オレゴン)	us-west-2	4
アフリカ (ケープタウン)	af-south-1	3
南米 (サンパウロ)	sa-east-1	3
アジアパシフィック (香港)	ap-east-1	3
アジアパシフィック (ハイデラバード)	ap-south-2	3
アジアパシフィック (ムンバイ)	ap-south-1	3
アジアパシフィック (ソウル)	ap-northeast-2	4
アジアパシフィック (シンガポール)	ap-southeast-1	3
アジアパシフィック (シドニー)	ap-southeast-2	3
アジアパシフィック (東京)	ap-northeast-1	3
カナダ (中部)	ca-central-1	3
中国 (北京) リージョン	cn-north-1	3

リージョン名	リージョン	アベイラビリティーゾーン (コンピューティング)
中国 (寧夏)	cn-northwest-1	3
欧州 (フランクフルト)	eu-central-1	3
欧州 (アイルランド)	eu-west-1	3
欧州 (ロンドン)	eu-west-2	3
欧州 (ミラノ)	eu-south-1	3
欧州 (パリ)	eu-west-3	3
欧州 (スペイン)	eu-south-2	3
欧州 (ストックホルム)	eu-north-1	3
中東 (アラブ首長国連邦)	me-central-1	3
イスラエル (テルアビブ)	il-central-1	3
AWS GovCloud (米国西部)	us-gov-west-1	3
AWS GovCloud (米国東部)	us-gov-east-1	3

Note

Amazon DocumentDB アーキテクチャは、ストレージとコンピューティングを分離します。ストレージレイヤーの場合、Amazon DocumentDB は 3 つの AWS アベイラビリティーゾーン (AZs) にデータの 6 つのコピーをレプリケートします。上記の表にリストされている AZ は、特定のリージョンでコンピューティングインスタンスをプロビジョニングするために使用できる AZ の数です。例えば、ap-northeast-1 で Amazon DocumentDB クラスターを起動

している場合、ストレージは 3 つの AZ にわたって 6 つの方法でレプリケートされますが、コンピューティングインスタンスは 2 つの AZ でしか使用できません。

次の表に、特定の Amazon DocumentDB リソースの ARN の構築時に使用する形式を示します。Amazon DocumentDB は Amazon RDS ARNs。Amazon DocumentDB ARN には `rds` を含み、`docdb` は含みません。

リソースタイプ	ARN 形式/例
インスタンス (db)	`arn:aws:rds:` *region*`:`*account_number*`:db:`*resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:db:`*sample-db-instance*
クラスター (cluster)	`arn:aws:rds:` *region*`:`*account_number*`:cluster:` *resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:cluster:` *sample-db-cluster*
クラスターパラメータグループ (cluster-pg)	`arn:aws:rds:` *region*`:`*account_number*`:cluster-pg:` *resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:cluster-pg:` *sample-db-cluster-parameter-group*
セキュリティグループ (secgrp)	`arn:aws:rds:` *region*`:`*account_number*`:secgrp:`*resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:secgrp:`*sample-public-secgrp*
クラスタースナップショット (cluster-snapshot)	`arn:aws:rds:` *region*`:`*account_number*`:cluster-snapshot:` *resource_id*

リソースタイプ	ARN 形式/例
	`arn:aws:rds:us-east-1:` *1234567890* `:cluster-snapshot:` *sample-db-cluster-snapshot*
サブネットグループ (subgrp)	`arn:aws:rds:` *region*`:`*account_number*`:subgrp:`*resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:subgrp:`*sample-subnet-10*

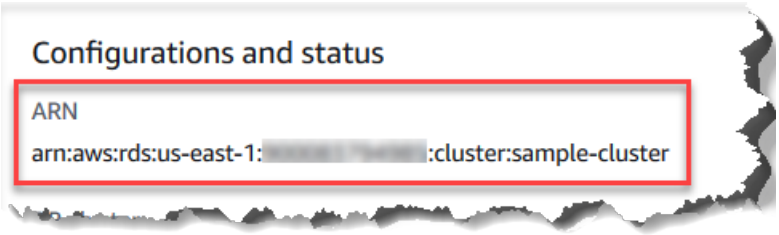
Amazon DocumentDB リソースの ARN を検索する

Amazon DocumentDB リソースの ARN は、 AWS Management Console または を使用して確認できます AWS CLI。

Using the AWS Management Console

コンソールを使用して ARN を検索するには、ARN を取得するリソースに移動し、そのリソースの詳細を表示します。

たとえば、クラスターの詳細ページで設定タブを選択すると、クラスターの ARN を取得できます。ARN は、次のスクリーンショットに示すように、設定とステータスセクションにあります。



Using the AWS CLI

特定の Amazon DocumentDB リソース AWS CLI の を使用して ARN を取得するには、そのリソースの `describe`オペレーションを使用します。次の表は、各 AWS CLI オペレーションと、ARN を取得するオペレーションで使用される ARN プロパティを示しています。

AWS CLI コマンド	ARN プロパティ
`describe-db-instances`	`DBInstanceArn`

AWS CLI コマンド	ARN プロパティ
describe-db-clusters	DBClusterArn
describe-db-parameter-groups	DBParameterGroupArn
describe-db-cluster-parameter-groups	DBClusterParameterGroupArn
describe-db-security-groups	DBSecurityGroupArn
describe-db-snapshots	DBSnapshotArn
describe-db-cluster-snapshots	DBClusterSnapshotArn
describe-db-subnet-groups	DBSubnetGroupArn

Example - クラスターの ARN の検索

次の AWS CLI オペレーションでは、クラスター の ARN を検索しますsample-cluster。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterArn'
```

Windows の場合:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterArn'
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
[
    "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster"
]
```

Example - 複数のパラメータグループの ARN の検索

Linux、macOS、Unix の場合:

```
aws docdb describe-db-cluster-parameter-groups \
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

Windows の場合:

```
aws docdb describe-db-cluster-parameter-groups ^
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
[
    "arn:aws:rds:us-east-1:123456789012:cluster-pg:custom3-6-param-grp",
    "arn:aws:rds:us-east-1:123456789012:cluster-pg:default.aurora5.6",
    "arn:aws:rds:us-east-1:123456789012:cluster-pg:default.docdb3.6"
]
```

Amazon DocumentDB リソースへのタグ付け

Amazon DocumentDB (MongoDB 互換) タグを使用して Amazon DocumentDB リソースにメタデータを追加できます。これらのタグを AWS Identity and Access Management (IAM) ポリシーとともに使用して、Amazon DocumentDB リソースへのアクセスを管理し、リソースに適用できるアクションを制御できます。また、タグを使用して、類似のリソースの費用をグループ化することで、コストを追跡できます。

次の Amazon DocumentDB リソースにタグ付けができます。

- クラスター
- インスタンス
- スナップショット
- クラスタースナップショット
- パラメータグループ
- クラスターパラメータグループ
- セキュリティグループ

- [サブネットグループ]

Amazon DocumentDB リソースタグの概要

Amazon DocumentDB タグは、Amazon DocumentDB リソースを定義してそのリソースに関連付ける名前と値のペアです。その名前はキーと呼ばれます。キーの値の指定は省略可能です。タグを使用して、Amazon DocumentDB リソースに任意の情報を割り当てることができます。例えば、タグキーを使用してカテゴリを定義し、タグ値をそのカテゴリのアイテムにすることができます。具体的には、`project` というタグキーと `Salix` というタグ値を定義して、Amazon DocumentDB リソースが Salix プロジェクトに割り当てられていることを示すことができます。また、`environment=test` や `environment=production` などのタグキーを使用して Amazon DocumentDB リソースがテスト用であるか本番稼働用であるかを示すこともできます。Amazon DocumentDB リソースに関連付けられたメタデータ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。

タグを使用して、独自のコスト構造を反映するように AWS 請求書を整理できます。これを行うには、サインアップしてタグキー値を含む AWS アカウント 請求書を取得します。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。詳細については、AWS Billing and Cost Management ユーザーガイド の コスト配分タグを使用する を参照してください。

各 Amazon DocumentDB リソースにはタグセットがあり、それぞれのリソースに割り当てられているすべてのタグが含まれています。タグセットには最大 10 個のタグを含めることができ、空にすることもできます。Amazon DocumentDB リソースに追加したタグのキーがそのリソースの既存のタグのキーと同じ場合、既存の値は新しい値によって上書きされます。

AWS はタグに意味論的意味を適用しません。タグは文字列として厳密に解釈されます。リソース作成時に使用する設定によっては、Amazon DocumentDB によってインスタンスまたは他の Amazon DocumentDB リソースにタグが設定されることがあります。例えば、Amazon DocumentDB によってインスタンスが本稼働用またはテスト用であることを示すタグが追加されることがあります。

スナップショットにタグを追加することはできますが、請求書にはこのグループが反映されません。

AWS Management Console または を使用して、Amazon DocumentDB リソースのタグ AWS CLI を追加、一覧表示、削除できます。を使用する場合は AWS CLI、使用するリソースの Amazon リソースネーム (ARN) を指定する必要があります。Amazon DocumentDB ARN の詳細については、Amazon DocumentDB Amazon リソースネーム (ARN) の理解 を参照してください。

タグの制約

Amazon DocumentDB タグには以下の制限があります。

- リソースあたりのタグの最大数は 10 です
- キーの最大長 - 128 文字 (Unicode)
- 値の最大長 - 256 文字 (Unicode)
- [キー] と [値] に使用できる文字 - アルファベットの大文字と小文字 (UTF-8 文字セット)、数字、スペース、次の文字です (_ . : / = + - および @) (Java 正規表現: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$")。
- タグのキーと値では、大文字と小文字が区別されます。
- プレフィックス aws: をタグキーや値に使用することはできません。 AWS用に予約されています。

Amazon DocumentDB リソースでのタグの追加と更新

AWS Management Console または を使用して、リソースに最大 10 個のタグを追加できます AWS CLI。

Using the AWS Management Console

リソースにタグを追加するプロセスは、タグの追加先となるリソースにかかわらず、似ています。この例では、クラスターにタグを追加します。

コンソールを使用してクラスターにタグを追加または更新するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。
3. タグを追加するクラスターの名前を指定します。
4. [タグ] セクションまで下にスクロールし、[編集] を選択します。
5. このリソースに追加するタグごとに、以下が必要になります。
    a. 新しいタグを追加するには、[キー] ボックスにタグの名前を入力します。タグの値を変更するには、[キー] 列でタグの名前を見つけます。
    b. 新規または更新された値にタグに付けるには、[値] ボックスにタグの値を入力します。

c. さらにタグを追加する場合は、[追加] を選択します。それ以外の場合は、完了したら、[保存] を選択します。

Using the AWS CLI

リソースにタグを追加するプロセスは、タグの追加先となるリソースにかかわらず、似ています。この例では、クラスターに 3 つのタグを追加します。2 番目のタグ key2 には値がありません。

これらのパラメータadd-tags-to-resourceで AWS CLI オペレーションを使用します。

パラメータ

- **--resource-name** — タグを追加する Amazon DocumentDB リソースの ARN。
- **--tags** Key=*key-name*,Value=*tag-value* 形式でこのリソースに追加するタグ (キーと値のペア) のリスト。

Example

Linux、macOS、Unix の場合:

```
aws docdb add-tags-to-resource \
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tags Key=key1,Value=value1 Key=key2 Key=key3,Value=value3
```

Windows の場合:

```
aws docdb add-tags-to-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tags Key=key1,Value=value1 Key=key2 Key=key3,Value=value3
```

add-tags-to-resource オペレーションでは何も出力されません。オペレーションの結果を表示するには、list-tags-for-resource オペレーションを使用します。

Amazon DocumentDB リソースでのタグのリスト表示

AWS Management Console または を使用して AWS CLI 、Amazon DocumentDB リソースのタグのリストを取得できます。

Using the AWS Management Console

リソースのタグをリスト表示するプロセスは、タグの追加先となるリソースにかかわらず、似ています。この例では、クラスターのタグをリストします。

コンソールを使用してクラスター上のタグをリスト表示するには

1. Amazon DocumentDB コンソールを、次の場所で開きます。https://console.aws.amazon.com/docdb
2. ナビゲーションペインで、[クラスター] を選択します。
3. タグをリスト表示するクラスターの名前を指定します。
4. このリソースのタグをリスト表示するには、[タグ] セクションまで下にスクロールします。

Using the AWS CLI

リソースのタグをリスト表示するプロセスは、タグをリスト表示するリソースにかかわらず、似ています。この例では、クラスターのタグをリストします。

これらのパラメータ`list-tags-for-resource`で AWS CLI オペレーションを使用します。

パラメータ

- **`--resource-name`** — 必須 タグを追加する Amazon DocumentDB リソースの ARN。

Example

Linux、macOS、Unix の場合:

```
aws docdb list-tags-for-resource \
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

Windows の場合:

```
aws docdb list-tags-for-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
```

```
    "TagList": [
        {
            "Key": "key1",
            "Value": "value1"
        },
        {
            "Key": "key2",
            "Value": ""
        },
        {
            "Key": "key3",
            "Value": "value3"
        }
    ]
}
```

Amazon DocumentDB リソースからのタグの削除

AWS Management Console または を使用して AWS CLI 、Amazon DocumentDB リソースからタグを削除できます。

Using the AWS Management Console

リソースからタグを削除するプロセスは、タグの追加先となるリソースにかかわらず、似ています。この例では、クラスターからタグを削除します。

コンソールを使用してクラスターからタグを削除するには

1. Amazon DocumentDB コンソールを、次の場所で開きます。https://console.aws.amazon.com/docdb
2. ナビゲーションペインで、[クラスター] を選択します。
3. タグを削除するクラスターの名前を指定します。
4. [タグ] セクションまで下にスクロールし、[編集] を選択します。
5. このリソースからすべてのタグを削除するには、[すべて削除] を選択します。それ以外の場合、このリソースから削除するタグごとに、以下を実行します。

    a. [キー] 列で、タグの名前を見つけます。

    b. タグキーと同じ行で [削除] を選択します。

    c. 完了したら、[保存] を選択します。

Using the AWS CLI

リソースからタグを削除するプロセスは、タグの削除元となるリソースにかかわらず、似ています。この例では、クラスターからタグを削除します。

これらのパラメータremove-tags-from-resourceで AWS CLI オペレーションを使用します。

- **--resource-name** — 必須 タグを追加する Amazon DocumentDB リソースの ARN。
- **--tag-keys** — 必須 このリソースから削除するタグキーのリスト。

Example

Linux、macOS、Unix の場合:

```
aws docdb remove-tags-from-resource \
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tag-keys key1 key3
```

Windows の場合:

```
aws docdb remove-tags-from-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tag-keys key1 key3
```

removed-tags-from-resource オペレーションでは何も出力されません。オペレーションの結果を表示するには、list-tags-for-resource オペレーションを使用します。

Amazon DocumentDB のメンテナンス

Amazon DocumentDB は定期的に、Amazon DocumentDB リソースのメンテナンスを行います。通常、メンテナンスにはデータベースエンジンの更新（クラスターのメンテナンス）やインスタンスの基盤となるオペレーティングシステム（OS）の更新（インスタンスのメンテナンス）が伴います。データベースエンジンの更新は必須のパッチであり、セキュリティ修正、バグ修正、データベースエンジンの強化などが含まれます。一方ほとんどのオペレーティングシステムパッチはオプションですが、しばらく適用しないで放置していると、セキュリティ体制の維持のためにパッチが必要となった結果、自動的に適用されることがあります。そのため、Amazon DocumentDB インスタンス向けのオペレーティングシステム更新が利用可能になったらすぐに適用することをお勧めします。

データベースエンジンパッチの適用には、Amazon DocumentDB クラスターを短時間オフラインにする必要があります。パッチが利用可能になるたびに、Amazon DocumentDB クラスターの次のメンテナンスウィンドウでの実施予定に組み込まれ、自動的に適用されます。

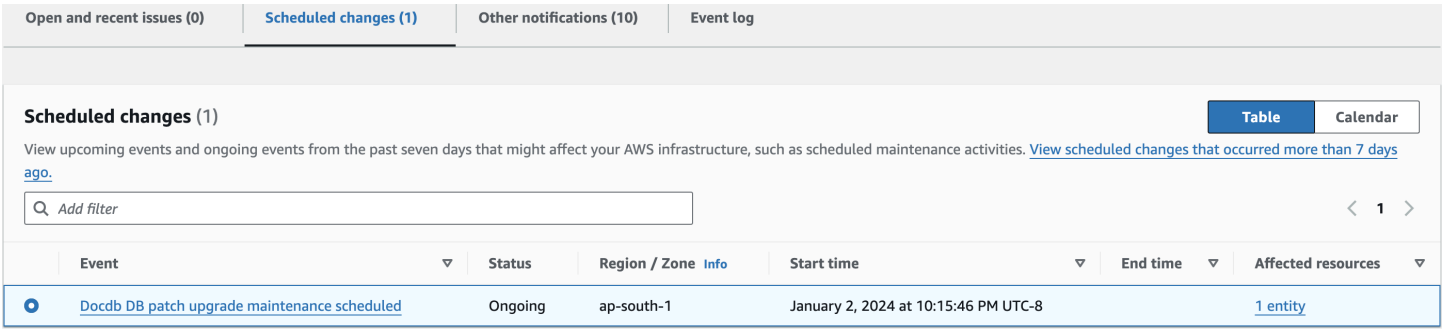
クラスターとインスタンスの両方のメンテナンスに、それぞれ独自のメンテナンスウィンドウがあります。クラスターやインスタンスの修正をすぐには適用しないことを選択した場合でも、メンテナンスウィンドウ中に自動的に適用されます。デフォルトでは、クラスターを作成すると、Amazon DocumentDB はクラスターと各個別のインスタンスの両方にメンテナンスウィンドウを割り当てます。クラスターまたはインスタンスを作成するときにメンテナンスウィンドウを選択できます。また、ビジネススケジュールや業務に合わせてメンテナンスウィンドウをいつでも変更できます。一般的には、メンテナンスによるアプリケーションへの影響を最小限に抑えるメンテナンスウィンドウ（夜間、週末など）を選択することをお勧めします。

トピック

- Amazon DocumentDB エンジンパッチの通知
- 実施待ちの Amazon DocumentDB メンテナンスアクションを確認する
- Amazon DocumentDB エンジンの更新
- ユーザーが起動する更新
- Amazon DocumentDB メンテナンスウィンドウの管理する
- Amazon DocumentDB オペレーティングシステムの更新

Amazon DocumentDB エンジンパッチの通知

コンソールの AWS Health Dashboard (AHD) のヘルスイベント AWS と E メールを通じて、必要なデータベースエンジンパッチのメンテナンス通知が送信されます。Amazon DocumentDB エンジンメンテナンスパッチが特定の AWS リージョンで利用可能になると、リージョン内の影響を受けるすべての Amazon DocumentDB ユーザーアカウントに、パッチの影響を受ける各 Amazon DocumentDB バージョンの AHD と E メール通知が送信されます。これらの通知は、 AWS コンソールの AHD のスケジュールされた変更セクションで表示できます。通知には、パッチが利用可能になるタイミング、自動適用スケジュール、影響を受けるクラスターのリスト、およびリリースノートが記載されます。この通知は、 AWS アカウントのルートユーザーの E メールアドレスにも E メールで配信されます。



このような通知を受け取り次第、スケジュールされた自動適用日より前に、Amazon DocumentDB クラスターにエンジンパッチを手動で適用することができます。または、予定通り次回のメンテナンスウィンドウ (デフォルトオプション) でエンジンパッチの自動適用を受けることもできます。

> **Note**
>
> AHD 通知の [ステータス]は、新しいエンジンパッチバージョンの Amazon DocumentDB エンジンパッチがリリースされるまでは [進行中] と表示されます。
> エンジンパッチが Amazon DocumentDB クラスターに適用されると、クラスターのエンジンパッチバージョンが更新され、通知対象のバージョン番号が反映されます。`db.runCommand({getEngineVersion: 1})` コマンドを実行して、更新が行われたかどうかを確認できます。

AWS Health は、イベントを使用してスケーラブルなイベント駆動型アプリケーションを構築し、Amazon Simple Queue Service (SQS) など AWS Lambda 20 を超えるターゲットと統合する Amazon EventBridge とも統合されています。エンジンパッチが利用可能になる前に、`AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_SCHEDULED` イベントコードを使用して Amazon EventBridge をセットアップできます。EventBridge を設定して、イベントに応答し、イベント情報のキャプチャ、追加のイベントの開始、 へのプッシュ通知などの追加のチャネルを介した通知の送信 AWS Console Mobile Application、Amazon DocumentDB エンジンパッチがリージョンで利用可能になったときの是正措置やその他のアクションを実行するなどのアクションを自動実行できます。

Amazon DocumentDB がエンジンパッチをキャンセルするまれなシナリオでは、AHD 通知とキャンセルを通知する E メールが送信されます。`AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_CANCELLED` イベントコードの使用により、このようなイベントにも対応するように Amazon EventBridge を設定できます。Amazon EventBridge ルール運用の詳細については、「Amazon EventBridge ユーザーガイド」を参照してください。

実施待ちの Amazon DocumentDB メンテナンスアクションを確認する

AWS Management Console または を使用して、クラスターでメンテナンス更新が利用可能かどうかを確認できます AWS CLI。

更新が利用可能な場合は、以下のいずれかの操作を行います。

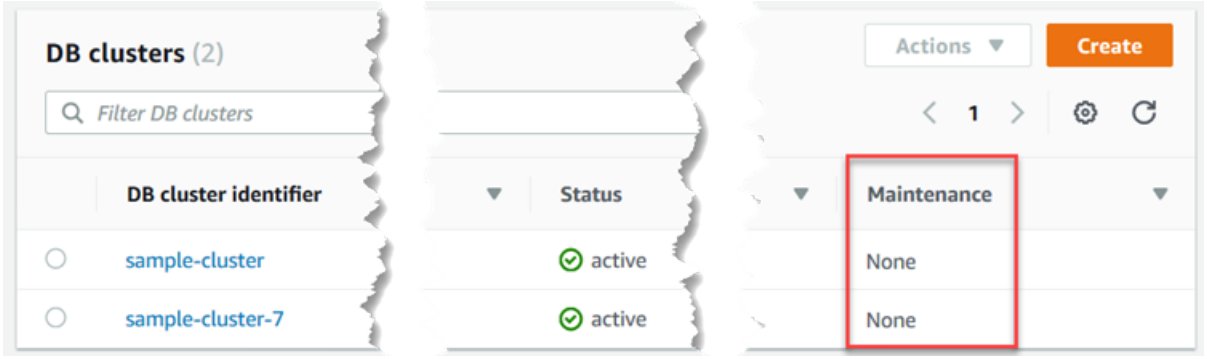
- 現在次のメンテナンスウィンドウにスケジュールされているメンテナンスアクションを延期することができます (OS パッチのみ)。
- メンテナンスアクションをすぐに適用する。
- メンテナンスアクションを次のメンテナンスウィンドウ中に開始するようにスケジュールを設定する。

> **Note**
>
> 特に対応を行わない場合、エンジンパッチなどの必須メンテナンスアクションは、次のメンテナンスウィンドウ時に自動的に適用されます。

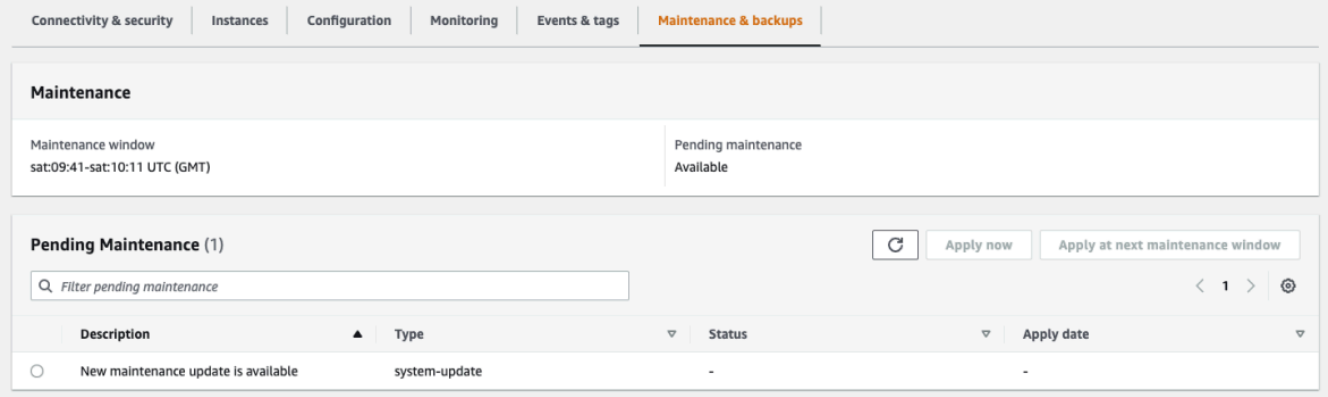
メンテナンスウィンドウによって、保留中のオペレーションを開始する時刻が決定されますが、これらのオペレーションの合計実行時間は制限されません。

Using the AWS Management Console

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. 更新が可能な場合、ここに示すように [使用可能]、[必須]、または [次のウィンドウ] という文字で、Amazon DocumentDB コンソールのクラスターのメンテナンス列に表示されます。



4. アクションを実行するには、クラスターを選択してその詳細を表示し、次に [メンテナンス & バックアップ] を選択します。[保留中のメンテナンス] 項目が表示されます。



Using the AWS CLI

次の AWS CLI オペレーションを使用して、保留中のメンテナンスアクションを決定します。以下の出力は、保留中のメンテナンスアクションがないことを示しています。

```
aws docdb describe-pending-maintenance-actions
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "PendingMaintenanceActions": []
}
```

Amazon DocumentDB エンジンの更新

Amazon DocumentDB では、メンテナンスオペレーションを適用するタイミングを選択することができます。Amazon DocumentDB が更新を適用するタイミングは、 AWS Management Console または を使用して決定できます AWS CLI。

このトピックの手順を使用して、クラスターをすぐにアップグレードするか、アップグレードをスケジュールします。

Using the AWS Management Console

コンソールを使用して、Amazon DocumentDB クラスターの更新を管理することができます。

クラスターの更新を管理するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. クラスターのリストで、メンテナンスオペレーションを適用するクラスターの名前の横にあるボタンを選択します。
4. [アクション] メニューで以下のいずれかを選択します。

   - 保留中のメンテナンスタスクをすぐに実行する場合は、[今すぐアップグレード]。
   - クラスターの次のメンテナンスウィンドウで保留中のメンテナンスタスクを実行する場合は、[次のウィンドウでアップグレード]。

   または、クラスターのメンテナンスとバックアップタブの「メンテナンス保留中」セクションにある「今すぐ適用」または次のメンテナンスウィンドウで適用」をクリックすることもできます (前のセクションの「 の使用 AWS Management Console」を参照) 。

   > **Note**
   >
   > 保留中のメンテナンスタスクがない場合、上記のオプションはすべて非アクティブになります。

Using the AWS CLI

保留中の更新をクラスターに適用するには、 `apply-pending-maintenance-action` AWS CLI オペレーションを使用します。

パラメータ

- **`--resource-identifier`** - 保留中のメンテナンスアクションが適用されるリソースの Amazon DocumentDB ARN (Amazon リソースネーム)です。
- **`--apply-action`** - このリソースに適用する保留中のメンテナンスアクションです。

  有効な値: `system-update` および `db-upgrade`。
- **`--opt-in-type`** - オプトインリクエストのタイプを指定するか、オプトインクエストを元に戻す値です。タイプ `immediate` のオプトインリクエストは元に戻すことができません。

有効な値:

- immediate - メンテナンスアクションをすぐに適用します。
- next-maintenance - リソースの次のメンテナンスウィンドウ中にメンテナンスアクションを適用します。
- undo-opt-in - 既存の next-maintenance オプトインリクエストをキャンセルします。

Example

Linux、macOS、Unix の場合:

```
aws docdb apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-east-1:123456789012:db:docdb \
    --apply-action system-update \
    --opt-in-type immediate
```

Windows の場合:

```
aws docdb apply-pending-maintenance-action ^
    --resource-identifier arn:aws:rds:us-east-1:123456789012:db:docdb ^
    --apply-action system-update ^
    --opt-in-type immediate
```

少なくとも 1 つの保留中の更新があるリソースのリストを返すには、 describe-pending-maintenance-actions AWS CLI コマンドを使用します。

Example

Linux、macOS、Unix の場合:

```
aws docdb describe-pending-maintenance-actions \
    --resource-identifier arn:aws:rds:us-east-1:001234567890:db:docdb
```

Windows の場合:

```
aws docdb describe-pending-maintenance-actions ^
    --resource-identifier arn:aws:rds:us-east-1:001234567890:db:docdb
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "PendingMaintenanceActions": [
        {
            "ResourceIdentifier": "arn:aws:rds:us-
east-1:001234567890:cluster:sample-cluster",
            "PendingMaintenanceActionDetails": [
                {
                    "Action": "system-update",
                    "CurrentApplyDate": "2019-01-11T03:01:00Z",
                    "Description": "db-version-upgrade",
                    "ForcedApplyDate": "2019-01-18T03:01:00Z",
                    "AutoAppliedAfterDate": "2019-01-11T03:01:00Z"
                }
            ]
        }
    ]
}
```

describe-pending-maintenance-actions AWS CLI オペレーションの --filtersパラメータを指定して、クラスターのリソースのリストを返すこともできます。--filters オペレーションの形式は、Name=*filter-name*,Values=*resource-id*,... です。

db-cluster-id は、フィルターの Name パラメータの許容値です。この値では、クラスター識別子または ARN のリストが許容されます。返されるリストには、これらの識別子または ARN で識別されたクラスターの保留中のメンテナンスアクションのみが含まれます。

次の例では、sample-cluster1 クラスターと sample-cluster2 クラスターの保留中のメンテナンスアクションが返されます。

Example

Linux、macOS、Unix の場合:

```
aws docdb describe-pending-maintenance-actions \
   --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Windows の場合:

```
aws docdb describe-pending-maintenance-actions ^
   --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

日付の適用

各メンテナンスアクションにはそれぞれ、保留中のメンテナンスアクションを記述する際に確認できる適用日が設定されています。から保留中のメンテナンスアクションの出力を読み取ると AWS CLI、3 つの日付が表示されます。これらの日付値は、メンテナンスがオプションnullの場合です。対応するメンテナンスアクションがスケジュールまたは適用されると、値が入力されます。

- **CurrentApplyDate** - メンテナンスアクションが即座に、または次のメンテナンスウィンドウに適用される日付です。
- **ForcedApplyDate** - メンテナンスウィンドウとは関係なく、メンテナンスが自動的に適用される日付です。
- **AutoAppliedAfterDate** - クラスターのメンテナンスウィンドウでメンテナンスが適用される日付です。

ユーザーが起動する更新

Amazon DocumentDB のユーザーとして、クラスターまたはインスタンスの更新を開始することができます。例えば、インスタンスのクラスをより多くのメモリやより少ないメモリに変更したり、クラスターのパラメータグループを変更したりすることができます。Amazon DocumentDB は、これらの変更を、Amazon DocumentDB が開始した更新とは異なる方法で表示します。クラスターまたはインスタンスの変更の詳細については、以下を参照してください。

- Amazon DocumentDB クラスターの変更
- Amazon DocumentDB インスタンスの変更

ユーザーが開始した変更のうち、保留中の変更リストを確認するには、次のコマンドを実行します。

Example

ユーザーが開始したインスタンスの変更のうち保留中のものを確認するには

Linux、macOS、Unix の場合:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

Windows の場合:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

この場合、sample-cluster-instance には db.r5.xlarge インスタンスクラスに対する保留中の変更があり、sample-cluster-instance-2 には保留中の変更はありません。

```
[
    [
        "sample-cluster",
        "sample-cluster-instance",
        {
            "DBInstanceClass": "db.r5.xlarge"
        }
    ],
    [
        "sample-cluster",
        "sample-cluster-instance-2",
        {}
    ]
]
```

Amazon DocumentDB メンテナンスウィンドウの管理する

各インスタンスおよびクラスターには週次のメンテナンスウィンドウがあり、そのウィンドウ内に保留中の変更が適用されます。メンテナンスウィンドウは、変更やソフトウェアのパッチ適用がリクエストされた場合や必要となった場合に、いつ実行するかを制御できる機会です。メンテナンスイベントを特定の週に予定した場合、そのイベントはユーザーが指定した 30 分のメンテナンスウィンドウ中に開始されます。ほとんどのメンテナンスイベントは 30 分のメンテナンスウィンドウ中に完了しますが、大規模なメンテナンスイベントは 30 分以上かかる場合があります。

30 分のメンテナンスウィンドウは、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。インスタンス、またはクラスターの作成時に優先するメンテナンスウィンドウを指定しない場合、Amazon DocumentDB はランダムに選択された曜日の 30 分のメンテナンスウィンドウを割り当てます。

次の表に、デフォルトでメンテナンスウィンドウ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。

リージョン名	リージョン	UTC 時間帯
米国東部 (オハイオ)	us-east-2	03:00 ~ 11:00
米国東部 (バージニア北部)	us-east-1	03:00 ~ 11:00
米国西部 (オレゴン)	us-west-2	06:00 ~ 14:00
アフリカ (ケープタウン)	af-south-1	03:00 ~ 11:00
アジアパシフィック (香港)	ap-east-1	06:00 ~ 14:00
アジアパシフィック (ハイデラバード)	ap-south-2	06:30 ~ 14:30
アジアパシフィック (ムンバイ)	ap-south-1	06:00 ~ 14:00
アジアパシフィック (ソウル)	ap-northeast-2	13:00 ~ 21:00
アジアパシフィック (シンガポール)	ap-southeast-1	14:00 ~ 22:00
アジアパシフィック (シドニー)	ap-southeast-2	12:00 ~ 20:00
アジアパシフィック (東京)	ap-northeast-1	13:00 ~ 21:00
カナダ (中部)	ca-central-1	03:00 ~ 11:00
中国 (北京)	cn-north-1	06:00 ~ 14:00
中国 (寧夏)	cn-northwest-1	06:00 ~ 14:00
欧州 (フランクフルト)	eu-central-1	21:00 ~ 05:00
欧州 (アイルランド)	eu-west-1	22:00 ~ 06:00
欧州 (ロンドン)	eu-west-2	22:00 ~ 06:00
欧州 (ミラノ)	eu-south-1	02:00 ~ 10:00

リージョン名	リージョン	UTC 時間帯
欧州 (パリ)	eu-west-3	23:59 ~ 07:29
欧州 (スペイン)	eu-south-2	02:00 ~ 10:00
欧州 (ストックホルム)	eu-north-1	4:00 ~ 12:00
中東 (アラブ首長国連邦)	me-central-1	05:00 ~ 13:00
南米 (サンパウロ)	sa-east-1	00:00 ~ 08:00
イスラエル (テルアビブ)	il-central-1	04:00 ~ 12:00
AWS GovCloud (米国東部)	us-gov-east-1	17:00 ~ 01:00
AWS GovCloud (米国西部)	us-gov-west-1	06:00 ~ 14:00

Amazon DocumentDB メンテナンスウィンドウを変更する

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。この間、クラスターまたはインスタンスは、システムの変更 (スケールストレージオペレーションやインスタンスクラスの変更など) が適用されていて停止が必要な場合に限り、使用できません。また、必要な変更を加えるのに必要な最小限の時間のみ使用不可となります。

データベースエンジンのアップグレードのために、Amazon DocumentDB は、個々のインスタンスのメンテナンスウィンドウではなく、クラスターの優先的なメンテナンスウィンドウを使用します。

メンテナンスウィンドウを変更するには

- クラスターの場合には、「Amazon DocumentDB クラスターの変更」を参照してください。
- インスタンスの場合には、「Amazon DocumentDB インスタンスの変更」を参照してください。

Amazon DocumentDB オペレーティングシステムの更新

Amazon DocumentDB クラスターのインスタンスでは、オペレーティングシステムの更新が必要になる時があります。Amazon DocumentDB は、データベースパフォーマンスと顧客の全体的なセキュリティ体制改善のために、オペレーティングシステムを新しいバージョンにアップグレードしま

す。オペレーティングシステムの更新では、Amazon DocumentDB インスタンスの クラスターエンジンのバージョンまたはインスタンスクラスは変更されません。

クラスターの可用性を最大化するため、最初にクラスターのリーダーインスタンスを更新し、次にライターインスタンスを更新することをお勧めします。フェイルオーバーが発生するとダウンタイムが長引く可能性があるため、リーダーインスタンスとライターインスタンスを同時に更新することはお勧めしません。

Amazon DocumentDB のほとんどのオペレーティングシステムの更新はオプションであり、特定の適用日付は設定されていません。ただし、これらの更新をしばらく適用せずに放置していると、インスタンスのメンテナンスウィンドウ時に最終的に必要になり、想定外のタイミングで自動的に適用される場合があります。これは、データベースのセキュリティ体制を維持するために行われます。予期しないダウンタイムを避けるため、Amazon DocumentDB インスタンス向けオペレーティングシステム更新が利用可能になったらすぐに適用するように努め、ビジネスニーズに応じて都合の良いときにインスタンスメンテナンスウィンドウを設定することをお勧めします。

新しいオプションの更新が利用可能になったときに通知を受けるには、セキュリティパッチイベントカテゴリの RDS-EVENT-0230 をサブスクライブできます。Amazon DocumentDB イベントのサブスクライブについては、「Amazon DocumentDB イベントサブスクリプションをサブスクライブする」を参照してください。

インスタンスがプライマリインスタンスの場合、クラスターまたはインスタンスでメンテナンスが実行されると、フェイルオーバーが発生することを想定してください。可用性を向上させるためには、Amazon DocumentDB クラスターに複数のインスタンスを使用することをお勧めします。詳細については、「Amazon DocumentDB フェイルオーバー」を参照してください。

> **Note**
>
> 特定の管理機能では、Amazon DocumentDB は Amazon Relational Database Service (Amazon RDS) と共有の運用テクノロジーを使用します。

> **Important**
>
> Amazon DocumentDB インスタンスは、オペレーティングシステムのアップグレード中はオフラインになります。マルチインスタンスクラスターを使用することで、クラスターのダウンタイムを最小限に抑えることができます。マルチインスタンスクラスターがない場合は、セカンダリインスタンスの追加によりマルチインスタンスクラスターを一時的に構築し、メ

ンテナンスが完了したら追加した読み取りインスタンス (複数可) を削除することができます (セカンダリインスタンス向けの通常の料金が適用されます)。

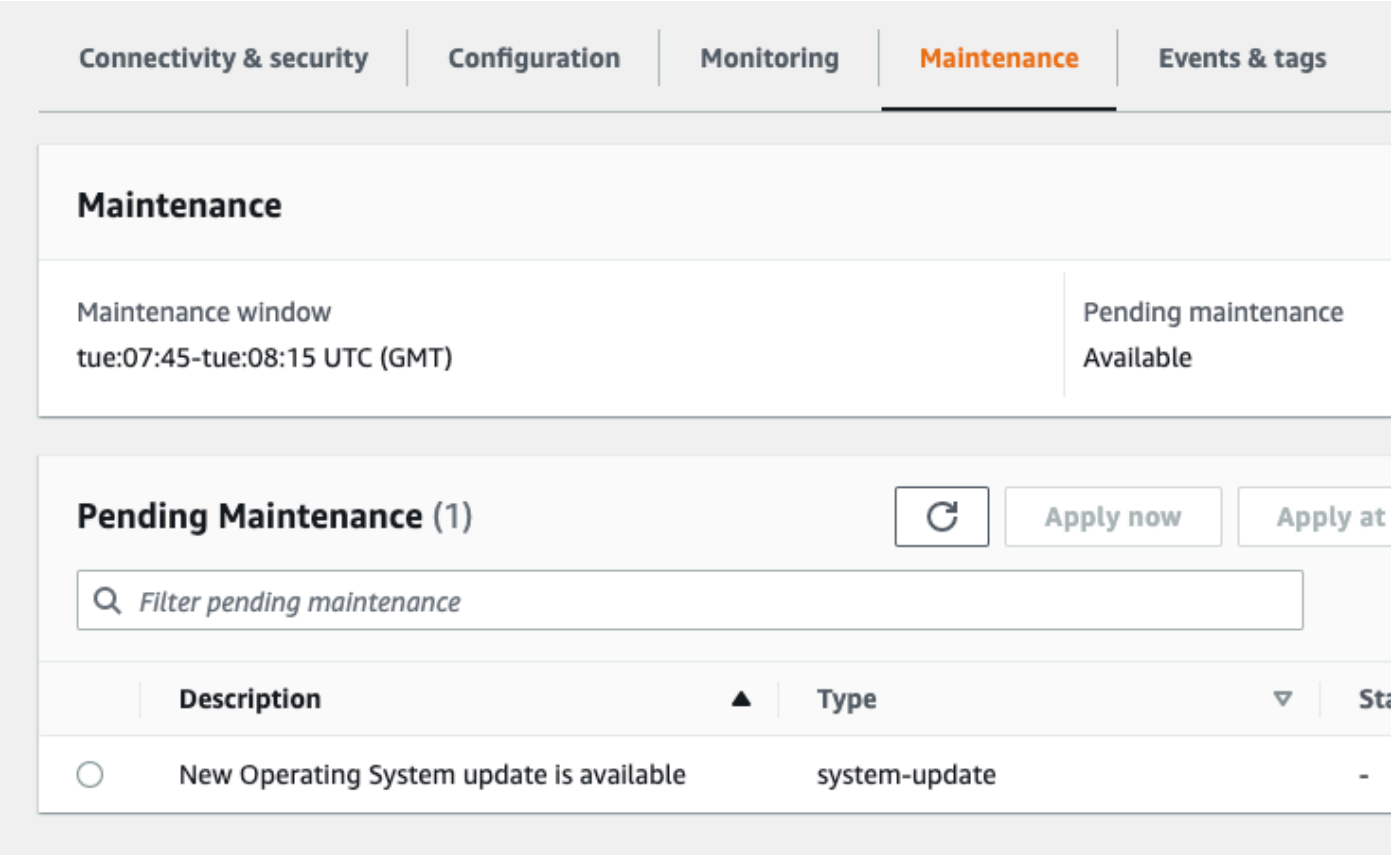
> **Note**
>
> さまざまなコンプライアンス義務を果たすためには、すべてのオプションおよび必須の更新を最新の状態に保つことが必要になる場合があります。Amazon DocumentDB によって提供されるすべての更新は、メンテナンス期間中に定期的に適用することをお勧めします。

AWS Management Console または を使用して AWS CLI 、更新が利用可能かどうかを判断できます。

Using the AWS Management Console

AWS Management Consoleを使用して、利用可能な更新があるかどうかを確認するには:

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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] を選択してから、インスタンスを選択します。
3. [メンテナンス] を選択します。
4. [保留中のメンテナンス] セクションで、オペレーティングシステムの更新を検索します。



オペレーティングシステムの更新を選択し、[保留中のメンテナンス] セクションの [今すぐ適用] または [次のメンテナンスウィンドウで適用] をクリックすることができます。メンテナンス値が [次のウィンドウ] である場合は、[後でアップグレード] を選択してメンテナンス項目を延期します。既にスタートしているメンテナンスアクションは延期できません。

または、ナビゲーションペインでクラスターをクリックし、アクションメニューから今すぐ適用または次のメンテナンスウィンドウで適用を選択して、クラスターのリストからインスタンスを選択することもできます。

Using the AWS CLI

を使用して更新が使用可能かどうかを確認するには AWS CLI、 `describe-pending-maintenance-actions` コマンドを呼び出します。

```
aws docdb describe-pending-maintenance-actions
```

```
{
  "ResourceIdentifier": "arn:aws:docdb:us-east-1:123456789012:db:mydb2",
  "PendingMaintenanceActionDetails": [
    {
```

```
      "Action": "system-update",
      "Description": "New Operating System update is available"
    }
  ]
}
```

オペレーティングシステムの更新は、Amazon DocumentDB エンジンのバージョンとインスタンスクラスに固有です。したがって、Amazon DocumentDB インスタンスは、異なる時間に更新を受信または要求します。そのエンジンのバージョンとインスタンスクラスに基づいたインスタンスにオペレーティングシステムの更新がある場合は、更新がコンソールに表示されます。コマンドを実行する AWS CLI `describe-pending-maintenance-actions`か、`DescribePendingMaintenanceActions` API オペレーションを呼び出して表示することもできます。

なお、Amazon DocumentDB エンジンの最新のクラスターパッチリリースを実行していない場合、利用可能なメンテナンスとしてリストアップされているオペレーティングシステムの更新が表示されない場合があります。オペレーティングシステム更新を確実に把握および管理するには、まず Amazon DocumentDB エンジンを最新のエンジンパッチバージョンにアップグレードする必要があります。

サービスにリンクされたロールを理解する

Amazon DocumentDB (MongoDB 互換) は、 AWS Identity and Access Management (IAM) サービスにリンクされたロールを使用します。サービスにリンクされたロール は、Amazon DocumentDB に直接リンクされた特殊なタイプの IAM ロールです。サービスにリンクされたロールは Amazon DocumentDB によって事前定義されており、サービスがユーザーに代わって他の AWS サービスを呼び出すために必要なすべてのアクセス許可が含まれています。

サービスにリンクされたロールを使用すると、必要な権限を手動で追加する必要がないため、Amazon DocumentDB の使用が簡単になります。Amazon DocumentDB は、サービスにリンクされたロールの権限を定義します。特に定義されていない限り、Amazon DocumentDB のみがそのロールを引き受けることができます。定義されたアクセス許可には、信頼ポリシーとアクセス許可ポリシーが含まれ、そのアクセス許可ポリシーを他の IAM エンティティにアタッチすることはできません。

ロールを削除するには、まず関連リソースを削除します。これにより、リソースにアクセスするためのアクセス許可を誤って削除することができないため、Amazon DocumentDB リソースが保護されます。

サービスにリンクされたロールをサポートする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。

Amazon DocumentDB のサービスにリンクされたロールの許可

Amazon DocumentDB (MongoDB 互換) は、AWSServiceRoleForRDS という名前のサービスにリンクされたロールを使用して、Amazon DocumentDB がクラスターに代わって AWS サービスを呼び出すことを許可します。

サービスにリンクされたロール AWSServiceRoleForRDS では、以下のサービスを信頼してロールを引き受けます。

- `docdb.amazonaws.com`

ロールの許可ポリシーは、指定したリソースに対して以下のアクションを実行することを Amazon DocumentDB に許可します。

- ec2 でのアクション:
  - `AssignPrivateIpAddresses`
  - `AuthorizeSecurityGroupIngress`
  - `CreateNetworkInterface`
  - `CreateSecurityGroup`
  - `DeleteNetworkInterface`
  - `DeleteSecurityGroup`
  - `DescribeAvailabilityZones`
  - `DescribeInternetGateways`
  - `DescribeSecurityGroups`
  - `DescribeSubnets`
  - `DescribeVpcAttribute`
  - `DescribeVpcs`
  - `ModifyNetworkInterfaceAttribute`
  - `RevokeSecurityGroupIngress`
  - `UnassignPrivateIpAddresses`

- sns でのアクション:
  - ListTopic
  - Publish
- cloudwatch でのアクション:
  - PutMetricData
  - GetMetricData
  - CreateLogStream
  - PullLogEvents
  - DescribeLogStreams
  - CreateLogGroup

> **Note**
>
> サービスリンク役割の作成、編集、削除を IAM エンティティ (ユーザー、グループ、役割など) に許可するにはアクセス許可を設定する必要があります。次のエラーメッセージが返される場合があります。
> リソースを作成できません。サービスにリンクされたロールを作成するために必要なアクセス許可があることを確認します。それ以外の場合は、時間をおいてからもう一度お試しください。
> このエラーが表示された場合は、次のアクセス許可が有効であることを確認します。
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
> AWSServiceRoleForRDS",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"rds.amazonaws.com"
>         }
>     }
> }
> ```

詳細については、「IAM ユーザーガイド」の「サービスリンクロールの許可」を参照してください。

Amazon DocumentDB でのサービスにリンクされたロールの作成

サービスリンクロールを手動で作成する必要はありません。クラスターを作成すると、Amazon DocumentDB がサービスにリンクされたロールを作成します。

サービスにリンクされたこのロールを削除したが、再作成する必要がある場合は、同じプロセスで、アカウントにロールを再作成することができます。クラスターを作成すると、Amazon DocumentDB によってサービスにリンクされたロールが再度作成されます。

Amazon DocumentDB サービスにリンクされたロールの変更

Amazon DocumentDB は、AWSServiceRoleForAmazonMQ サービスリンクロールの編集を許可しません。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 を使用したロールの説明の変更はできます。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの編集」を参照してください。

Amazon DocumentDB で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、サービスにリンクされたロールを削除する前に、すべての クラスターを削除する必要があります。

Amazon DocumentDB 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、まずそのロールにアクティブなセッションがないことを確認し、そのロールで使用されているリソースをすべて削除する必要があります。

サービスにリンクされたロールがアクティブなセッションを持っているかどうかをコンソールを使用して確認するには

1. にサインイン AWS Management Console し、 で IAM コンソールを開きますhttps://console.aws.amazon.com/iam/。
2. IAM コンソールのナビゲーションペインで、ロール を選択してから、AWSServiceRoleForRDS ロールの名前 (チェックボックスではない) 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。

> **Note**
>
> Amazon DocumentDB が AWSServiceRoleForRDS ロールを使用しているかどうかわからない場合は、ロールの削除を試みることができます。サービスでロールが使用されている場合、削除は失敗し、ロールが使用されている リージョンが表示されます。ロールが使用されている場合は、ロールを削除する前にセッションが終了するのを待つ必要があります。サービスにリンクされたロールのセッションを取り消すことはできません。

AWSServiceRoleForRDS ロールを削除する場合、最初に すべて のインスタンスおよびクラスターを削除する必要があります。インスタンスとクラスターを削除する方法については、次のトピックを参照してください。

- Amazon DocumentDB インスタンスの削除
- Amazon DocumentDB クラスターの削除

Amazon DocumentDB サービスリンクロールがサポートされるリージョン

Amazon DocumentDB ではサービスを利用できるすべてのリージョンで、サービスにリンクされたロールの使用がサポートされます。詳細については、「https://docs.aws.amazon.com/documentdb/latest/developerguide/regions-and-azs.html#regions-and-azs-availability」を参照してください。

Amazon DocumentDB を用いた変更ストリームの使用

Amazon DocumentDB (MongoDB 互換) の変更ストリーム機能は、クラスターのコレクション内で発生する変更イベントの時系列シーケンスを提供します。変更ストリームからイベントを読み取って、次のようなさまざまなユースケースを実装できます。

- 変更通知
- Amazon OpenSearch Service (OpenSearch Service) を用いたフルテキスト検索
- Amazon Redshift を用いてのイベントの分析

アプリケーションは変更ストリームを使用して、個々のコレクションでデータ変更をサブスクライブすることができます。変更ストリームイベントは、クラスターで発生する際に順序付けされ、イベントが記録されてから 3 時間 (デフォルト) 保存されます。change_stream_log_retention_duration のパラメータを使って、保存期間を7日まで延長できます。変更ストリーム保持期間を変更するには、Modifying the Change Stream Log Retention Duration を参照してください。

トピック

- サポートされているオペレーション
- 請求
- 制限
- 変更ストリームの有効化
- 例: Python での変更ストリームの使用
- ドキュメント全体の参照
- 変更ストリームの再開
- startAtOperationTime を用いた変更ストリームの再開
- postBatchResumeToken を用いた変更ストリームの再開
- 変更ストリーム内のトランザクション
- 変更ストリームログの保持期間の変更
- セカンダリインスタンスでの変更ストリームの使用

サポートされているオペレーション

Amazon DocumentDB では、変更ストリームに対して次の操作がサポートされています。

- すべての変更イベントは、MongoDB の db.collection.watch()、db.watch()、および client.watch() のAPI 中でサポートされている全ての変更イベント。
- 更新のための完全ドキュメント参照。
- 集約ステージ:$match、$project、$redact、および $addFields そして $replaceRoot。
- 再開トークンからの変更ストリームの再開
- startAtOperation (Amazon DocumentDB v4.0+ に適用される) を使用して、タイムスタンプから変更ストリームを再開する

請求

Amazon DocumentDB 変更ストリーム機能はデフォルトで無効になっており、この機能が有効化されるまで追加料金は発生しません。クラスターで変更ストリームを使用すると、追加の読み取り、および書き込み IO とストレージのコストが発生します。`modifyChangeStreams` の API オペレーションを使用して、クラスターのためのこの機能を有効にできます。料金のさらなる詳細については、Amazon DocumentDB の料金 を参照してください。

制限

変更ストリームには、Amazon DocumentDB で以下の制限があります。

- Amazon DocumentDB 3.6. および Amazon DocumentDB 4.0 では、変更ストリームは Amazon DocumentDB クラスターのプライマリインスタンスへの接続からのみ開くことができます。レプリカインスタンス上の変更ストリームからの読み取りは、Amazon DocumentDB 3.6 および Amazon DocumentDB 4.0 ではサポートされていません。`watch()` API オペレーションを呼び出すときは、すべての読み取りがプライマリインスタンスに送信されるようにする `primary` の読み取り設定を指定する必要があります (「例」セクションを参照)。
- Amazon DocumentDB 5.0 では、グローバルクラスターを含むプライマリインスタンスとセカンダリインスタンスの両方から変更ストリームを開くことができます。セカンダリ読み取り設定を指定して、変更ストリームをセカンダリインスタンスにリダイレクトできます。その他のベストプラクティスと制限事項については、「セカンダリインスタンスでの変更ストリームの使用」を参照してください。
- コレクションの変更ストリームに書き込まれるイベントは、最大 7 日間 (デフォルトは 3 時間) 使用できます。変更ストリームのデータは、新しい変更が行われていない場合でも、ログの保持期間が終了すると削除されます。
- `updateMany` や `deleteMany` のようなコレクションで長時間実行されている書き込みオペレーションは、長時間実行されている書き込みオペレーションが完了するまで、変更ストリームイベントの書き込みを一時的に停止する可能性があります。
- Amazon DocumentDB は、MongoDB オペレーションログ (oplog) をサポートしていません。
- Amazon DocumentDB を用いて、特定のコレクションで変更ストリームを明示的に有効にする必要があります。
- 変更ストリームイベントの合計サイズ (要求された場合、変更データとドキュメント全体を含む) が 16 MB より大きい場合、変更ストリームでの読み取りエラーがクライアントで発生します。
- Ruby ドライバは、現在、Amazon DocumentDB v3.6 での `db.watch()` と `client.watch()` の使用時にはサポートされていません。

- フィールドの更新値が以前の値と同じ場合、Amazon DocumentDB と MongoDB では変更ストリームの updateDescription コマンドからの出力が異なります。
  - 提供されるフィールドが $set コマンド内で指定されており、かつターゲット値が既にソース値と等しい場合、Amazon DocumentDB は updateDescription 出力内でフィールドを返しません。
  - MongoDB は、指定された値が現在の値と等しい場合でも、出力内でフィールドを返します。

変更ストリームの有効化

Amazon DocumentDB 変更ストリームは、特定のデータベース内のすべてのコレクションに対して有効にすることも、選択したコレクションに対してのみ有効にすることもできます。以下は、mongo シェルを使用してさまざまなユースケースの変更ストリームを有効にする方法の例です。データベース名とコレクション名を指定する場合、空の文字列はワイルドカードとして扱われます。

```
//Enable change streams for the collection "foo" in database "bar"
db.adminCommand({modifyChangeStreams: 1,
    database: "bar",
    collection: "foo",
    enable: true});
```

```
//Disable change streams on collection "foo" in database "bar"
db.adminCommand({modifyChangeStreams: 1,
    database: "bar",
    collection: "foo",
    enable: false});
```

```
//Enable change streams for all collections in database "bar"
db.adminCommand({modifyChangeStreams: 1,
    database: "bar",
    collection: "",
    enable: true});
```

```
//Enable change streams for all collections in all databases in a cluster
db.adminCommand({modifyChangeStreams: 1,
    database: "",
    collection: "",
    enable: true});
```

次のいずれかに該当する場合、変更ストリームはコレクションに対して有効になります。

- データベースとコレクションの両方が明示的に有効になっている。
- コレクションを含むデータベースが有効になっている。
- すべてのデータベースが有効になっている。

親データベースでも変更ストリームが有効になっている場合、またはクラスター内のすべてのデータベースが有効になっている場合は、データベースからコレクションを削除しても、そのコレクションの変更ストリームは無効になりません。削除されたコレクションと同じ名前で新しいコレクションが作成された場合、そのコレクションに対して変更ストリームが有効になります。

$listChangeStreams 集約パイプラインステージを使用すると、クラスターで有効なすべての変更ストリームを一覧表示できます。Amazon DocumentDB でサポートされているすべての集約ステージは、パイプラインで追加の処理に使用できます。以前に有効にしたコレクションが無効になっている場合、そのコレクションは $listChangeStreams 出力に表示されません。

```
//List all databases and collections with change streams enabled
cursor = new DBCommandCursor(db,
    db.runCommand(
        {aggregate: 1,
        pipeline: [{$listChangeStreams: 1}],
        cursor:{}}));
```

```
//List of all databases and collections with change streams enabled
{ "database" : "test", "collection" : "foo" }
{ "database" : "bar", "collection" : "" }
{ "database" : "", "collection" : "" }
```

```
//Determine if the database “bar” or collection “bar.foo” have change streams enabled
cursor = new DBCommandCursor(db,
  db.runCommand(
      {aggregate: 1,
       pipeline: [{$listChangeStreams: 1},
                  {$match: {$or: [{database: "bar", collection: "foo"},
                                  {database: "bar", collection: ""},
                                  {database: "", collection: ""}]}}
                 ],
      cursor:{}}));
```

例: Python での変更ストリームの使用

コレクションレベルで、 Python を用いて Amazon DocumentDB 変更ストリームを使用する例を次に示します。

```
import os
import sys
from pymongo import MongoClient, ReadPreference

username = "DocumentDBusername"
password = <Insert your password>

clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem')

db = client['bar']

#While ‘Primary’ is the default read preference, here we give an example of
#how to specify the required read preference when reading the change streams
coll = db.get_collection('foo', read_preference=ReadPreference.PRIMARY)
#Create a stream object
stream = coll.watch()
#Write a new document to the collection to generate a change event
coll.insert_one({'x': 1})
#Read the next change event from the stream (if any)
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf94f600000002010000000200009025'},
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'insert'}
"""

#A subsequent attempt to read the next change event returns nothing, as there are no
 new changes
print(stream.try_next())

"""
```

```
Expected Output:
None
"""

#Generate a new change event by updating a document
result = coll.update_one({'x': 1}, {'$set': {'x': 2}})
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf99d40000000101000000100009025'},
'clusterTime': Timestamp(1571789268, 1),
'documentKey': {'_id': ObjectId('5daf9502ea258751778163d7')},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'update',
'updateDescription': {'removedFields': [], 'updatedFields': {'x': 2}}}
"""
```

データベースレベルで、 Python を用いて Amazon DocumentDB 変更ストリームを使用する例を次に示します。

```
import os
import sys
from pymongo import MongoClient

username = "DocumentDBusername"
password = <Insert your password>
clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem')

db = client['bar']
#Create a stream object
stream = db.watch()
coll = db.get_collection('foo')
#Write a new document to the collection foo to generate a change event
coll.insert_one({'x': 1})

#Read the next change event from the stream (if any)
print(stream.try_next())

"""
Expected Output:
```

```
{'_id': {'_data': '015daf94f600000002010000000200009025'},
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'insert'}
"""
#A subsequent attempt to read the next change event returns nothing, as there are no
 new changes
print(stream.try_next())

"""
Expected Output:
None
"""

coll = db.get_collection('foo1')

#Write a new document to another collection to generate a change event
coll.insert_one({'x': 1})
print(stream.try_next())

"""
Expected Output: Since the change stream cursor was the database level you can see
 change events from different collections in the same database
{'_id': {'_data': '015daf94f600000002010000000200009025'},
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo1', 'db': 'bar'},
'operationType': 'insert'}
"""
```

ドキュメント全体の参照

変更の更新イベントには、ドキュメント全体は含まれません。加えられた変更のみが含まれます。ユースケースで、更新の影響を受ける完全なドキュメントが必要な場合は、ストリームを開くときに完全なドキュメント参照を有効にできます。

変更の更新ストリームイベントの fullDocument ドキュメントは、ドキュメント参照時に更新されたドキュメントの最新バージョンを表します。更新オペレーションと fullDocument ルックアップ

の間で変更が発生した場合、fullDocument ドキュメントが更新時にドキュメントの状態を表していない可能性があります。

更新ルックアップを有効にしてストリームオブジェクトを作成するには、次の例を使用します。

```
stream = coll.watch(full_document='updateLookup')

#Generate a new change event by updating a document
result = coll.update_one({'x': 2}, {'$set': {'x': 3}})

stream.try_next()
```

ストリームオブジェクトの出力は次のようになります。

```
{'_id': {'_data': '015daf9b7c000000010100000100009025'},
'clusterTime': Timestamp(1571789692, 1),
'documentKey': {'_id': ObjectId('5daf9502ea258751778163d7')},
'fullDocument': {'_id': ObjectId('5daf9502ea258751778163d7'), 'x': 3},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'update',
'updateDescription': {'removedFields': [], 'updatedFields': {'x': 3}}}
```

変更ストリームの再開

再開トークンを使用すると、変更ストリームを後で再開できます。このトークンは、最後に取得した変更イベントドキュメントの _id フィールドと同じです。

```
import os
import sys
from pymongo import MongoClient

username = "DocumentDBusername"
password = <Insert your password>
clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem', retryWrites='false')

db = client['bar']
coll = db.get_collection('foo')
#Create a stream object
stream = db.watch()
coll.update_one({'x': 1}, {'$set': {'x': 4}})
```

```
event = stream.try_next()
token = event['_id']
print(token)

"""
Output: This is the resume token that we will later us to resume the change stream
{'_data': '015daf9c5b0000000101000000100009025'}
"""
#Python provides a nice shortcut for getting a stream's resume token
print(stream.resume_token)

"""
Output
{'_data': '015daf9c5b0000000101000000100009025'}
"""
#Generate a new change event by updating a document
result = coll.update_one({'x': 4}, {'$set': {'x': 5}})
#Generate another change event by inserting a document
result = coll.insert_one({'y': 5})
#Open a stream starting after the selected resume token
stream = db.watch(full_document='updateLookup', resume_after=token)
#Our first change event is the update with the specified _id
print(stream.try_next())

"""
#Output: Since we are resuming the change stream from the resume token, we will see all
 events after the first update operation. In our case, the change stream will resume
 from the update operation {x:5}

{'_id': {'_data': '015f7e8f0c0000000601000000006000fe038'},
'operationType': 'update',
'clusterTime': Timestamp(1602129676, 6),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e8f0ac423bafbfd9adba2')},
'fullDocument': {'_id': ObjectId('5f7e8f0ac423bafbfd9adba2'), 'x': 5},
'updateDescription': {'updatedFields': {'x': 5}, 'removedFields': []}}
"""
#Followed by the insert
print(stream.try_next())

"""
#Output:
{'_id': {'_data': '015f7e8f0c0000000701000000007000fe038'},
'operationType': 'insert',
```

```
'clusterTime': Timestamp(1602129676, 7),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e8f0cbf8c233ed577eb94')},
'fullDocument': {'_id': ObjectId('5f7e8f0cbf8c233ed577eb94'), 'y': 5}}
"""
```

startAtOperationTime を用いた変更ストリームの再開

startAtOperationTime を使って、特定のタイムスタンプから後で変更ストリームを再開できます。

> **Note**
>
> startAtOperationTime を使用する能力は、Amazon DocumentDB 4.0+ で入手可能です。startAtOperationTime を使用する場合、変更ストリームカーソルは、指定された Timestamp 以降で発生した変更のみを返します。startAtOperationTime と resumeAfter のコマンドは相互に排他的であるため、一緒に使用することはできません。

```
import os
import sys
from pymongo import MongoClient

username = "DocumentDBusername"
password = <Insert your password>
clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='rds-root-ca-2020.pem',retryWrites='false')
db = client['bar']
coll = db.get_collection('foo')
#Create a stream object
stream = db.watch()
coll.update_one({'x': 1}, {'$set': {'x': 4}})
event = stream.try_next()
timestamp = event['clusterTime']
print(timestamp)
"""
Output
Timestamp(1602129114, 4)
"""
#Generate a new change event by updating a document
```

```
result = coll.update_one({'x': 4}, {'$set': {'x': 5}})
result = coll.insert_one({'y': 5})
#Generate another change event by inserting a document
#Open a stream starting after specified time stamp

stream = db.watch(start_at_operation_time=timestamp)
print(stream.try_next())

"""
#Output: Since we are resuming the change stream at the time stamp of our first update
 operation (x:4), the change stream cursor will point to that event
{'_id': {'_data': '015f7e941a0000000301000000003000fe038'},
'operationType': 'update',
'clusterTime': Timestamp(1602130970, 3),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e9417c423bafbfd9adbb1')},
'updateDescription': {'updatedFields': {'x': 4}, 'removedFields': []}}
"""

print(stream.try_next())
"""
#Output: The second event will be the subsequent update operation (x:5)
{'_id': {'_data': '015f7e9502000000050100000005000fe038'},
'operationType': 'update',
'clusterTime': Timestamp(1602131202, 5),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e94ffc423bafbfd9adbb2')},
'updateDescription': {'updatedFields': {'x': 5}, 'removedFields': []}}
"""

print(stream.try_next())

"""
#Output: And finally the last event will be the insert operation (y:5)
{'_id': {'_data': '015f7e9502000000060100000006000fe038'},
'operationType': 'insert',
'clusterTime': Timestamp(1602131202, 6),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e95025c4a569e0f6dde92')},
'fullDocument': {'_id': ObjectId('5f7e95025c4a569e0f6dde92'), 'y': 5}}
"""
```

postBatchResumeToken を用いた変更ストリームの再開

Amazon DocumentDB 変更ストリームは、 という追加のフィールドを返すようになりましたpostBatchResumeToken。このフィールドは、 $changestream コマンドと getMore コマンドから返されます。

Python の $changestream コマンドの例 :

```
db.command({"aggregate": "sales", "pipeline": [{ "$changeStream": {}}], "cursor":
 {"batchSize": 1}})
```

正常な出力:

```
cursor" : {
   "firstBatch" : [ ],
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
   "id" : NumberLong("9660788144470"),
   "ns" : "test.sales"
}
```

Python の getMore コマンドの例 :

```
db.command({"getMore": NumberLong(<cursor id>), "collection": "sales", "batchSize":
 1 })
```

正常な出力

```
cursor" : {
   "nextBatch" : [ ],
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
   "id" : NumberLong("9660788144470"),
   "ns" : "test.sales"
}
```

postBatchResumeToken フィールドは、再開トークンの使用方法と同様に、resumeAfterフィールドで新しい変更ストリームカーソルを開くために使用できます。

選択した の後に開始するストリームを開きますpostBatchResumeToken。

```
post_batch_resume_token = output['cursor']['postBatchResumeToken']
stream = db.watch(full_document='updateLookup', resume_after=post_batch_resume_token)
```

実際のイベントを反映するオペレーションログ (oplog) エントリに常に対応する通常の再開トークンとは異なり、 は、変更ストリームがサーバーでスキャンした oplog エントリpostBatchResumeTokenに対応します。これは必ずしも一致する変更ではありません。

古い通常の再開トークンで再開しようとすると、データベースは指定されたタイムスタンプと現在の時刻の間のすべての oplog エントリを強制的にスキャンします。これにより、サブクエリスキャンごとに短時間、内部で多くのクエリが生成されます。これにより、CPU 使用率が急増し、データベースのパフォーマンスが低下します。最後の で再開すると、一致しない oplog エントリのスキャンがpostBatchResumeTokenスキップされます。

変更ストリーム内のトランザクション

変更ストリームイベントには、コミットされていないトランザクションや中止されたトランザクションのイベントは含まれません。たとえば、1 つのINSERTオペレーションと 1 つのUPDATEオペレーションでトランザクションを開始し、INSERTオペレーションが成功してもUPDATEオペレーションが失敗すると、トランザクションはロールバックされます。このトランザクションはロールバックされたため、変更ストリームにはこのトランザクションのイベントは含まれません。

変更ストリームログの保持期間の変更

AWS Management Console または を使用して、変更ストリームログの保持期間を 1 時間から 7 日の間で変更できます AWS CLI。

Using the AWS Management Console

変更ストリームログの保持期間を変更するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. [パラメータグループ] ペインで、クラスターに関連付けられたクラスターパラメータグループを選択します。クラスターに関連付けられているクラスターパラメータグループを調べるには、「Amazon DocumentDB クラスターのパラメータグループの確認をする」を参照してください。
4. 結果のページには、クラスターパラメータグループのパラメータと対応する詳細が表示されます。change_stream_log_retention_duration パラメータを選択します。
5. ページの右上にある [編集] を選択して、パラメータの値を変更します。change_stream_log_retention_duration のパラメータは、1 時間から 7 日の間で変更できます。
6. 変更を行い、[クラスターのパラメータを変更] を選択して変更を保存します。変更を破棄するには、[キャンセル] を選択します。

Using the AWS CLI

クラスターパラメータグループの change_stream_log_retention_duration パラメータを変更するには、以下のパラメータを指定して modify-db-cluster-parameter-group オペレーションを使用します。

- **--db-cluster-parameter-group-name** — 必須。変更するクラスターパラメータグループの名前。クラスターに関連付けられているクラスターパラメータグループを調べるには、「Amazon DocumentDB クラスターのパラメータグループの確認をする」を参照してください。
- **--parameters** — 必須。変更するパラメータ。各パラメータの入力には以下を含める必要があります。
  - **ParameterName** - 変更しているパラメータの名前。この場合は change_stream_log_retention_duration
  - **ParameterValue** - このパラメータの新しい値。
  - **ApplyMethod** - このパラメータの変更を適用する方法です。有効な値は、immediate および pending-reboot です。

> **Note**
>
> ApplyType が static であるパラメータでは、ApplyMethod が pending-reboot である必要があります。

1. パラメータ change_stream_log_retention_duration の値を変更するには、次のコマンドを実行し、parameter-value をパラメータの変更後の値に置き換えます。

   Linux、macOS、Unix の場合:

   ```
   aws docdb modify-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --parameters
    "ParameterName=change_stream_log_retention_duration,ParameterValue=<parameter-
   value>,ApplyMethod=immediate"
   ```

   Windows の場合:

   ```
   aws docdb modify-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
       --parameters
    "ParameterName=change_stream_log_retention_duration,ParameterValue=<parameter-
   value>,ApplyMethod=immediate"
   ```

   このオペレーションによる出力は、次のようになります (JSON 形式) 。

   ```
   {
       "DBClusterParameterGroupName": "sample-parameter-group"
   }
   ```

2. 少なくとも 5 分お待ち下さい。
3. sample-parameter-group のパラメータ値を一覧表示し、変更が行われたことを確認します。

   Linux、macOS、Unix の場合:

   ```
   aws docdb describe-db-cluster-parameters \
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   Windows の場合:

   ```
   aws docdb describe-db-cluster-parameters ^
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "Parameters": [
        {
            "ParameterName": "audit_logs",
            "ParameterValue": "disabled",
            "Description": "Enables auditing on cluster.",
            "Source": "system",
            "ApplyType": "dynamic",
            "DataType": "string",
            "AllowedValues": "enabled,disabled",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
        },
        {
            "ParameterName": "change_stream_log_retention_duration",
            "ParameterValue": "12345",
            "Description": "Duration of time in seconds that the change stream
 log is retained and can be consumed.",
            "Source": "user",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "3600-86400",
            "IsModifiable": true,
            "ApplyMethod": "immediate"
        }
    ]
}
```

> **Note**
>
> ストリームログの保存期間を変更しても、ログサイズが 51,200 MB より大きくなるまで (>)、設定した change_stream_log_retention_duration の値より古いログは削除されません。

セカンダリインスタンスでの変更ストリームの使用

セカンダリインスタンスで変更ストリームの使用を開始するには、変更ストリームカーソルを readPreference でセカンダリとして開きます。

変更ストリームカーソルを開いて、特定のコレクション、または特定のクラスターもしくはデータベース内のすべてのコレクションの変更イベントを監視できます。任意の Amazon DocumentDB インスタンス上で変更ストリームカーソルを開き、ライターインスタンスとリーダーインスタンスの両方から変更ストリームドキュメントを取得できます。変更ストリームトークン (resumeToken や startOperationTime など) は、ライターインスタンスとリーダーインスタンスで開かれたさまざまな変更ストリームカーソル間で共有できます。

例

```
import os
import sys
from pymongo import MongoClient, ReadPreference

username = "DocumentDBusername"
password = <Your password>

clusterendpoint = "DocumentDBClusterEndpoint"

client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem')

db = client['bar']

# Make sure to use SECONDARY to redirect cursor reads from secondary instances
coll = db.get_collection('foo', read_preference=ReadPreference.SECONDARY)

# Create a stream object on RO. The token needs to generated from PRIMARY.
stream = coll.watch(resumeAfter=token)

for event in stream:
   print(event)
```

セカンダリインスタンス上の変更ストリームに関するガイドラインと制限事項

- 変更ストリームイベントは、プライマリインスタンスからセカンダリインスタンスにレプリケートする必要があります。また、Amazon CloudWatch で DBInstanceReplicaLag メトリクスからのラグをモニタリングすることもできます。
- セカンダリインスタンスのタイムスタンプは、プライマリインスタンスと常に同期しているとは限りません。この場合、ずれを解消するためセカンダリインスタンスのタイムスタンプに遅延が発生する可能性があります。ベストプラクティスとして、startAtOperationTime または resumeToken を使用してセカンダリインスタンスでウォッチを起動することをお勧めします。

- ドキュメントサイズが大きく、かつfullDocumentLookup を実行していて、プライマリインスタンスへの同時書き込みワークロードが多い場合、プライマリインスタンスと比較してセカンダリインスタンスのスループットが低下する可能性があります。ベストプラクティスとして、セカンダリのバッファキャッシュヒット率をモニタリングし、バッファキャッシュヒット率が高いかどうか確認することをお勧めします。

変更ストリーム AWS Lambda での の使用

Amazon DocumentDB は と統合 AWS Lambda されているため、Lambda 関数を使用して変更ストリームのレコードを処理できます。Lambda イベントソースマッピングは、Lambda を直接呼び出さない Amazon DocumentDB イベントを処理する際の Lambda 関数の呼び出しに使用できるリソースです。Amazon DocumentDB 変更ストリームをイベントソースとして利用すると、データの変更に対応するイベント駆動型アプリケーションを構築できます。例えば、Lambda 関数を使用して、新しいドキュメントを処理したり、既存のドキュメントの更新を追跡したり、ドキュメントの削除を記録したりできます。

イベントソースマッピングの設定では、Amazon DocumentDB 変更ストリームから Lambda 関数にレコードを送信するようにできます。イベントは一度に 1 つずつ送信することも、バッチ処理して効率を向上させ、順番に処理することもできます。イベントソースマッピングのバッチ処理動作は、特定のウィンドウ時間 (0～300 秒) またはバッチレコード数 (最大 10000 レコード) に基づいて設定できます。複数のイベントソースマッピングを作成することで、複数の Lambda 関数で同じデータを処理したり、1 つの関数で複数のストリームの項目を個別に処理したりできます。

関数からエラーが返された場合、Lambda は成功するまでバッチの実行を再試行します。変更ストリームのイベントの有効期限が切れた場合は、イベントソースマッピングを無効にします。この場合は、新しいイベントソースマッピングを作成し、任意の開始位置に設定できます。Lambda イベントソースマッピングは、ポーラーに備わる分散性が理由で、少なくとも 1 回イベントを処理します。その結果、まれに Lambda 関数が重複するイベントを受け取ることがあります。 AWS Lambda 関数を操作するためのベストプラクティスに従い、べき等関数を構築して、重複するイベントに関連する問題を回避します。詳細については、「 AWS Lambda デベロッパーガイド」のAmazon DocumentDB AWS Lambda console での の使用」を参照してください。

パフォーマンスのベストプラクティスとして、Lambda 関数は存続期間を短くする必要があります。また、不必要な処理の遅延が発生するのを防ぐため、複雑な口ジックは実行しないでください。特に高速ストリームの場合は、同期的に長時間実行する Lambda よりも、非同期的な後処理ステップ関数ワークフローをトリガーすることをお勧めします。詳細については AWS Lambda、「 AWS Lambda デベロッパーガイド」を参照してください。

制限

Amazon DocumentDB および を使用する際に考慮すべき制限事項を次に示します AWS Lambda。

- AWS Lambda は現在、Amazon DocumentDB 4.0 および 5.0 でのみサポートされています。
- AWS Lambda は現在、エラスティッククラスターまたはグローバルクラスターではサポートされていません。
- AWS Lambda ペイロードサイズは 6MB を超えることはできません。Lambda バッチサイズの詳細については、AWS Lambda 開発者ガイドの「Lambda イベントソースマッピング」セクションの「バッチの動作」を参照してください。

Amazon DocumentDB のエラスティッククラスター

Amazon DocumentDB エラスティッククラスターは、1 秒あたり数百万回の読み取り/書き込み速度とペタバイトのストレージ容量でワークロードに対応できます。また、インスタンスを選択、管理、アップグレードする必要がなくなるため、開発者は Amazon DocumentDB を簡単に操作できるようになります。

Amazon DocumentDB エラスティッククラスター作成の目的は以下の通りです。

- 豊富なクエリ機能と MongoDB API との互換性を備え、事実上無制限のスケーリング機能を持つデータベースをお探しのお客様にソリューションを提供する。
- 接続制限を緩和し、パッチ適用によるダウンタイムを短縮する。
- 引き続き、JSON ワークロード向けのクラウドネイティブで、伸縮性のある最高水準のアーキテクチャへの投資してください。

トピック

- エラスティッククラスターのユースケース
- エラスティッククラスターの利点
- エラスティッククラスターのリージョンとバージョンの可用性
- 制限
- Amazon DocumentDB のエラスティッククラスター
- Amazon DocumentDB エラスティッククラスターの開始方法
- Amazon DocumentDB エラスティッククラスターのベストプラクティス
- Amazon DocumentDB エラスティッククラスターの管理
- Amazon DocumentDB エラスティッククラスターで保管中のデータ暗号化
- エラスティッククラスター上のサービスにリンクされたロール

エラスティッククラスターのユースケース

ドキュメントデータベースは、迅速で反復性のある開発のための柔軟なスキーマを必要とするワークロードに適しています。Amazon DocumentDB のユースケース例については、「ドキュメントデータベースのユースケース」を参照してください。

以下では、ドキュメントデータベースに大きな利点があるユースケース例をいくつか紹介します。

ユーザープロファイル

ドキュメントデータベースに柔軟性の高いスキーマがあるため、多種多様の属性およびデータ値を大量に持つドキュメントを保存できます。エラスティッククラスターは、さまざまなユーザーがさまざまな種類の情報を提供するオンラインプロファイルに対する実用的なソリューションです。アプリケーションが何億というユーザープロファイルをサポートしていると仮定してみましょう。エラスティッククラスターは、そのようなアプリケーションに対応できます。なぜなら、エラスティッククラスター自体をスケールアップおよびスケールアウトして、ユーザープロファイルへの何百万回もの書き込みと読み取りに対応できるためです。オフピークの時間帯には、スケールダウンしてコストを削減することもできます。

コンテンツの管理と履歴の記録

コンテンツを効果的に管理するために、さまざまなソースからコンテンツを収集して集計し、お客様に提供する必要があります。ドキュメントデータベースは、柔軟性の高いスキーマがあるため、あらゆる種類のデータの収集および保存に最適です。イメージやコメント、ビデオなど、ユーザーが生成したコンテンツをはじめとして、新しい種類のコンテンツを作成し、取り込むことができます。時間が経つにつれて、データベースにはより多くのストレージが必要になります。エラスティッククラスターでは、データをより多くのストレージボリュームに分散できるため、1 つのクラスターにペタバイト規模のデータを保存できます。

エラスティッククラスターの利点

AWS サービス統合

Amazon DocumentDB エラスティッククラスターは、Amazon DocumentDB と同じ方法で他の AWS サービスと統合されます。

- 移行 — AWS Database Migration Service (DMS) を使用して、MongoDB およびその他のリレーショナルデータベースから Amazon DocumentDB エラスティッククラスターに移行できます。
- モニタリング — Amazon CloudWatch を使用して、エラスティッククラスターの状態とパフォーマンスをモニタリングできます。
- セキュリティ — AWS Identity and Access Management (IAM) を使用して認証と認可を設定して、Elastic クラスターの作成や更新などの Elastic クラスターリソースを管理し、Amazon VPC を

使用して VPC のみの安全な接続を実現できます。エラスティッククラスターの IAM 認証ログインはサポートされていません。

- データ管理 — を使用して AWS Glue 、Amazon S3、Amazon Redshift、Amazon OpenSearch Service などの他の AWS サービスとの間でデータをインポートおよびエクスポートできます。

エラスティッククラスターのリージョンとバージョンの可用性

Elastic クラスターのリージョン可用性

次の表は、Amazon DocumentDB エラスティッククラスターが現在利用可能な AWS リージョンと、各リージョンのエンドポイントを示しています。

リージョン名	リージョン	アベイラビリティーゾーン
米国東部 (バージニア北部)	us-east-1	5
米国東部 (オハイオ)	us-east-2	3
米国西部 (オレゴン)	us-west-2	3
アジアパシフィック (香港)	ap-east-1	3
アジアパシフィック (ムンバイ)	ap-south-1	3
アジアパシフィック (ソウル)	ap-northeast-2	3
アジアパシフィック (シンガポール)	ap-southeast-1	3
アジアパシフィック (シドニー)	ap-southeast-2	3
アジアパシフィック (東京)	ap-northeast-1	3
カナダ (中部)	ca-central-1	3
南米 (サンパウロ)	sa-east-1	3

リージョン名	リージョン	アベイラビリティーゾーン
欧州 (フランクフルト)	eu-central-1	3
欧州 (アイルランド)	eu-west-1	3
欧州 (ロンドン)	eu-west-2	3
欧州 (ミラノ)	eu-south-1	3
欧州 (パリ)	eu-west-3	3

バージョンの可用性

エラスティッククラスターは MongoDB 5.0 互換のワイヤープロトコルをサポートします。Amazon DocumentDB 4.0 インスタンスベースのクラスターと Elastic クラスターの違いについては、「」を参照してくださいAmazon DocumentDB 4.0 と Elastic クラスター の機能上の違い。

制限

エラスティッククラスターの管理

このリリースでは、以下のクラスター管理機能と機能はサポートされていません。

- カスタムポート (27017 のみがサポートされています)
- IAM ユーザーとロールを使用してデータベースを認証する機能
- グローバルクラスターの作成機能
- 既存の Amazon DocumentDB イベントとイベントのサブスクリプション
- レンジシャーディング
- シャードの既存のコレクション
- マルチフィールドシャードキー
- シャードキーを変更する
- ポイントインタイム復元
- クローン作成

- Performance Insights

> **Note**
>
> エラスティッククラスターの制限については、「Amazon DocumentDB のクォータと制限事項」を参照してください。

クエリ操作と書き込み操作

このリリースでは、次のクエリおよび書き込みオペレーションのコマンドと機能はサポートされていません。

- スケーリング中の DDL コマンド
- プロファイラー
- パラメータグループ
- AWS Config
- AWS Backup

コレクションとインデックスの管理

このリリースでは、以下のコレクションおよびインデックス管理機能はサポートされていません。

- 一意のインデックス
- 部分インデックス
- テキストインデックス
- ベクトルインデックス
- ドキュメント圧縮

管理と診断

このリリースでは、以下の管理および診断コマンドと機能はサポートされていません。

- AWS Secrets Manager
- ロールベースのアクセスコントロール (RBAC) におけるカスタムロール

- 接続時、 0 の Write Concern はサポートされていません。
- VPC に属していて、既存のエラスティッククラスターに割り当てられていないサブネットの変更。

オプトイン機能

このリリースでは、以下の Amazon DocumentDB オプトイン機能はサポートされていません。

- ACID トランザクション
- DDL/DML 監査
- Change streams
- セッション系コマンド

Amazon DocumentDB のエラスティッククラスター

このセクションのトピックでは、Amazon DocumentDB エラスティッククラスターを支えているメカニズムと機能について説明します。

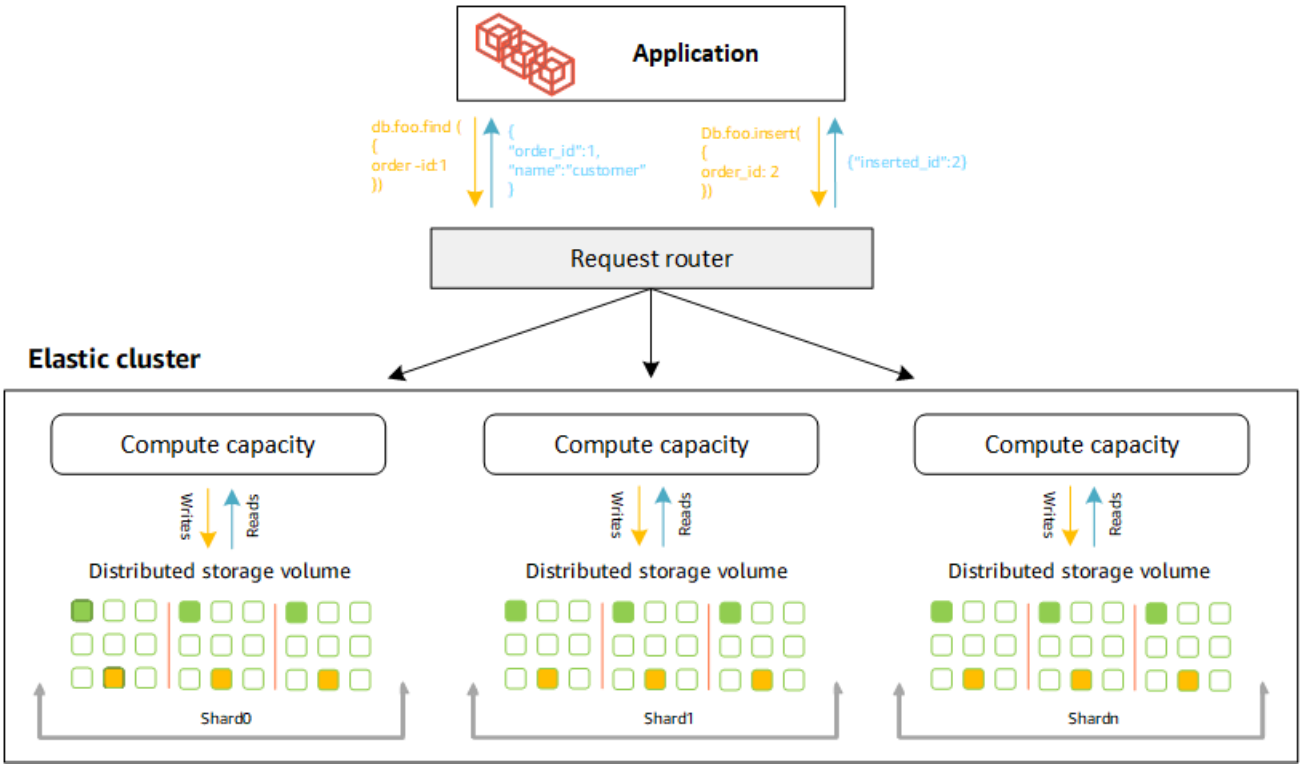
トピック

- Amazon DocumentDB エラスティッククラスターのシャーディング
- エラスティッククラスターの移行
- エラスティッククラスターのスケーリング
- エラスティッククラスターの信頼性
- エラスティッククラスターのストレージと可用性
- Amazon DocumentDB 4.0 と Elastic クラスター の機能上の違い

Amazon DocumentDB エラスティッククラスターのシャーディング

Amazon DocumentDB エラスティッククラスターは、ハッシュベースのシャーディングを使用して分散ストレージシステム間でデータを分割します。シャーディングはパーティショニングとも呼ばれ、大きなデータセットを複数のノードにまたがる小さなデータセットに分割し、垂直スケーリングの限界を超えてデータベースをスケールアウトできるようにします。Elastic クラスターは、Amazon DocumentDB のコンピューティングとストレージの分離または「デカップリング」を使用するた

め、互いに独立してスケーリングできます。エラスティッククラスターは、コンピューティングノード間で小さなデータを移動してコレクションを再分割するのではなく、分散型ストレージシステム内でデータを効率的にコピーします。



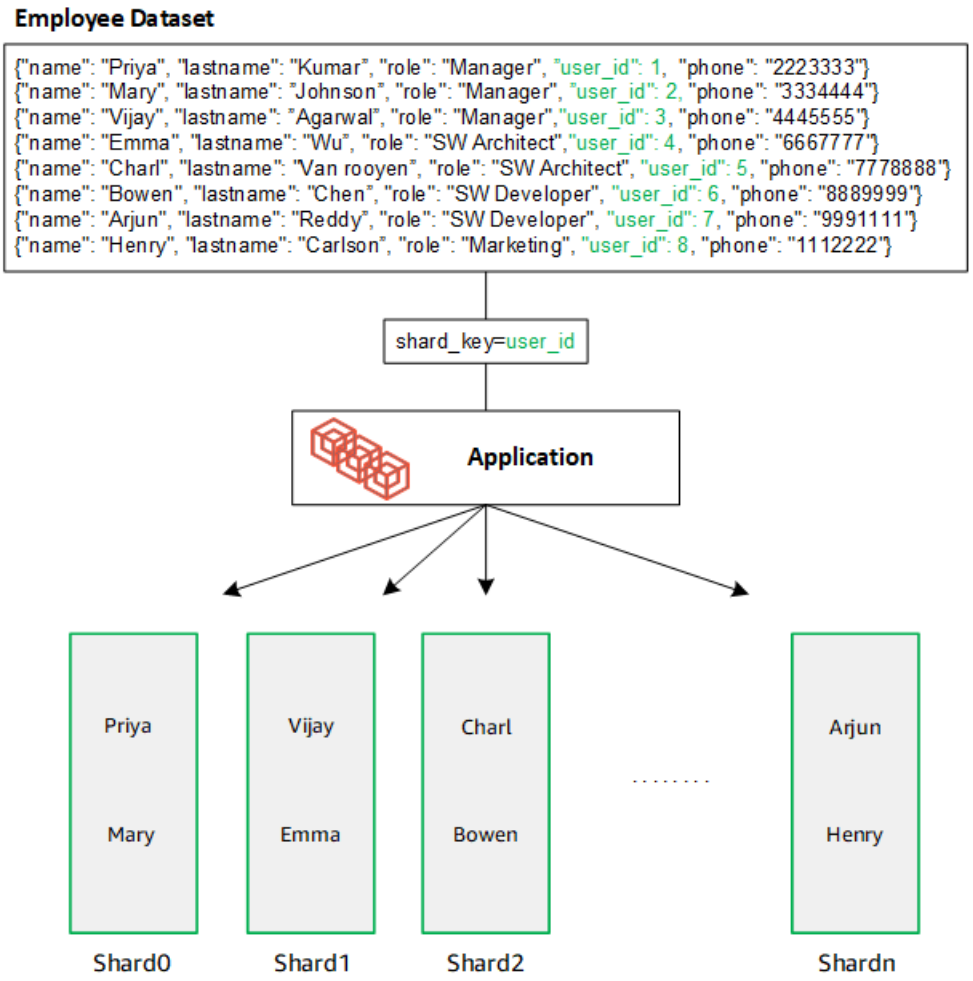
シャードの定義

以下はシャード命名法の定義です。

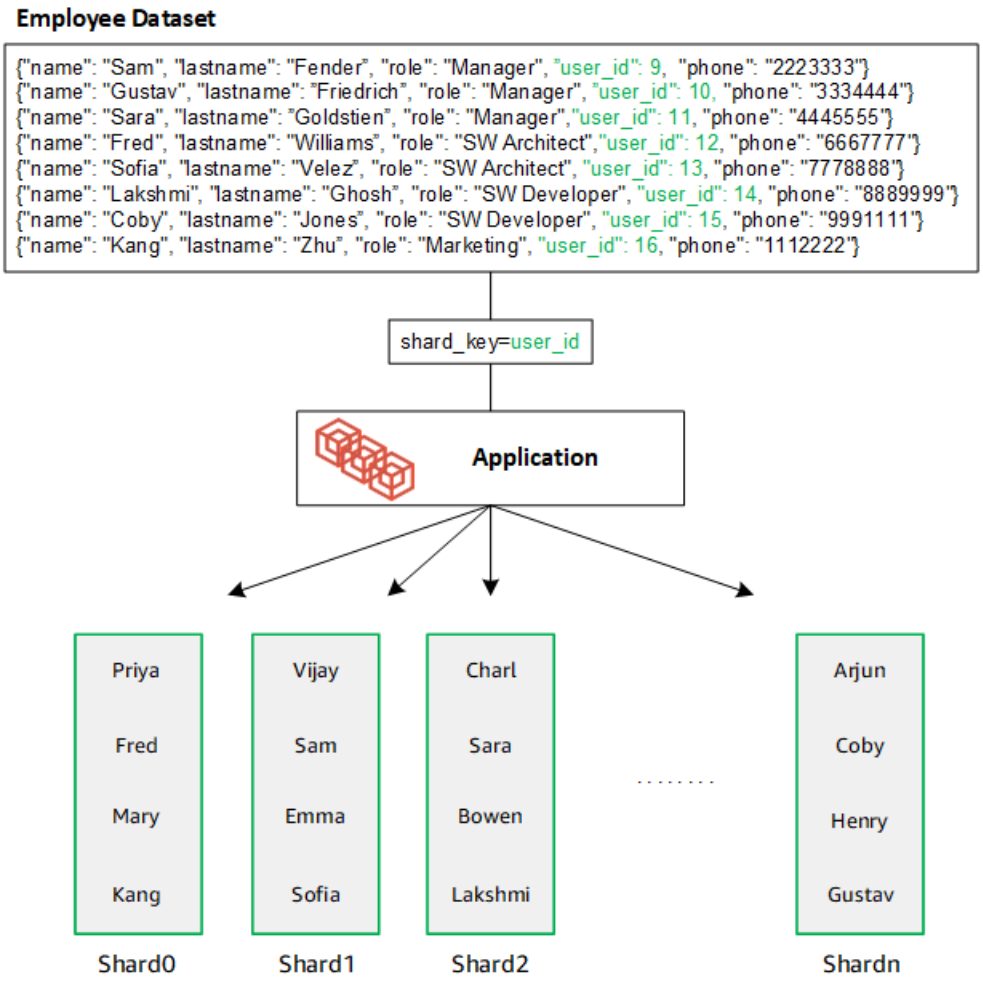
- シャード: - シャードは エラスティッククラスターのコンピューティング能力を提供します。1 つのライターインスタンスと 0 ~ 15 個のリードレプリカがあります。デフォルトでは、シャードにはライターと単一のリードレプリカの 2 つのインスタンスがあります。最大 32 個のシャードを設定し、各シャードインスタンスには最大 64 vCPUs を設定できます。
- シャードキー - シャードキーは、エラスティッククラスターが読み取りと書き込みトラフィックを分散するために使用する、シャードコレクション内の JSON ドキュメントの必須フィールドです。
- シャードコレクション — シャードコレクションは、データがデータパーティション内のエラスティッククラスターに分散されるコレクションです。
- パーティション: パーティションはシャードされたデータの論理的な部分です。シャードコレクションを作成すると、データはシャードキーに基づいて各シャード内のパーティションに自動的に編成されます。1 つのシャードには複数のパーティションがあります。

設定された複数のシャードに対するデータの分散

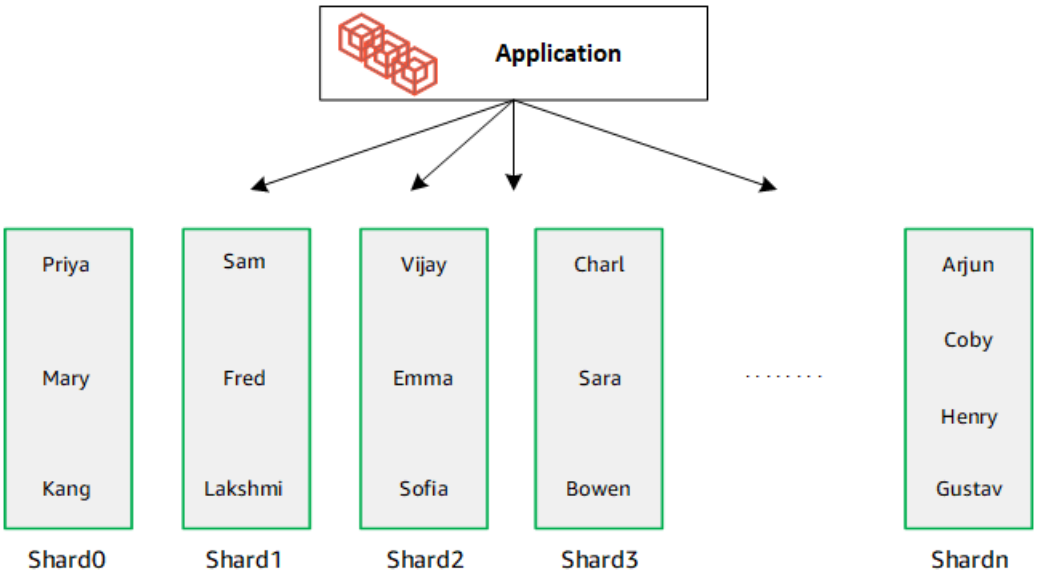
多数の一意の値を持つシャードキーを作成します。シャードキーが適切であると、その土台のシャード間でデータが均等にパーティショニングされ、ワークロードに最高のスループットとパフォーマンスを提供されます。次の例は、「user_id」という名前のシャードキーを使用する従業員名のデータです。



DocumentDB はハッシュシャーディングを使用して、土台のシャードの間でデータをパーティショニングします。追加データも同じ方法で挿入および分散されます。



シャードが追加されてデータベースがスケールアウトされると、Amazon DocumentDB はデータを自動的に再分散します。



エラスティッククラスターの移行

Amazon DocumentDB では、シャードされた MongoDB のデータをエラスティッククラスターに移行することができます。オフライン、オンライン、ハイブリッドの移行方法がサポートされています。詳細については、「Amazon DocumentDB への移行」を参照してください。

エラスティッククラスターのスケーリング

Amazon DocumentDB エラスティッククラスターは、エラスティッククラスター内のシャードの数 (スケールアウト) と、各シャードに適用される vCPUs の数 (スケールアップ) を増やす機能を提供します。必要に応じてシャードの数とコンピューティング能力 (vCPU) を減らすこともできます。

スケーリングのベストプラクティスについては、「エラスティッククラスターのスケーリング」を参照してください。

> **Note**
>
> クラスターレベルのスケーリングもできます。詳細については、「Amazon DocumentDB クラスターのスケーリング」を参照してください。

エラスティッククラスターの信頼性

Amazon DocumentDB は、信頼性、耐久性、耐障害性を実現するように設計されています。可用性を向上させるために、エラスティッククラスターは異なるアベイラビリティーゾーンに配置されたシャードごとに 2 つのノードをデプロイします。Amazon DocumentDB には、信頼性の高いデータベースソリューションとなるいくつかの自動機能も含まれています。詳細については、「Amazon DocumentDB の信頼性」を参照してください。

エラスティッククラスターのストレージと可用性

Amazon DocumentDB のデータは クラスターボリューム に保存されます。これは、SSD (Solid State Drive) を使用する単一の仮想ボリュームです。クラスターボリュームは、データの 6 つのコピーで構成され、1 つの AWS リージョン内の複数のアベイラビリティーゾーンに自動的にレプリケートされます。このレプリケーションにより、データの高い耐久性が保証され、データ損失の可能性が低くなります。また、データのコピーが他のアベイラビリティーゾーンにすでに存在するため、フェイルオーバー中のクラスターの可用性が高まります。ストレージ、高可用性、レプリケーションの詳細については、「」を参照してくださいAmazon DocumentDB の 仕組み。

Amazon DocumentDB 4.0 と Elastic クラスター の機能上の違い

Amazon DocumentDB 4.0 と エラスティッククラスターの間には、次のような機能上の違いがあります。

- top および collStats の結果はシャード別にパーティショニングされます。シャードコレクションの場合、データは複数のパーティションに分散され、collStats は、各パーティションから集約された collScans のレポートを作成します。
- シャードされたコレクションの top と collStats からコレクション統計は、クラスターのシャード数が変更されるとリセットされます。
- バックアップビルトインロールが serverStatus をサポートするようになりました。アクション - バックアップロールを持つデベロッパーおよびアプリケーションは、Amazon DocumentDB クラスターの状態に関する統計を収集できます。
- replSetGetConfig 出力では SecondaryDelaySecs フィールドが slaveDelay を置き換えます。
- hello コマンドは isMaster を置き換え、hello は エラスティッククラスターの役割を説明したドキュメント返します。
- エラスティッククラスターでは、 $elemMatch 演算子は配列の最初の入れ子レベルのドキュメントのみを一致検索します。Amazon DocumentDB 4.0 では、この演算子はすべてのレベルを横断して、一致したドキュメントを返します。例:

```
db.foo.insert(
[
    {a: {b: 5}},
    {a: {b: [5]}},
    {a: {b: [3, 7]}},
    {a: [{b: 5}]},
    {a: [{b: 3}, {b: 7}]},
    {a: [{b: [5]}]},
    {a: [{b: [3, 7]}]},
    {a: [[{b: 5}]]},
    {a: [[{b: 3}, {b: 7}]]},
    {a: [[{b: [5]}]]},
    {a: [[{b: [3, 7]}]]}
]);
// Elastic clusters
> db.foo.find({a: {$elemMatch: {b: {$elemMatch: {$lt: 6, $gt: 4}}}}}, {_id: 0})
```

```
{ "a" : [ { "b" : [ 5 ] } ] }

// Docdb 4.0: traverse more than one level deep
> db.foo.find({a: {$elemMatch: {b: {$elemMatch: {$lt: 6, $gt: 4}}}}}, {_id: 0})
{ "a" : [ { "b" : [ 5 ] } ] }
{ "a" : [ [ { "b" : [ 5 ] } ] ] }
```

- Amazon DocumentDB 4.0 では、「$」プロジェクションはすべてのドキュメントとすべてのフィールドを返します。エラスティッククラスターでは、「$」プロジェクションが指定された find コマンドは、「$」プロジェクションと一致するフィールドのみを含むクエリパラメータに一致するドキュメントを返します。
- エラスティッククラスターでは、 $regexおよび $optionsクエリパラメータを持つfindコマンドは、「$regex と $options の両方でオプションを設定できません」というエラーを返します。

- エラスティッククラスターの場合、$indexOfCP は以下のいずれかの場合に 「-1」を返します。
  - string expression でサブ文字列が見つからない。
  - 数値 start は end より大きい。
  - 数値 start は文字列のバイト長より大きい。

  Amazon DocumentDB 4.0 の場合、$indexOfCP は位置 start の数値が end または文字列のバイト長よりも大きい場合に「0」を返します。
- エラスティッククラスターでは、 _id fieldsなどの の射影オペレーション は{"_id.nestedField" : 1}、射影フィールドのみを含むドキュメントを返します。一方、Amazon DocumentDB 4.0 では、ネストされたフィールド射影コマンドはドキュメントを除外しません。

Amazon DocumentDB エラスティッククラスターの開始方法

このセクションでは、初めてのエラスティッククラスターを作成してクエリする方法を順を追って説明します。

Amazon DocumentDB に接続して開始する方法は複数あります。以下の手順は、ユーザーが当社の強力なドキュメントデータベースの使用を開始するための最も迅速、シンプルで簡単な方法です。このガイドではAWS CloudShell、 を使用して、 から直接 Amazon DocumentDB クラスターに接続してクエリを実行します AWS Management Console。 AWS 無料利用枠の対象となる新規のお客様は、Amazon DocumentDB と CloudShell を無料で使用できます。 AWS CloudShell 環境また

は Amazon DocumentDB エラスティッククラスターが無料利用枠を超えるリソースを使用する場合、それらのリソースの通常の AWS 料金が課金されます。このガイドでは、5 分以内に Amazon DocumentDB の使用を開始します。

トピック

- 前提条件
- ステップ 1: エラスティッククラスターを作成する
- ステップ 2: Elastic クラスターに接続する
- ステップ 3: コレクションをシャードし、データを挿入してクエリする
- ステップ 4: 探索する

前提条件

最初の Amazon DocumentDB クラスターを作成する前に、以下の操作を行う必要があります。

Amazon Web Services (AWS) アカウントを作成する

Amazon DocumentDB を使用する前に、Amazon Web Services (AWS) アカウントを持っている必要があります。 AWS アカウントは無料です。使用しているサービスとリソースに対してのみ料金をお支払いいただきます。

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

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

   サインアップ手順の一環として、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。

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

必要な AWS Identity and Access Management (IAM) アクセス許可を設定します。

クラスター、インスタンス、クラスターパラメータグループなどの Amazon DocumentDB リソースを管理するには、 がリクエストの認証 AWS に使用できる認証情報が必要です。詳細については、「Amazon DocumentDB の Identity and Access Management」を参照してください。

1. の検索バーで AWS Management Console、IAM と入力し、ドロップダウンメニューで IAM を選択します。
2. IAM コンソールにアクセスしたら、ナビゲーションペインから [ユーザー] を選択します。
3. ユーザーネームを選択します。
4. [Add permissions] (アクセス許可の追加) をクリックします。
5. [ポリシーを直接アタッチする] を選択します。
6. 検索バーに `AmazonDocDBElasticFullAccess` と入力し、検索結果に表示されたら、それを選択します。
7. [次へ] をクリックします。
8. [Add permissions] (アクセス許可の追加) をクリックします。

> **Note**
>
> AWS アカウントには、各リージョンにデフォルトの VPC が含まれます。Amazon VPC を使用する場合は、「Amazon VPC ユーザーガイド」の「Amazon VPC を作成する」トピックの手順を完了します。

ステップ 1: エラスティッククラスターを作成する

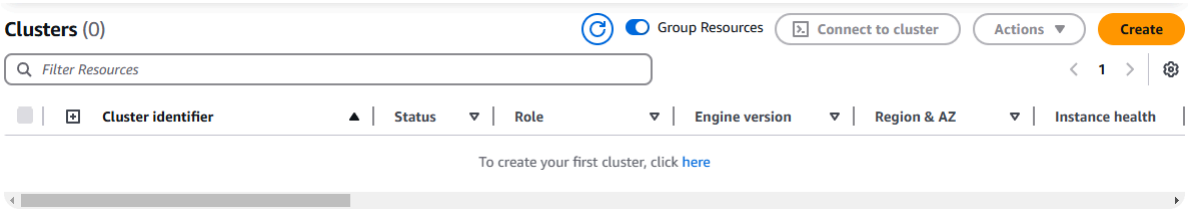
このセクションでは、 AWS Management Console または を使用して、以下の手順 AWS CLI で新しい Elastic クラスターを作成する方法について説明します。

Using the AWS Management Console

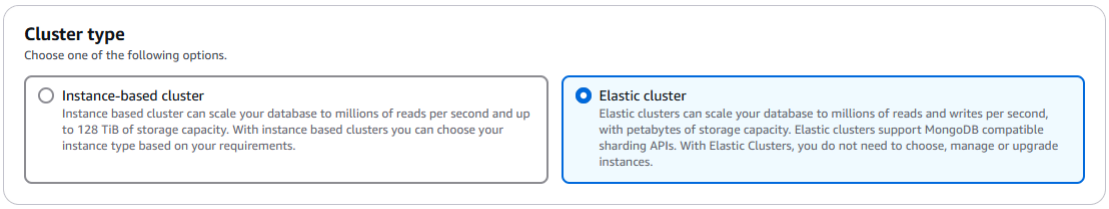
AWS Management Consoleを使用して エラスティッククラスター設定を作成するには、以下の手順に従います。

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

2. Amazon DocumentDB マネジメントコンソールの [クラスター] で [作成] を選択します。



3. Amazon DocumentDB クラスターの作成ページのクラスタータイプセクションで、Elastic クラスターを選択します。



4. 設定セクションで、以下を設定します。

   a. クラスター名フィールドに、一意のクラスター識別子を入力します (フィールドの下の命名要件に従います) 。

   b. [シャード数] フィールドに、クラスターに必要なシャードの数を入力します。クラスターあたりのシャードの最大数は 32 です。

> **Note**
>
> シャードごとに 2 つのノードが展開されます。両方のノードのシャード容量は同じになります。

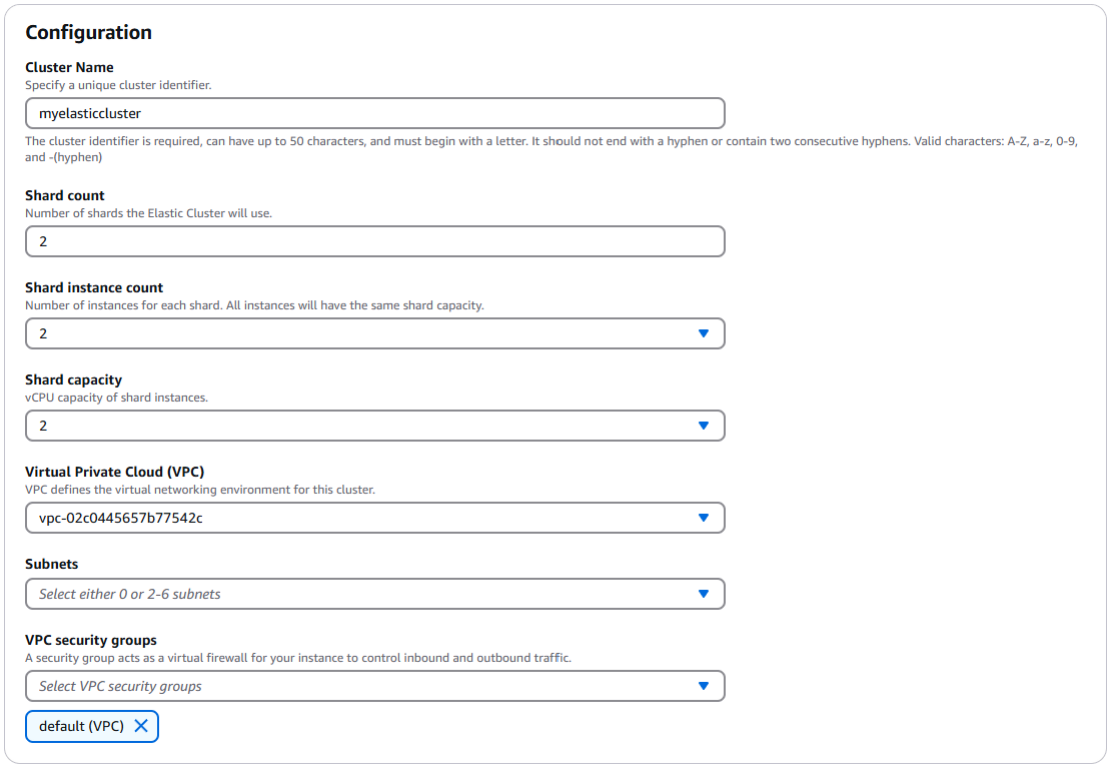
   c. [シャードインスタンス数] フィールドで、各シャードに関連付けるレプリカインスタンスの数を選択します。シャードインスタンスの最大数は 16 で、1 刻みで設定できます。すべてのレプリカインスタンスのシャード容量は、次のフィールドで定義するものと同じになります。テスト目的では、デフォルト値の 2 で十分です。

> **Note**
>
> レプリカインスタンスの数は、Elastic クラスター内のすべてのシャードに適用されます。シャードインスタンス数の値が 1 の場合、ライターインスタンスが 1 つあり、それ以外の追加のインスタンスはすべてレプリカで、読み取りや可用性の向上に使用できます。テスト目的では、デフォルト値の 2 で十分です。

d. [シャード容量] フィールドで、各シャードインスタンスに関連付ける仮想 CPU (vCPU) の数を選択します。1 シャードインスタンスあたりの vCPU の最大数は 64 です。使用できる値は、2、4、8、16、32、64 です。テスト目的では、デフォルト値の 2 で十分です。

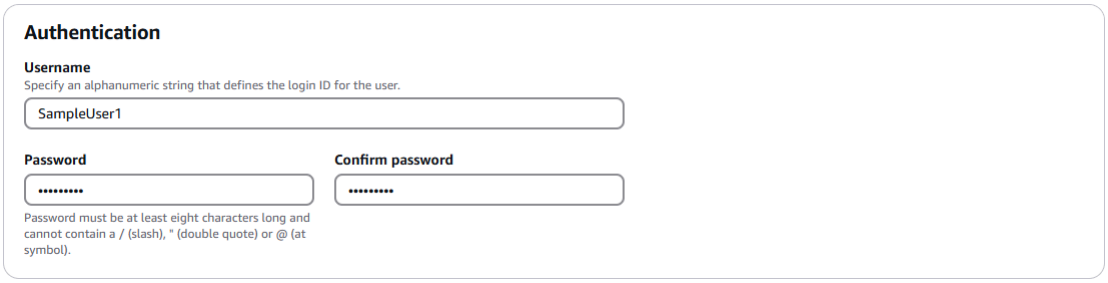
e. Virtual Private Cloud (VPC) フィールドで、ドロップダウンリストから VPC を選択します。

f. [サブネット] と [VPC セキュリティグループ] では、デフォルトを使用するか、選択した 3 つのサブネットと最大 3 つの VPC セキュリティグループ (最低 1 つ) を選択できます。



5. 認証セクションで、プライマリユーザーのログイン名を識別する文字列をユーザー名フィールドに入力します。

   パスワードフィールドに、指示に準拠した一意のパスワードを入力し、確認します。



6. 暗号化セクションで、デフォルト設定 (デフォルトキー) のままにします。

   必要に応じて、作成した AWS KMS key ARN を入力できます。詳細については、「Amazon DocumentDB エラスティッククラスターで保管中のデータ暗号化」を参照してください。

> **Important**
>
> エラスティッククラスターでは暗号化を有効にする必要があります。

7. [バックアップ] セクションで、該当するバックアップ要件に従ってフィールドを編集します。テスト目的で、デフォルト設定を保持できます。



   a. [バックアップ保持期間] - リストで、このクラスターの自動バックアップを削除するまで保持する日数を選択します。

   b. [バックアップウィンドウ] - Amazon DocumentDB によってこのクラスターのバックアップが作成される毎日の時刻と期間を設定します。

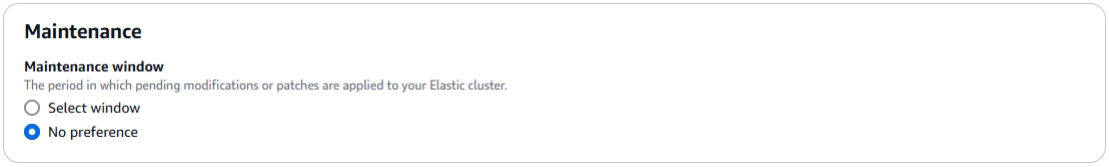
      i. バックアップの作成タイミングと期間を設定したい場合は、[ウィンドウを選択] を選択します。

         [開始時刻] - 最初のリストで、自動バックアップを開始する時間 (UTC 時) を選択します。2 番目のリストで、自動バックアップを開始する時間 (UTC 分) を選択します。

         [期間] - リストで、自動バックアップの作成に割り当てる時間を選択します。

ii. Amazon DocumentDB 動作に従ってバックアップの作成タイミングや期間を自動選択する場合は、[設定なし] を選択します。

8. [メンテナンス] セクションで、クラスターに変更やパッチを適用する日と時間、期間を選択します。テスト目的で、デフォルト設定を保持できます。



9. [クラスターを作成] を選択します。

エラスティッククラスターは現在プロビジョニング中です。このプロセスには数分かかることがあります。エラスティッククラスターのステータスがクラスターリストで利用可能と表示されると、クラスターに接続できます。

Using the AWS CLI

を使用して Elastic クラスターを作成するには AWS CLI、以下のパラメータを指定して `create-cluster`オペレーションを使用します。

- --cluster-name — 必須 作成時に入力した、または最終変更時のエラスティックスケールクラスターの現在の名前。
- --shard-capacity — 必須 各シャードに割り当てられた vCPU の数。最大数は 64 です。使用できる値は、2、4、8、16、32、64 です。
- --shard-count — 必須 クラスターに割り当てられたシャードの数。最大数は 32 です。
- --shard-instance-count - オプションです。このクラスター内のすべてのシャードに適用されるレプリカインスタンスの数。最大数は 16 です。
- --admin-user-name — 必須 管理者ユーザーに関連付けられているユーザー名。
- --admin-user-password — 必須 管理者ユーザーに関連付けられているパスワード。
- --auth-type — 必須 エラスティッククラスターへのアクセスに使用するパスワードの取得先の特定に使用している認証タイプ。有効なタイプは、`PLAIN_TEXT` または `SECRET_ARN` です。
- --vpc-security-group-ids - オプションです。このクラスターに関連付ける EC2 VPC セキュリティグループのリストを設定します。
- --preferred-maintenance-window - オプションです。週 1 回のシステムメンテナンスを実行できる時間帯 (世界標準時 (UTC)) を設定します。

  形式は `ddd:hh24:mi-ddd:hh24:mi` です。有効な曜日 (ddd): 月、火、水、木、金、土、日

デフォルトは、1 週間のうちのランダムな日に起こる、Amazon Web Services リージョンあたり 8 時間の範囲からランダムに選択された 30 分のウィンドウです。

最短 30 分のウィンドウ。

- --kms-key-id - オプションです。暗号化されたクラスターの KMS キー識別子を設定します。

  KMS キー識別子は、 AWS KMS 暗号化キーの Amazon リソースネーム (ARN) です。新しいクラスターの暗号化に使用する KMS 暗号化キーを所有する Amazon Web Services アカウントと同じアカウントを使用してクラスターを作成する場合、KMS 暗号化キーの ARN の代わりに KMS キーエイリアスを使用できます。

  KmsKeyId で暗号化キーが指定がなく、かつ `StorageEncrypted` パラメータが true の場合、Amazon DocumentDB はデフォルトの暗号化キーを使用します。
- --preferred-backup-window - オプションです。毎日自動バックアップを作成する希望の時間帯。デフォルトは、それぞれ 8 時間の時間ブロックからランダムに選択された 30 分のウィンドウです AWS リージョン。
- --backup-retention-period - オプション。自動バックアップを保管する日数。デフォルト値は 1 です。
- --storage-encrypted - オプションです。クラスターが暗号化されているかどうかを設定します。

  `--no-storage-encrypted` はクラスターが暗号化されていない指定です。
- --subnet-ids - オプションです。ネットワークのサブネット ID を設定します。

次の例では、各 *############* を独自の情報に置き換えます。

> **Note**
>
> 以下の例には、特定の KMS キーの作成例が含まれています。デフォルトの KMS キーを使用するには、`--kms-key-id` パラメータを含めないでください。

Linux、macOS、Unix の場合:

```
aws docdb-elastic create-cluster \
     --cluster-name sample-cluster-123 \
```

```
    --shard-capacity 8 \
    --shard-count 4 \
    --shard-instance-count 3 \
    --auth-type PLAIN_TEXT \
    --admin-user-name testadmin \
    --admin-user-password testPassword \
    --vpc-security-group-ids ec-65f40350 \
    --kms-key-id arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 \
    --subnet-ids subnet-9253c6a3, subnet-9f1b5af9 \
    --preferred-backup-window 18:00-18:30 \
    --backup-retention-period 7
```

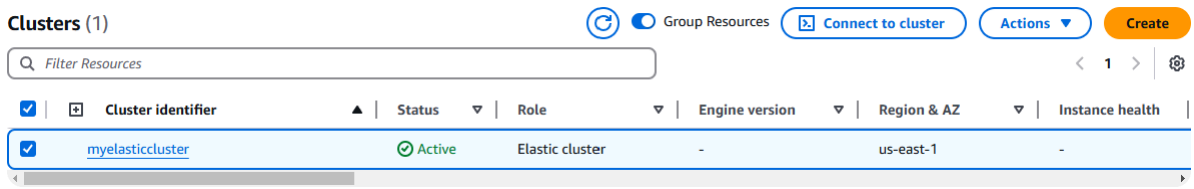
Windows の場合:

```
aws docdb-elastic create-cluster ^
    --cluster-name sample-cluster-123 ^
    --shard-capacity 8 ^
    --shard-count 4 ^
    --shard-instance-count 3 ^
    --auth-type PLAIN_TEXT ^
    --admin-user-name testadmin ^
    --admin-user-password testPassword ^
    --vpc-security-group-ids ec-65f40350 ^
    --kms-key-id arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 ^
    --subnet-ids subnet-9253c6a3, subnet-9f1b5af9 \
    --preferred-backup-window 18:00-18:30 \
    --backup-retention-period 7
```

ステップ 2: Elastic クラスターに接続する

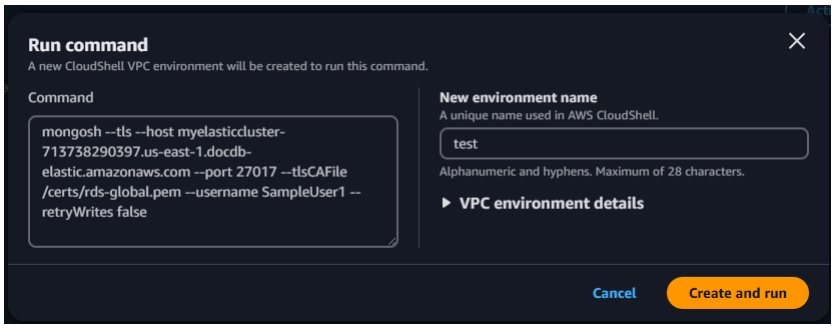
を使用して Amazon DocumentDB エラスティッククラスターに接続します AWS CloudShell。

1. Amazon DocumentDB マネジメントコンソールのクラスターで、作成した Elastic クラスターを見つけます。クラスターの横にあるチェックボックスをクリックして選択します。



2. クラスターに接続する (アクションドロップダウンメニューの横にある をクリックします。このボタンは、クラスターの横にあるチェックボックスをクリックし、クラスターのステータスが Available と表示された後にのみ有効になります。CloudShell の [Run コマンド] 画面が表示されます。

3. New environment name フィールドに「test」などの一意の名前を入力し、Create and run をクリックします。VPC 環境の詳細は、Amazon DocumentDB データベースに自動的に設定されます。



4. プロンプトが表示されたら、ステップ 1: Amazon DocumentDB エラスティッククラスターを作成する (サブステップ 5) で作成したパスワードを入力します。



パスワードを入力してプロンプトが になったらdirect: mongos] <env-name>>、Amazon DocumentDB クラスターに正常に接続されます。

> **Note**
>
> トラブルシューティングについては、「Amazon DocumentDB のトラブルシューティング」を参照してください。

ステップ 3: コレクションをシャードし、データを挿入してクエリする

エラスティッククラスターでは Amazon DocumentDB でのシャーディングのサポートが追加されています。クラスターに接続できたので、クラスターをシャーディングし、データを挿入して、クエリを実行できます。

1. コレクションをシャーディングするには、次のように入力します。

```
sh.shardCollection("db.Employee1" , { "Employeeid" : "hashed" })
```

2. 1 つのドキュメントを挿入するには、次のように入力します。

```
db.Employee1.insertOne({"Employeeid":1, "Name":"Joe", "LastName": "Bruin", "level":
 1 })
```

   以下のような出力が表示されます。

```
WriteResult({ "nInserted" : 1 })
```

3. findOne( ) コマンドで書き込んだドキュメントを読み取るには、 コマンドを入力します (コマンドから一つのドキュメントが返される)。

```
db.Employee1.findOne()
```

   以下のような出力が表示されます。

```
{
"_id" : ObjectId("61f344e0594fe1a1685a8151"),
"EmployeeID" : 1,
"Name" : "Joe",
"LastName" : "Bruin",
"level" : 1
}
```

4. さらにクエリを実行するには、ゲームプロファイルのユースケースを検討してみてください。最初に、「Employee」というタイトルのコレクションにエントリをいくつか挿入します。次のように入力します。

```
db.profiles.insertMany([ { "_id": 1, "name": "Matt", "status": "active", "level":
 12, "score": 202 },
```

```
     { "_id": 2, "name": "Frank", "status": "inactive", "level": 2, "score": 9 },
     { "_id": 3, "name": "Karen", "status": "active", "level": 7, "score": 87 },
     { "_id": 4, "name": "Katie", "status": "active", "level": 3, "score": 27 }
])
```

以下のような出力が表示されます。

```
{ acknowledged: true,
     insertedIds: {
        '0': ObjectId('679d02cd6b5a0581be78bcbd'),
        '1': ObjectId('679d02cd6b5a0581be78bcbe'),
        '2': ObjectId('679d02cd6b5a0581be78bcbf'),
        '3': ObjectId('679d02cd6b5a0581be78bcc0')
    }
}
```

5. プロファイルコレクション内のすべてのドキュメントを取得するには、find() コマンドを使用します。

```
db.Employee.find()
```

ステップ 4 で入力したデータが表示されます。

6. 1 つのドキュメントに対してクエリを発行するには、フィルター (「Katie」など) を含めます。次のように入力します。

```
db.Employee.find({name: "Katie"})
```

以下のような出力が表示されます。

```
[

   {

     _id: ObjectId('679d02cd6b5a0581be78bcc0'),

     Employeeid: 4,

     name: 'Katie',

     lastname: 'Schaper',
```

```
     level: 3

   }

]
```

7. プロファイルを見つけて変更するには、findAndModify コマンドを入力します。この例では、従業員「Matt」にはより高いレベル「14」が割り当てられています。

```
db.Employee.findAndModify({
    query: { "Employeeid" : 1, "name" : "Matt"},
    update: { "Employeeid" : 1, "name" : "Matt", "lastname" : "Winkle", "level" :
 14 }
})
```

以下のような出力が表示されます (レベルはまだ変更されていないことに注意してください)。

```
{

   _id: ObjectId('679d02cd6b5a0581be78bcbd'),

   Employeeid: 1,

   name: 'Matt',

   lastname: 'Winkle',

   level: 12

}
```

8. レベルが上がったことを確認するには、次のクエリを入力します。

```
db.Employee.find({name: "Matt"})
```

以下のような出力が表示されます。

```
[
   {

   _id: ObjectId('679d02cd6b5a0581be78bcbd'),
```

```
    Employeeid: 1,

    name: 'Matt',

    lastname: 'Winkle',

    level: 14

    }
]
```

ステップ 4: 探索する

お疲れ様でした。Amazon DocumentDB エラスティッククラスターの開始手順が正常に完了しました。

次のステップ このデータベースを一般的な機能の一部を使用して、完全に活用する方法を学びましょう。

- Amazon DocumentDB エラスティッククラスターのベストプラクティス
- Amazon DocumentDB エラスティッククラスターの管理

> **Note**
>
> この開始手順で作成した Elastic クラスターは、削除しない限り、引き続きコストが発生します。手順については、「Elastic クラスターの削除」を参照してください。

Amazon DocumentDB エラスティッククラスターのベストプラクティス

Amazon DocumentDB のエラスティッククラスターを使用するためのベストプラクティスについて説明します。インスタンスベースの Amazon DocumentDB クラスターのベストプラクティスはすべて、エラスティッククラスターに適用されます。新しいベストプラクティスが確認されると、このセクションは更新されます。

トピック

- シャードキーの選択
- 接続管理
- 非シャード型コレクション
- エラスティッククラスターのスケーリング
- エラスティッククラスターのモニタリング

シャードキーの選択

以下は、シャードキーの作成に関するガイドラインです。

- クラスター内のすべてのシャードにデータを分散するには、均等分散型のハッシュキーを使用する (ホットキーの使用は避ける)。
- スキャッターギャザークエリを回避するには、読み取り/更新/削除リクエストでシャードキーを使用する。
- 読み取り/更新/削除操作では、入れ子型のシャードキーの使用を避ける。
- バッチ操作では、すべてのシャードが並列実行され、レイテンシーが改善されるよう `ordered` を false に設定する。

接続管理

以下は、データベースへの接続管理に関するガイドラインです。

- 接続数、および、新しい接続の開閉の頻度をモニタリングする。
- アプリケーションの設定のすべてのサブネットに接続を分散する。複数のサブネットにクラスターを設定しているにもかかわらず、サブネットの一部しか使用されないと、最大接続数でボトルネックが発生する可能性があります。

非シャード型コレクション

以下は、非シャード型コレクションに関するガイドラインです。

- 非シャード型コレクションで負荷を分散するには、使用率の高い非シャード型コレクションを複数のデータベースに保存するようにする。Amazon DocumentDB のエラスティッククラスターは、データベースを複数のシャードに配置します。この時、同じデータベースの非シャード型コレクションは、同じシャードに配置されます。

エラスティッククラスターのスケーリング

以下は、エラスティッククラスターのスケーリングに関するガイドラインです。

- スケーリング操作によって、データベースエラーやネットワークエラーが短時間断続的に発生する場合があります。可能な限り、ピーク時間帯のスケーリングは避けてください。スケーリングは、メンテナンスウィンドウ中に行うようにしてください。
- コンピュートを増やす場合は、シャード数を増減するよりも、シャードの容量を増減 (各シャードの vCPU 数の変更) することをお勧めします。高速であり、データベースやネットワークの断続的なエラーが発生する時間が短くなります。
- 成長が見込まれる場合は、シャード容量をスケールするではなく、シャード数を増やすことをお勧めします。シャード容量を増やしてクラスターをスケールすることで、迅速なスケーリングが必要とされる状況に対応できます。
- クライアント側の再試行ポリシーをモニタリングして、エクスポネンシャルバックオフとジッターを試し、スケーリング中にエラーが発生したときにデータベースが過負荷にならないようにしてください。

エラスティッククラスターのモニタリング

以下は、エラスティッククラスターのモニタリングに関するガイドラインです。

- 各シャードのメトリクスのピーク対平均比を追跡して、トラフィックが不均等 (ホットキー/ホットスポットがある) になっていないか判断してください。ピーク対平均比を追跡するための主なメトリクスは次のとおりです。
  - `PrimaryInstanceCPUUtilization`
    - これはシャード単位にモニタリングできます。
    - クラスターレベルでは、平均から p99 までのスキューをモニタリングできます。
  - `PrimaryInstanceFreeableMemory`
    - これはシャード単位にモニタリングできます。
    - クラスターレベルでは、平均から p99 までのスキューをモニタリングできます。
  - `DatabaseCursorsMax`
    - スキューを判断するには、シャード単位でモニタリングする必要があります。
  - `Documents-Inserted/Updated/Returned/Deleted`
    - スキューを判断するには、シャード単位でモニタリングする必要があります。

Amazon DocumentDB エラスティッククラスターの管理

Amazon DocumentDB Elestic クラスターを管理するには、適切な Amazon DocumentDB コントロールプレーンアクセス許可を持つ IAM ポリシーが必要です。これらのアクセス許可により、クラスターとインスタンスを作成、変更、削除できます。Amazon DocumentDBFullAccess ポリシーは、Amazon DocumentDB Elastic クラスターを管理するために必要なすべてのアクセス許可を提供します。

以下のトピックでは、Amazon DocumentDB Elastic クラスターを操作するときのさまざまなタスクの実行方法を示します。

トピック

- Elastic クラスター設定の変更
- Elastic クラスターのモニタリング
- Elastic クラスターの削除
- Elastic クラスタースナップショットの管理
- Amazon DocumentDB Elastic クラスターの停止と開始
- Amazon DocumentDB エラスティッククラスターのメンテナンス

Elastic クラスター設定の変更

このセクションでは、 AWS Management Console または を使用して Elastic クラスターを変更する方法について説明します AWS CLI 。

クラスターを変更する主な用途は、シャード数やシャードの計算能力を増減してシャードをスケールすることです。

Using the AWS Management Console

を使用して Elastic クラスターの設定を変更するには AWS Management Console :

1. AWS Management Console にサインインし、Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択します。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ナビゲーションペインの左上隅にあるメニューアイコンを選択します。

3. [クラスター識別子] 列で、変更するクラスターの名前を選択します。
4. [Modify] (変更) を選択します。
5. 変更するフィールドを編集し、[クラスターの変更] を選択します。

Configuration

Cluster identifier

SampleCluster

Shard count

Number of shards the Elastic Cluster will use.

2

Shard instance count

Number of instances for each shard. All instances will have the same shard capacity.

2

Shard capacity

vCPU capacity of each shard.

2

Maintenance

Maintenance window

The period in which pending modifications or patches are applied to your Elastic cluster.

- ○ Select window
- ◉ No preference

Authentication

Username

SampleUser

New password

Confirm new password

Password must be at least eight characters long and cannot contain a / (slash), " (double quote) or @ (at symbol).

Network settings

Subnets

Select either 0 or 2-6 subnets

subnet-0b2962f92a0f5a8fb ✕ subnet-08c6d849efd4dfe96 ✕

VPC security groups

> **Note**
>
> または、[クラスター] ページに移動し、該当するクラスターの横にあるボックスにチェックを入れ、[アクション]、[変更] の順に選択して [クラスターの変更] ダイアログにアクセスします。

Using the AWS CLI

を使用して Elastic クラスターの設定を変更するには AWS CLI、以下のパラメータを指定して `update-cluster`オペレーションを使用します。

- **`--cluster-arn`** — 必須 変更するクラスターの ARN 識別子。
- **`--shard-capacity`** - オプションです。各シャードに割り当てられた vCPU の数。最大数は 64 です。使用できる値は、2、4、8、16、32、64 です。
- **`--shard-count`** - オプションです。クラスターに割り当てられたシャードの数。最大数は 32 です。
- **`--shard-instance`** 数 — オプション。このクラスター内のすべてのシャードに適用されるレプリカインスタンスの数。最大数は 16 です。
- **`--auth-type`** - オプションです。エラスティッククラスターへのアクセスに使用するパスワードの取得先の特定に使用している認証タイプ。有効なタイプは、`PLAIN_TEXT` または `SECRET_ARN` です。
- **`--admin-user-password`** - オプションです。 – 管理者ユーザーに関連付けられているパスワード。
- **`--vpc-security-group-ids`** - オプションです。このクラスターに関連付ける Amazon EC2 と Amazon 仮想プライベートクラウド (VPC) セキュリティグループのリストを設定します。
- **`--preferred-maintenance-window`** - オプションです。週 1 回のシステムメンテナンスを実行できる時間帯 (世界標準時 (UTC))を設定します。

  形式は `ddd:hh24:mi-ddd:hh24:mi` です。有効な曜日 (ddd): 月、火、水、木、金、土、日

  デフォルトは、1 週間のうちのランダムな日に起こる、Amazon Web Services リージョンあたり 8 時間の範囲からランダムに選択された 30 分のウィンドウです。

  最短 30 分のウィンドウ。
- **`--subnet-ids`** - オプションです。ネットワークのサブネット ID を設定します。

次の例では、各 ############## を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb-elastic update-cluster \
    --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 \
    --shard-capacity 8 \
    --shard-count 4 \
    --shard-instance-count 3 \
    --admin-user-password testPassword \
    --vpc-security-group-ids ec-65f40350 \
    --subnet-ids subnet-9253c6a3, subnet-9f1b5af9
```

Windows の場合:

```
aws docdb-elastic update-cluster ^
    --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 ^
    --shard-capacity 8 ^
    --shard-count 4 ^
    --shard-instance-count 3 ^
    --admin-user-password testPassword ^
    --vpc-security-group-ids ec-65f40350 ^
    --subnet-ids subnet-9253c6a3, subnet-9f1b5af9
```

変更後の Elastic クラスターのステータスをモニタリングするには、「Elastic クラスターのモニタリング」を参照してください。

Elastic クラスターのモニタリング

このセクションでは、 AWS Management Console または を使用して Elastic クラスターをモニタリングする方法を次の手順 AWS CLI で説明します。

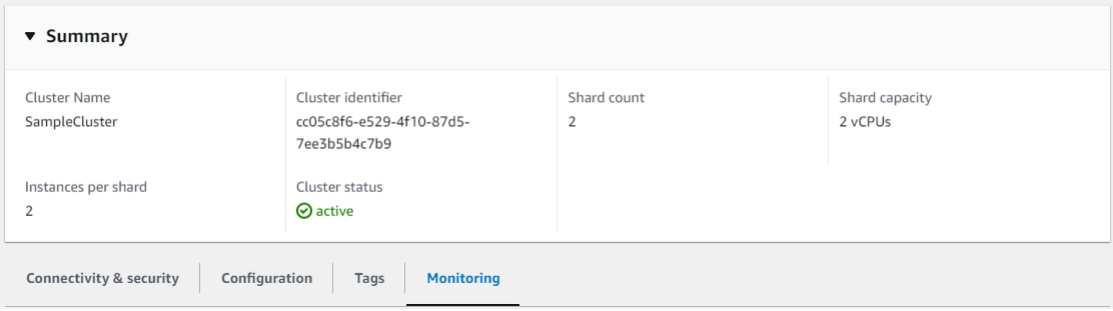
Using the AWS Management Console

を使用して Elastic クラスター設定をモニタリングするには AWS Management Console：

1. AWS Management Console にサインインし、Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択します。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ナビゲーションペインの左上隅にあるメニューアイコンを選択します。

3. [クラスター識別子] 列で、モニタリングするクラスターの名前を選択します。
4. モニタリングタブを選択します。



Amazon CloudWatch からの多くのチャートは、以下のモニタリングカテゴリについて表示されます。

- リソース使用率
- スループット
- オペレーション
- システム

を通じて Amazon CloudWatch にアクセスして AWS Management Console 、Elastic クラスター用に独自のモニタリング環境を設定することもできます。

Using the AWS CLI

を使用して特定の Elastic クラスター設定をモニタリングするには AWS CLI、以下のパラメータを指定して `get-cluster`オペレーションを使用します。

- **`--cluster-arn`** — 必須 情報が必要なクラスターの識別子。

次の例では、各 `##############` を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb-elastic get-cluster \
    --cluster-arn arn:aws:docdb-elastic:us-west-2:123456789012:cluster:/68ffcdf8-
e3af-40a3-91e4-24736f2dacc9
```

Windows の場合:

```
aws docdb-elastic get-cluster ^
    --cluster-arn arn:aws:docdb:-elastic:us-west-2:123456789012:cluster:/68ffcdf8-
e3af-40a3-91e4-24736f2dacc9
```

このオペレーションによる出力は、次のようになります。

```
"cluster": {
        ...
        "clusterArn": "arn:aws:docdb-elastic:us-
west-2:123456789012:cluster:/68ffcdf8-e3af-40a3-91e4-24736f2dacc9",
        "clusterEndpoint": "stretch-11-477568257630.us-east-1.docdb-
elastic.amazonaws.com",
        "readerEndpoint": "stretch-11-477568257630-ro.us-east-1.docdb-
elastic.amazonaws.com",
        "clusterName": "stretch-11",
        "shardCapacity": 2,
        "shardCount": 3,
        "shardInstanceCount: 5,
        "status": "ACTIVE",
        ...
 }
```

詳細については、「Amazon DocumentDB リソース管理 API リファレンス」の「DescribeClusterSnapshot」を参照してください。

を使用してすべての Elastic クラスターの詳細を表示するには AWS CLI、以下のパラメータを指定して list-clustersオペレーションを使用します。

- **--next-token** - オプションです。出力される項目数 (--max-results) が基本の API コールによって返される合計項目数より少ない場合、出力には NextToken が含まれ、これにより、後続のコマンドを渡して、次の項目のセットを取得できます。

- **--max-results** - オプションです。コマンドの出力で返される項目の総数。指定された max-results の値よりも多くの結果が存在する場合、ページ割りトークン (next-token) がレスポンスに含まれるため、残りの結果を取得できます。
  - デフォルト: 100
  - 最小は 20、最大は 100 です

次の例では、各 ############## を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb-elastic list-clusters \
    --next-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ== \
    --max-results 2
```

Windows の場合:

```
aws docdb-elastic list-clusters ^
    --next-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ== ^
    --max-results 2
```

このオペレーションによる出力は、次のようになります。

```
{
   "Clusters": [
      {
         "ClusterIdentifier":"mycluster-1",
         "ClusterArn":"arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster"
         "Status":"available",
         "ClusterEndpoint":"sample-cluster.sharded-cluster-corcjozrlsfc.us-
west-2.docdb.amazonaws.com"
       }
       {
         "ClusterIdentifier":"mycluster-2",
         "ClusterArn":"arn:aws:docdb:us-west-2:987654321098:sharded-cluster:sample-
cluster"
         "Status":"available",
         "ClusterEndpoint":"sample-cluster2.sharded-cluster-corcjozrlsfc.us-
west-2.docdb.amazonaws.com"
       }
```

```
    ]
}
```

Elastic クラスターの削除

このセクションでは、 AWS Management Console または を使用して Elastic クラスターを削除する方法を次の手順 AWS CLI で説明します。

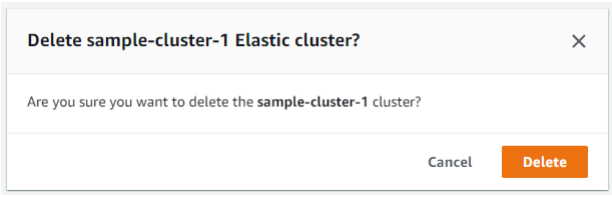
Using the AWS Management Console

AWS Management Consoleを使用して Elastic クラスター設定を削除するには、以下の手順に従います。

1. AWS Management Console にサインインし、Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択します。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ナビゲーションペインの左上隅にあるメニューアイコンを選択します。

3. クラスターリストの表で、削除するクラスター名の左にあるチェックボックスをオンにし、[アクション] を選択します。ドロップダウンメニューから [削除] を選択します。
4. [「cluster-name」の Elastic クラスターを削除しますか?] ダイアログボックスで [削除] を選択します。



クラスターが削除されるまでに数分かかります。クラスターのステータスをモニタリングするには、「Amazon DocumentDB クラスターのステータスのモニタリング」を参照してください。

Using the AWS CLI

を使用して Elastic クラスターを削除するには AWS CLI、次のパラメータを指定して `delete-cluster`オペレーションを使用します。

- **--cluster-arn** — 必須 削除するクラスターの ARN 識別子。
- **--no-skip-final-backup** - オプションです。最終バックアップを作成する場合は、最終バックアップの名前を指定して、このパラメータを含める必要があります。--final-backup-identifier または --skip-final-backup のどちらかを含める必要があります。
- **--skip-final-backup** - オプションです。クラスターを削除する前に最終バックアップを作成しない場合のみ、このパラメータを使用します。デフォルトでは、最終スナップショットを作成します。

次の AWS CLI コード例では、ARN が arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster のクラスターを最終バックアップで削除します。

次の例では、各 *############* を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb-elastic delete-cluster \
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster \
    --no-skip-final-backup \
    --final-backup-identifier finalArnBU-arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster
```

Windows の場合:

```
aws docdb-elastic delete-cluster ^
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster ^
    --no-skip-final-backup ^
    --final-backup-identifier finalArnBU-arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster
```

次の AWS CLI コード例では、ARN が arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster のクラスターを最終バックアップなしで削除します。

次の例では、各 *############* を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb-elastic delete-cluster \
```

```
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster \
    --skip-final-backup \
```

Windows の場合:

```
aws docdb-elastic delete-cluster ^
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster ^
    --skip-final-backup ^
```

`delete-cluster` オペレーションの出力は、削除するクラスターの表示になります。

クラスターが削除されるまでに数分かかります。クラスターのステータスをモニタリングするには、「Amazon DocumentDB クラスターのステータスのモニタリング」を参照してください。

Elastic クラススナップショットの管理

手動スナップショットは、Elastic クラスターの作成後に作成できます。自動バックアップは、Elastic クラススナップショットが作成されると直ちに作成されます。

> **Note**
>
> 手動スナップショットを作成するには、クラスターのスタータスが `Available` である必要があります。

このセクションでは、Elastic クラススナップショットを作成、表示、復元、削除する方法について説明します。

以下のトピックでは、Amazon DocumentDB Elastic クラススナップショットを操作するときのさまざまなタスクの実行方法を示します。

トピック

- 手動 Elastic クラススナップショットの作成
- Elastic クラススナップショットの表示
- スナップショットからの Elastic クラスターの復元
- Elastic クラススナップショットのコピー

- Elastic クラスタースナップショットの削除
- Elastic クラスタースナップショットの自動バックアップの管理

手動 Elastic クラスタースナップショットの作成

このセクションでは、 AWS Management Console または を使用して、次の手順 AWS CLI で手動 Elastic クラスタースナップショットを作成する方法について説明します。

Using the AWS Management Console

AWS Management Consoleを使用して手動 Elastic クラスタースナップショットを作成するには、以下の手順に従います。

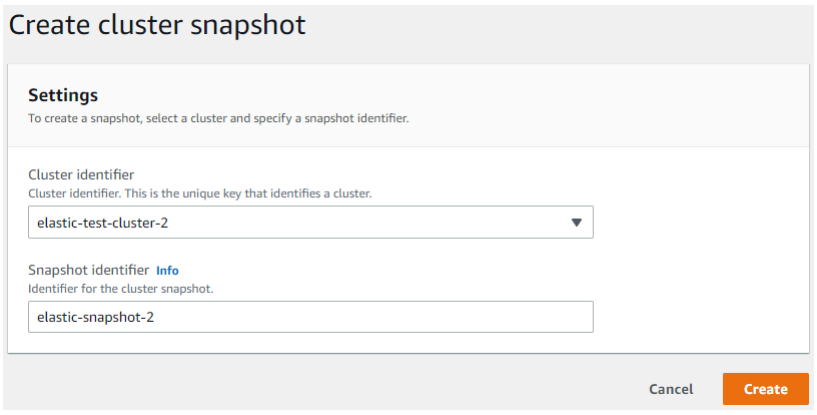
1. AWS Management Console にサインインし、Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[Snapshots] を選択してください。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ナビゲーションペインの左上隅にあるメニューアイコンを選択します。

3. [スナップショット] ページで、[作成] を選択します。
4. [クラスタースナップショットの作成] ページの [クラスター識別子] フィールドで、ドロップダウンリストからご使用の Elastic クラスターを選択します。

   [スナップショット識別子] フィールドに、Elastic クラスターの一意識別子を入力します。

   [作成] を選択します。



> **Note**
>
> または、[クラスター] ページに移動して [クラススナップショットの作成] ダイアログにアクセスし、対象クラスターの横にあるボックスにチェックを入れ、[アクション]、[スナップショットの取得] の順に選択します。

Elastic クラススナップショットがプロビジョニングされます。このプロセスには数分かかることがあります。[スナップショット] リストにステータスが `Available` と表示されたら、スナップショットを表示して復元できます。

Using the AWS CLI

を使用して手動 Elastic クラススナップショットを作成するには AWS CLI、以下のパラメータを指定して `create-cluster-snapshot`オペレーションを使用します。

- **`--snapshot-name`** — 必須 作成するクラスターの名前。
- **`--cluster-arn`** — 必須 スナップショットを作成するクラスターの ARN 識別子。

次の例では、各 *############* を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb-elastic create-cluster-snapshot \
    --snapshot-name sample-snapshot-1 \
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster
```

Windows の場合:

```
aws docdb-elastic create-cluster-snapshot ^
    --snapshot-name sample-snapshot-1 ^
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster
```

Elastic クラススナップショットの表示

このセクションでは、 AWS Management Console または を使用して Elastic クラススナップショット情報を表示する方法について説明します AWS CLI 。

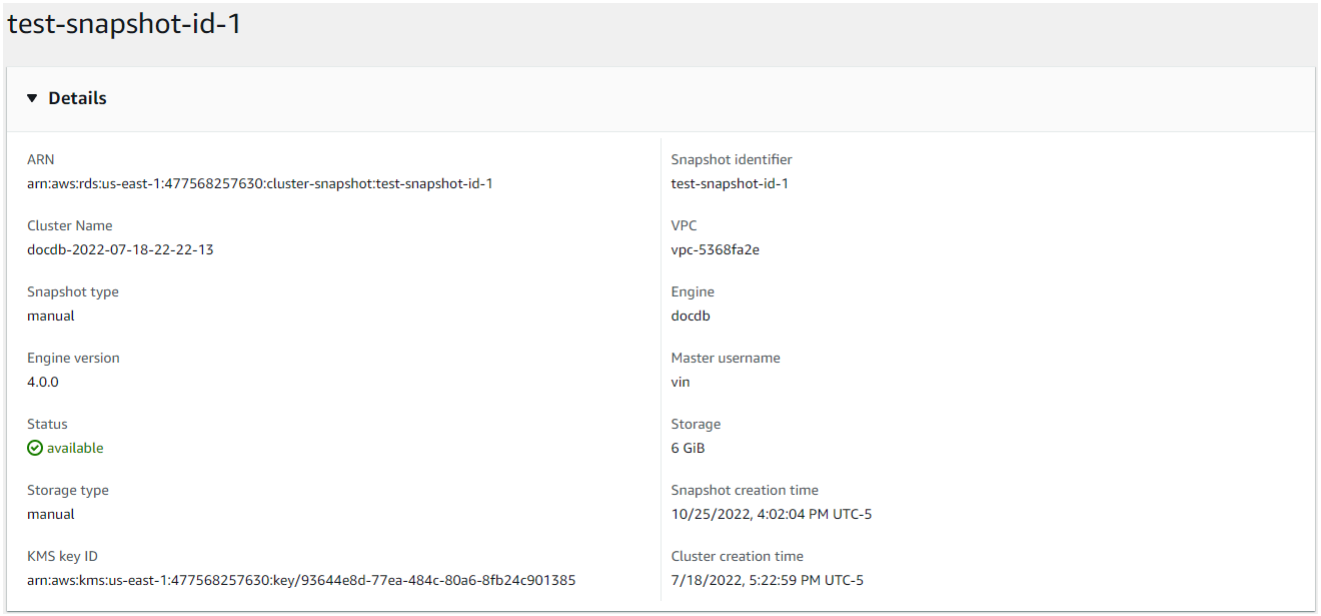
Using the AWS Management Console

を使用して特定の Elastic クラススナップショットに関する情報を表示するには AWS Management Console：

1. AWS Management Console にサインインし、Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[Snapshots] を選択してください。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ナビゲーションペインの左上隅にあるメニューアイコンを選択します。

3. [スナップショット] ページで、[スナップショット識別子] 列の名前をクリックして、リストから目的のスナップショットを選択します。
4. [詳細] にスナップショットの情報が表示されます。



Using the AWS CLI

を使用して特定の Elastic クラススナップショットに関する情報を表示するには AWS CLI、以下のパラメータを指定して `get-cluster-snapshot`オペレーションを使用します。

- **`--snapshot-arn`** — 必須 情報が必要なスナップショットの ARN 識別子。

次の例では、各 ############## を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Windows の場合:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

を使用して特定の Elastic クラスタースナップショットに関する情報を表示するには AWS CLI、以下のパラメータを指定して `get-cluster-snapshot`オペレーションを使用します。

- **`--snapshot-arn`** — 必須 情報が必要なスナップショットの ARN 識別子。

次の例では、各 ############## を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Windows の場合:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

を使用してすべての Elastic クラスタースナップショットに関する情報を表示するには AWS CLI、以下のパラメータを指定して `list-cluster-snapshots`オペレーションを使用します。

- **`--snapshot-type`** - オプションです。返されるクラスタースナップショットのタイプ。次のいずれかの値を指定できます。
  - `automated` - Amazon DocumentDB が AWS アカウントに対して自動的に作成したすべてのクラスタースナップショットを返します。

  - `manual` - AWS アカウント用に手動で作成したすべてのクラスタースナップショットを返します。
  - `shared` - AWS アカウントに共有されているすべての手動クラスタースナップショットを返します。
  - `public` - 公開とマークされたクラスタースナップショットをすべて返します。
- **`--next-token`** - オプションです。以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスにはこのマーカーを超え、 `max-results` で指定された値までのレコードのみが含まれます。
- **`--max-results`** - オプションです。レスポンスに含める結果の最大数。指定された `max-results` の値よりも多くの結果が存在する場合、ページ割りトークン (`next-token`) がレスポンスに含まれるため、残りの結果を取得できます。
  - デフォルト: 100
  - 最小は 20、最大は 100 です

次の例では、各 *############* を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb-elastic list-cluster-snapshots \
    --snapshot-type  value \
    --next-token  value \
    --max-results 50
```

Windows の場合:

```
aws docdb-elastic list-cluster-snapshots ^
    --snapshot-type  value ^
    --next-token  value ^
    --max-results 50
```

スナップショットからの Elastic クラスターの復元

このセクションでは、 AWS Management Console または を使用して、スナップショットから Elastic クラスターを復元する方法を次の手順 AWS CLI で説明します。

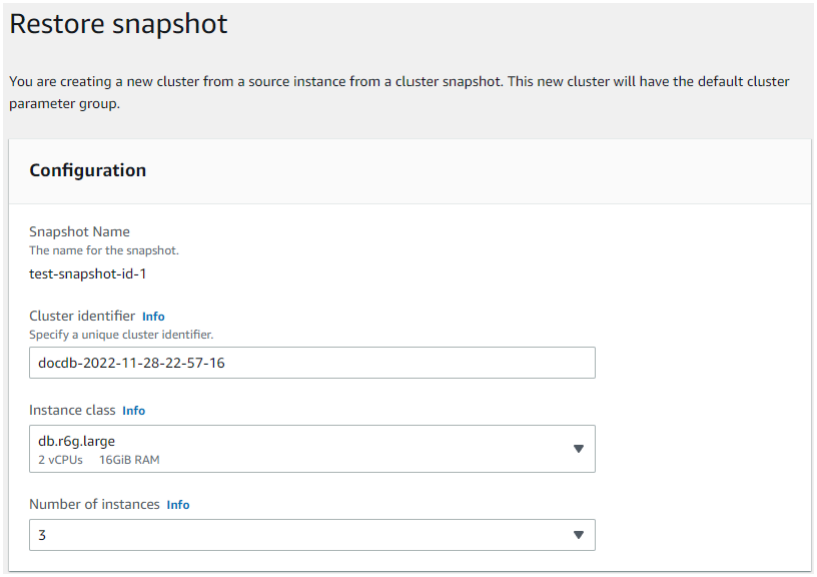
Using the AWS Management Console

AWS Management Consoleを試用してスナップショットから Elastic クラスターから復元するには:

1. AWS Management Console にサインインし、Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[Snapshots] を選択してください。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ナビゲーションペインの左上隅にあるメニューアイコンを選択します。

3. [スナップショット識別子] 列で、クラスターの復元に使用するスナップショットの左側にあるボタンを選択します。
4. [アクション]、[復元] の順に選択します。



5. [スナップショットの復元] ページで、[クラスター識別子] フィールドに新しいクラスターの名前を入力します。

> **Note**
>
> 手動スナップショットの復元では、新しいクラスターを作成する必要があります。

6. [仮想プライベートクラウド (VPC)] フィールドのドロップダウンリストから VPC を選択します。
7. [サブネット]と [VPC セキュリティグループ]では、デフォルトを使用するか、任意の 3 つのサブネットと最大 3 つまでの VPC セキュリティグループ (最低 1 つ) を選択できます。
8. クラスター設定に問題がなければ、[Restore cluster (クラスターの復元)] を選択し、クラスターが復元されるのを待ちます。

Using the AWS CLI

を使用してスナップショットから Elastic クラスターを復元するには AWS CLI、以下のパラメータを指定して `restore-cluster-from-snapshot`オペレーションを使用します。

- **`--cluster-name`** — 必須 作成時または最終変更時に入力された Elastic クラスターの現在名。
- **`--snapshot-arn`** — 必須 クラスターの復元に使用されているスナップショットの ARN 識別子。
- **`--vpc-security-group-ids`** - オプションです。1 つまたは複数の Amazon EC2 と Amazon 仮想プライベートクラウド (VPC) セキュリティグループをクラスターに関連付けます。
- **`--kms-key-id`** - オプションです。暗号化されたクラスターの KMS キー識別子を設定します。

  KMS キー識別子は、 AWS KMS 暗号化キーの Amazon リソースネーム (ARN) です。新しいクラスターの暗号化に使用する KMS 暗号化キーを所有する Amazon Web Services アカウントと同じアカウントを使用してクラスターを作成する場合、KMS 暗号化キーの ARN の代わりに KMS キーエイリアスを使用できます。

  KmsKeyId で暗号化キーが指定がなく、かつ `StorageEncrypted` パラメータが true の場合、Amazon DocumentDB はデフォルトの暗号化キーを使用します。
- **`--subnet-ids`** - オプションです。ネットワークサブネット ID。

次の例では、各 *############* を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb-elastic restore-cluster-from-snapshot \
     --cluster-name elastic-sample-cluster \
```

```
    --snapshot-arn sampleResourceName \
    --vpc-security-group-ids value ec-65f40350 \
    --kms-key-id arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 \
    --subnet-ids subnet-9253c6a3, subnet-9f1b5af9
```

Windows の場合:

```
aws docdb-elastic restore-cluster-from-snapshot ^
    --cluster-name elastic-sample-cluster ^
    --snapshot-arn sampleResourceName ^
    --vpc-security-group-ids value ec-65f40350 ^
    --kms-key-id arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 ^
    --subnet-ids subnet-9253c6a3, subnet-9f1b5af9
```

Elastic クラススナップショットのコピー

Amazon DocumentDB では、Elastic クラスターの手動スナップショットと自動スナップショットを同じリージョン内、あるいは同じアカウント内でコピーできます。このセクションでは、 AWS Management Console または を使用して Elastic クラススナップショットをコピーする方法について説明します AWS CLI。

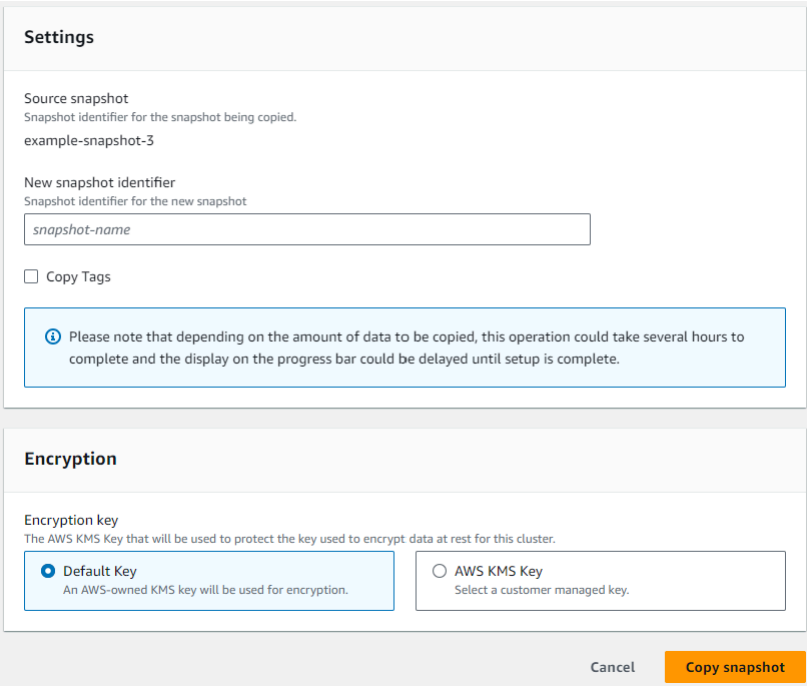
Using the AWS Management Console

を使用して Elastic クラススナップショットをコピーするには AWS Management Console：

1. AWS Management Console にサインインし、Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[Snapshots] を選択してください。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ナビゲーションペインの左上隅にあるメニューアイコンを選択します。

3. [スナップショット識別子] 列で、コピーしたいスナップショットの左側にあるボタンを選択します。
4. [アクション]、[コピー] の順に選択します。



5. [新しいスナップショット識別子] で、新しいスナップショットの名前を入力します。
6. コピー元の Elastic クラススナップショットからコピー先の Elastic クラススナップショットにすべてのタグをコピーしたい場合は、[タグのコピー] のチェックボックスをオンにします。
7. 暗号化では、デフォルトの KMS AWS キーまたは任意の KMS キーを選択します。任意の KMS キーを選択することで、過去に作成した既存の KMS キーを選択したり、あるいは新しい KMS キーを作成することができます。
8. 完了したら、[スナップショットのコピー] を選択します。

Using the AWS CLI

を使用して Elastic クラススナップショットをコピーするには AWS CLI、以下のパラメータを指定して `copy-cluster-snapshot`オペレーションを使用します。

- **`--source-db-cluster-snapshot-identifier`** — 必須 コピーしたい 既存の Elastic クラススナップショットの識別子。その Elastic クラスターのスナップショットが実際に存在し、利用可能な状態である必要があります。このパラメータは大文字と小文字が区別されません。
- **`--target-db-cluster-snapshot-identifier`** — 必須 既存のクラススナップショットから作成する新しいクラススナップショット向けの識別子。このパラメータは大文字と小文字が区別されません。

コピー先のスナップショットの名前付けに関する制約事項:

- 既存のスナップショットの名前にすることはできません。
- 長さは、[1 ~ 63] 個の英字、数字、またはハイフンです。
- 1 字目は英字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

次の例では、各 *############* を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb-elastic copy-cluster-snapshot \
     --source-cluster-snapshot-arn <sample ARN> \
     --target-cluster-snapshot-name my-target-copied-snapshot
```

Windows の場合:

```
aws docdb-elastic copy-cluster-snapshot ^
     --source-cluster-snapshot-arn <sample ARN> ^
     --target-cluster-snapshot-name my-target-copied-snapshot
```

Elastic クラススナップショットの削除

このセクションでは、 AWS Management Console または を使用して Elastic クラススナップショットを削除する方法について説明します AWS CLI。

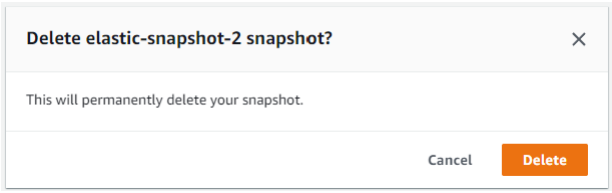
Using the AWS Management Console

AWS Management Consoleを試用してスナップショットから Elastic クラスターから復元するには:

1. AWS Management Console にサインインし、Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[Snapshots] を選択してください。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ナビゲーションペインの左上隅にあるメニューアイコンを選択します。

3. [スナップショット識別子] 列で、クラスターの復元に使用するスナップショットの左側にあるボタンを選択します。
4. [Actions (アクション)] を選択してから [Delete (削除)] を選択します。



5. [「snapshot-name」スナップショットの削除] ダイアログで、[削除] を選択します。

Using the AWS CLI

を使用して Elastic クラススナップショットを削除するには AWS CLI、次のパラメータを指定して `delete-cluster-snapshot`オペレーションを使用します。

- **`--snapshot-arn`** — 必須 クラスターの復元に使用されているスナップショットの ARN 識別子。

次の例では、各 *############* を独自の情報に置き換えます。

Linux、macOS、Unix の場合:

```
aws docdb-elastic delete-cluster-snapshot \
     --snapshot-arn sampleResourceName
```

Windows の場合:

```
aws docdb-elastic delete-cluster-snapshot ^
     --snapshot-arn sampleResourceName
```

Elastic クラススナップショットの自動バックアップの管理

Amazon DocumentDB は、お使いの各 Elastic クラスターのスナップショットを毎日作成します。新規または既存の Elastic クラススナップショット設定で、希望するバックアップウィンドウ (タイミング) とバックアップ保持期間を指定できます。このセクションでは、 AWS Management Console または を使用して、Elastic クラススナップショットに自動バックアップパラメータを設定する方法について説明します AWS CLI。

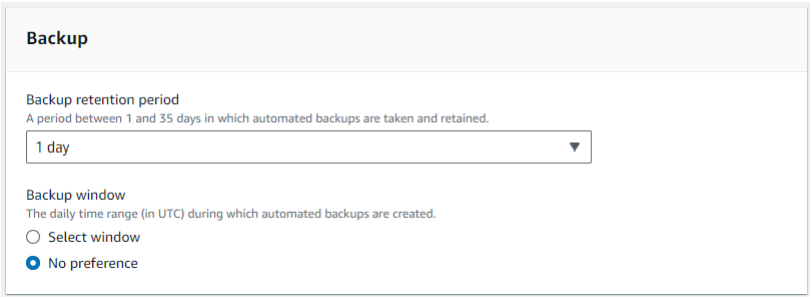
Using the AWS Management Console

を使用して新しい Elastic クラススナップショットの自動バックアップを設定するには AWS Management Console：

1. AWS Management Console にサインインし、Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択します。

> **Tip**
> 画面の左側にナビゲーションペインが表示されない場合は、ナビゲーションペインの左上隅にあるメニューアイコンを選択します。

3. [クラスター識別子] 列で、バックアップ設定を変更したいクラスターの左側にあるボタンを選択します。
4. [アクション]、[変更] の順にクリックします。
5. [バックアップ] セクションで、該当するバックアップ要件に従ってフィールドを編集します。



   a. [バックアップ保持期間] - リストで、このクラスターの自動バックアップを削除するまで保持する日数を選択します。

b. [バックアップウィンドウ] - Amazon DocumentDB によってこのクラスターのバックアップが作成される毎日の時刻と期間を設定します。

i. バックアップの作成タイミングと期間を設定したい場合は、[ウィンドウを選択] を選択します。

[開始時刻] - 最初のリストで、自動バックアップを開始する時間 (UTC 時) を選択します。2 番目のリストで、自動バックアップを開始する時間 (UTC 分) を選択します。

[期間] - リストで、自動バックアップの作成に割り当てる時間を選択します。

ii. Amazon DocumentDB 動作に従ってバックアップの作成タイミングや期間を自動選択する場合は、[設定なし] を選択します。

6. 完了したら [クラスターの変更] を選択します。

Using the AWS CLI

を使用して新しい Elastic クラススナップショットの自動バックアップを設定するには AWS CLI、以下のパラメータを指定して `create-cluster-snapshot`オペレーションを使用します。

- **`--preferred-backup-window`** - オプションです。毎日自動バックアップを作成する希望の時間帯。デフォルトは、それぞれ 8 時間の時間ブロックからランダムに選択された 30 分のウィンドウです AWS リージョン。

  制約:

  - `hh24:mi-hh24:mi` の形式である必要があります。
  - 時間は協定世界時 (UTC) である必要があります。
  - 必要なメンテナンス期間と競合してはいけません。
  - 少なくとも 30 分以上必要です。

- **`--backup-retention-period`** - オプション。自動バックアップを保管する日数。デフォルト値は 1 です。

  制約:

  - 最小でも 1 を指定する必要があります。
  - 1 から 35 の範囲の数を指定できます。

> **Note**
>
> 自動バックアップは、そのクラスターが「アクティブ」状態である場合にのみ実行されます。

> **Note**
>
> `aws docdb-elastic update-cluster` コマンドを使用して、既存の Elastic クラスターの `preferred-backup-window` および `backup-retention-period` パラメータを変更することもできます。

次の例では、各 *###############* を独自の情報に置き換えます。

次の `create-cluster` 例では、自動バックアップの保持期間が *7* 日間で、優先バックアップウィンドウが *18:00#18:30 UTC* である Amazon DocumentDB Elastic クラスターの*########*を作成します。

Linux、macOS、Unix の場合:

```
aws docdb-elastic create-cluster \
     --cluster-name sample-cluster \
     --shard-capacity 2 \
     --shard-count 2 \
     --admin-user-name SampleAdmin \
     --auth-type PLAIN_TEXT \
     --admin-user-password SamplePass123! \
     --preferred-backup-window 18:00-18:30 \
     --backup-retention-period 7
```

Windows の場合:

```
aws docdb-elastic create-cluster ^
     --cluster-name sample-cluster ^
     --shard-capacity 2 ^
     --shard-count 2 ^
     --admin-user-name SampleAdmin ^
     --auth-type PLAIN_TEXT ^
     --admin-user-password SamplePass123! ^
```

```
    --preferred-backup-window 18:00-18:30 ^
    --backup-retention-period 7
```

Amazon DocumentDB Elastic クラスターの停止と開始

Amazon DocumentDB Elastic クラスターの停止および開始機能は、開発環境やテスト環境のコストを管理するのに便利です。Amazon DocumentDB を使用するたびに Elastic クラスターを作成したり削除する代わりに、必要ではないときにクラスターを一時的に停止させることができます。必要になった時にクラスターを再開すれば、テストを再開することができます。

トピック

- Elastic クラスターの停止と開始の概要
- 停止中の Elastic クラスターのみで実行できるオペレーション。

Elastic クラスターの停止と開始の概要

Amazon DocumentDB Elastic クラスターが必要ではない期間は、そのクラスターを停止することができます。そして、クラスターを使用する必要がある時はいつでもクラスターを開始できます。開始と停止は、継続的な可用性を必要としない開発、テスト、または類似のアクティビティに使用される Elastic クラスターのセットアップと解放のプロセスを簡素化します。Elastic クラスターを停止および開始するには、 AWS Management Console または を 1 つのアクション AWS CLI で使用します。

Elastic クラスターの停止中は、そのクラスターのストレージボリュームは変更されません。指定された保持期間内でのストレージ、手動のスナップショット、および自動化されたバックアップストレージに対してのみ課金されます。Amazon DocumentDB は、7 日後に Elastic クラスターを自動的に再開するため、必要なメンテナンスの更新が遅延することはありません。7 日後にクラスターが再開すると、その Elastic クラスター内のインスタンスに対する課金が再開します。クラスターの停止中は、そのストレージボリュームに対するクエリを実行できません。クエリを実行するにはクラスターが使用可能状態であることが必要であるからです。

Amazon DocumentDB Elastic クラスターの停止中は、そのクラスターには一切変更を加えることはできません。これには、クラスターの削除も含まれます。

Using the AWS Management Console

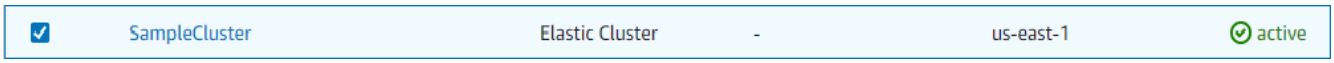
次の手順では、利用可能な状態にある Elastic クラスターの停止方法と、停止状態にある Elastic クラスターの再開方法を示します。

Amazon DocumentDB Elastic クラスターを停止または開始するには

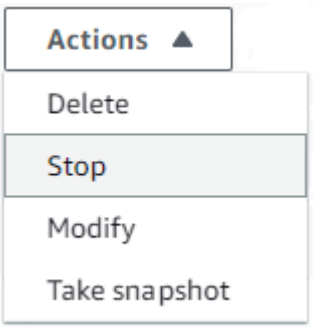
1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで クラスター を選択します。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

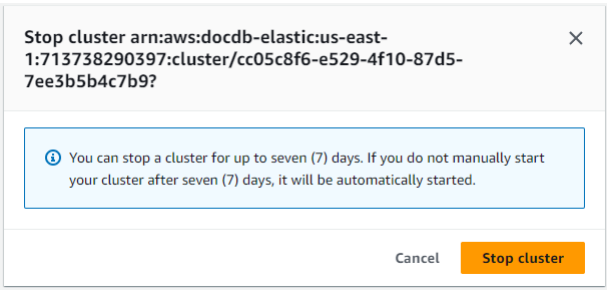
3. クラスターのリストで、停止または開始するクラスター名の左側にあるボタンを選択します。



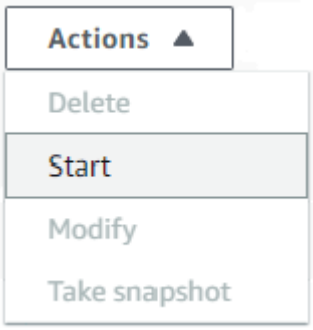
4. [Actions (アクション)] を選択したら、このクラスターで実行するアクションを選択します。
   - クラスターを停止したいときに、このクラスターが利用可能な場合:
     a. [停止] を選択します。



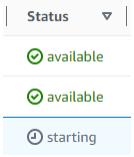
     b. 確認ダイアログで [クラスターの停止] を選択して Elastic クラスターの停止を確定するか、もしくは [キャンセル] を選択してクラスター動作を継続させます。



- クラスターを開始したいときにこのクラスターが停止されている場合、[Start (開始)] を選択します。



5. Elastic クラスターのステータスをモニタリングします。クラスターの再開を選択すると、クラスターが利用可能な状態にあればクラスターの使用を再開できます。詳細については、「クラスターのステータスの評価」を参照してください。



Using the AWS CLI

次のコード例では、アクティブもしくは利用可能な状態にある Elastic クラスターを停止する方法と、停止状態にある Elastic クラスターを再開する方法を示します。

を使用して Elastic クラスターを停止するには AWS CLI、 `stop-cluster`オペレーションを使用します。停止しているクラスターを開始するには、 `start-cluster` オペレーションを使用します。どちらのオペレーションも `--cluster-arn` パラメータを使用します。

パラメータ:

- **`--cluster-arn`** — 必須 停止もしくは再開したい Elastic クラスターの ARN 識別子。

Example — を使用して Elastic クラスターを停止するには AWS CLI

次の例では、各 `##############` を独自の情報に置き換えます。

次のコードにより、ARN が `arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2` である Elastic クラスターを停止します。

> **Note**
>
> これには、その Elastic クラスターがアクティブまたは利用可能な状態でなければなりません。

Linux、macOS、Unix の場合:

```
aws docdb-elastic stop-cluster \
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Windows の場合:

```
aws docdb-elastic stop-cluster ^
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Example — を使用して Elastic クラスターを起動するには AWS CLI

次の例では、各 `##############` を独自の情報に置き換えます。

次のコードにより、ARN が `arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2` である Elastic クラスターを再開します。

> **Note**
>
> これには、その Elastic クラスターが現在停止状態である必要があります。

Linux、macOS、Unix の場合:

```
aws docdb-elastic start-cluster \
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Windows の場合:

```
aws docdb-elastic start-cluster ^
```

```
    --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

停止中の Elastic クラスターのみで実行できるオペレーション。

クラスターが停止中は、Amazon DocumentDB Elastic クラスターの設定を変更することはできません。そのような管理アクションを実行する前に、クラスターを開始する必要があります。

Amazon DocumentDB は、停止中の Elastic クラスターにスケジュールされているメンテナンスについては、クラスターの再開後に適用します。Amazon DocumentDB は停止した Elastic クラスターを 7 日後に自動的に再開するため、メンテナンス実施状況が過剰に遅延することはありません。Elastic クラスターが再開すると、クラスター内のシャードへの課金が再開します。

Elastic クラスターの停止中、Amazon DocumentDB は自動バックアップを一切実行せず、またバックアップ保持期間を延長することはありません。

Amazon DocumentDB エラスティッククラスターのメンテナンス

トピック

- 保留中の Elastic クラスターメンテナンスアクションの表示
- Elastic クラスターエンジンの更新
- Elastic クラスターのオペレーティングシステムの更新

Amazon DocumentDB は定期的に Amazon DocumentDB エラスティッククラスターリソースのメンテナンスを実行します。ほとんどの場合、メンテナンスにはデータベースエンジンの更新 (エラスティッククラスターメンテナンス) またはエラスティッククラスターの基盤となるオペレーティングシステムの更新 (OS 更新) が含まれます。データベースエンジンの更新は必須のパッチであり、セキュリティ修正、バグ修正、データベースエンジンの強化などが含まれます。一方ほとんどのオペレーティングシステムパッチはオプションですが、しばらく適用しないで放置していると、セキュリティ体制の維持のためにパッチが必要となった結果、自動的に適用されることがあります。そのため、利用可能ならすぐに Amazon DocumentDB エラスティッククラスターにオペレーティングシステムの更新を適用することをお勧めします。

データベースエンジンパッチでは、Amazon DocumentDB エラスティッククラスターを短時間オフラインにする必要があります。使用可能になると、これらのパッチは Amazon DocumentDB エラスティッククラスターの今後のスケジュールされたメンテナンスウィンドウ中に自動的に適用されるようにスケジュールされます。

Elastic クラスターには、それぞれ独自のメンテナンスウィンドウがあります。すぐに適用しないことを選択した Elastic クラスターの変更は、メンテナンスウィンドウ中に適用されます。デフォルトでは、エラスティッククラスターを作成すると、Amazon DocumentDB はエラスティッククラスターのメンテナンスウィンドウを割り当てます。Elastic クラスターを作成するときに、メンテナンスウィンドウを選択できます。また、ビジネススケジュールや業務に合わせてメンテナンスウィンドウをいつでも変更できます。一般的には、メンテナンスによるアプリケーションへの影響を最小限に抑えるメンテナンスウィンドウ (夜間、週末など) を選択することをお勧めします。

保留中の Elastic クラスターメンテナンスアクションの表示

を使用して、Elastic クラスターでメンテナンス更新が利用可能かどうかを確認できます AWS CLI。

更新が利用可能な場合は、以下のいずれかの操作を行います。

- 現在次のメンテナンスウィンドウにスケジュールされているメンテナンスアクションを延期することができます (OS パッチのみ)。
- メンテナンスアクションをすぐに適用する。
- メンテナンスアクションを次のメンテナンスウィンドウ中に開始するようにスケジュールを設定する。
- 選択した適用ウィンドウ中にメンテナンスアクションを開始するようにスケジュールします。

メンテナンスウィンドウによって、保留中のオペレーションを開始する時刻が決定されますが、これらのオペレーションの合計実行時間は制限されません。

次の AWS CLI オペレーションを使用して、保留中のメンテナンスアクションを決定します。保留中のすべてのメンテナンスアクションを一覧表示します。

```
aws docdb-elastic list-pending-maintenance-actions
```

このオペレーションからの出力は次のようになります (JSON 形式) 。

```
{
'ResourcePendingMaintenanceActions': [
    {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
```

```
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
    },
],
'NextToken': 'string'
}
```

特定の で保留中のメンテナンスアクション (存在する場合) を取得しますresourceArn。

```
aws docdb-elastic get-pending-maintenance-action --resource-arn string-arn
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            }
        ]
    }
}
```

パラメータ :

- ResourceArn— 保留中のメンテナンスアクションが適用されるリソースの Amazon DocumentDB Amazon リソースネーム (ARN)。
- Action— リソースに適用されている保留中のメンテナンスアクション。

  有効な値:

  - ENGINE_UPDATE
  - ENGINE_UPGRADE
  - SECURITY_UPDATE
  - OS_UPDATE
  - MASTER_USER_PASSWORD_UPDATE
- AutoAppliedAfterDate— この日付以降の最初のメンテナンスウィンドウ。この場合、NEXT_MAINTENANCE OPT_INは無視されます。
- ForcedApplyDate— メンテナンスウィンドウに関係なく適用されます。この場合、 IMMEDIATE OPT_INは無視されます。
- OptInStatus - オプトインリクエストのタイプを指定するか、オプトインクエストを元に戻す値です。タイプ IMMEDIATE のオプトインリクエストは元に戻すことができません。

  有効な値:

  - IMMEDIATE - メンテナンスアクションをすぐに適用します。
  - NEXT_MAINTENANCE - リソースの次のメンテナンスウィンドウ中にメンテナンスアクションを適用します。
  - APPLY_ON— リソースの次のメンテナンスウィンドウに関係なく、指定された適用日にメンテナンスアクションを適用します。
  - UNDO_OPT_IN— 既存のリクエストをキャンセルNEXT_MAINTENANCEするか、APPLY_ONオプトインします。
- CurrentApplyDate— opt-in-type が の場合に表示されますAPPLY_ON。
- Description— メンテナンスアクションのオプションの説明。

Elastic クラスターエンジンの更新

Amazon DocumentDB では、メンテナンスオペレーションを適用するタイミングを選択することができます。Amazon DocumentDB が更新を適用するタイミングは、 を使用して決定できます AWS CLI。

保留中のメンテナンスアクションを適用します。

```
aws docdb-elastic apply-pending-maintenance-action
--resource-arn string-arn
--apply-action string-enum
--opt-in-type string-enum
```

```
[--apply-on string-date-range]
```

パラメータ :

- **--resource-arn**— 保留中のメンテナンスアクションが適用されるリソースの Amazon DocumentDB Amazon リソースネーム (ARN)。
- **--apply-action** - このリソースに適用する保留中のメンテナンスアクションです。

  有効な値:

  - ENGINE_UPDATE
  - ENGINE_UPGRADE
  - SECURITY_UPDATE
  - OS_UPDATE
  - MASTER_USER_PASSWORD_UPDATE
- **--opt-in-type** - オプトインリクエストのタイプを指定するか、オプトインクエストを元に戻す値です。タイプ IMMEDIATE のオプトインリクエストは元に戻すことができません。

  有効な値:

  - IMMEDIATE - メンテナンスアクションをすぐに適用します。
  - NEXT_MAINTENANCE - リソースの次のメンテナンスウィンドウ中にメンテナンスアクションを適用します。
  - APPLY_ON— リソースの次のメンテナンスウィンドウに関係なく、指定された適用日にメンテナンスアクションを適用します。
  - UNDO_OPT_IN— 既存のリクエストをキャンセルNEXT_MAINTENANCEするか、APPLY_ONオプトインします。
- **[--apply-on]**— opt-in-type が の場合は必須ですAPPLY_ON。形式: yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm (このオプションでは UTC 時間を使用します。 開始時刻は、最短 30 分から最長 14 日間、または保留中のアクションの強制/適用日のいずれか早い方から将来の任意の時刻にすることができます。 開始から終了までの時間は、最短で 30 分、最長で 8 時間です) 。

このオペレーションからの出力は次のようになります (JSON 形式) 。

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
```

```
            {
                'Action': 'SECURITY_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'IMMEDIATE',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
 }
}
```

パラメータ :

- ResourceArn— 保留中のメンテナンスアクションが適用されるリソースの Amazon DocumentDB Amazon リソースネーム (ARN)。
- Action— リソースに適用されている保留中のメンテナンスアクション。

  有効な値:

  - ENGINE_UPDATE
  - ENGINE_UPGRADE
  - SECURITY_UPDATE
  - OS_UPDATE
  - MASTER_USER_PASSWORD_UPDATE
- AutoAppliedAfterDate— この日付以降の最初のメンテナンスウィンドウ。この場合、NEXT_MAINTENANCE OPT_INは無視されます。
- ForcedApplyDate— メンテナンスウィンドウに関係なく適用されます。この場合、 IMMEDIATE OPT_INは無視されます。
- OptInStatus - オプトインリクエストのタイプを指定するか、オプトインクエストを元に戻す値です。タイプ IMMEDIATE のオプトインリクエストは元に戻すことができません。

  有効な値:

  - IMMEDIATE - メンテナンスアクションをすぐに適用します。
  - NEXT_MAINTENANCE - リソースの次のメンテナンスウィンドウ中にメンテナンスアクションを適用します。
  - APPLY_ON— リソースの次のメンテナンスウィンドウに関係なく、指定された適用日にメンテナンスアクションを適用します。

  - UNDO_OPT_IN— 既存のリクエストをキャンセルNEXT_MAINTENANCEするか、APPLY_ONオプトインします。
- CurrentApplyDate— opt-in-type が の場合に表示されますAPPLY_ON。
- Description— メンテナンスアクションのオプションの説明。

日付の適用

各メンテナンスアクションにはそれぞれ、保留中のメンテナンスアクションを記述する際に確認できる適用日が設定されています。から保留中のメンテナンスアクションの出力を読み取ると AWS CLI、次の 3 つの日付が表示されます。

- CurrentApplyDate - メンテナンスアクションが即座に、または次のメンテナンスウィンドウに適用される日付です。メンテナンスがオプションの場合、この値は null にすることができます。
- ForcedApplyDate - メンテナンスウィンドウとは関係なく、メンテナンスが自動的に適用される日付です。
- AutoAppliedAfterDate - クラスターのメンテナンスウィンドウでメンテナンスが適用される日付です。

ユーザー作成のメンテナンスアクション

Amazon DocumentDBelastic クラスターユーザーは、クラスター設定の更新を開始できます。

クラスターのプライマリパスワードの更新

```
aws docdb-elastic update-cluster
--cluster-arn string-arn
[--admin-user-password string]
[--auth-type string-enum]
[--apply-method string-enum]
[--apply-on string-date-range]
#... other parameters of the API that follow here are not relevant for this
 configuration
```

パラメータ：

- **--cluster-arn**— メンテナンスアクションが適用されるリソースの Amazon DocumentDB Amazon リソースネーム (ARN)。

- **[--admin-user-password]**— 管理者ユーザーに関連付けられたパスワード。
- **[--auth-type]**— Elastic クラスターへのアクセスに使用されるパスワードを取得する場所を決定するために使用される認証タイプ。有効なタイプは、PLAIN_TEXT または SECRET_ARN です。
- **[--apply-method]**— 適用するメソッドのタイプを指定する値。指定できる値は IMMEDIATE と APPLY_ON です。デフォルトは IMMEDIATE です。
- **[--apply-on]**— apply-methodが の場合は必須ですAPPLY_ON。形式: yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm (このオプションでは UTC 時間を使用します。 開始時刻は、最短 30 分から最長 14 日間の将来の任意の時刻にすることができます。 開始から終了までの時間は、最短で 30 分、最長で 8 時間です) 。

このオペレーションからの出力は次のようになります (JSON 形式) 。

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'MASTER_USER_PASSWORD_UPDATE',
                'OptInStatus': 'APPLY_ON',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
 }
}
```

Amazon DocumentDB メンテナンスウィンドウを変更する

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。この間、Elastic クラスターは、システムの変更 (スケールストレージオペレーションの変更など) が適用され、停止が必要な場合にのみ使用できなくなります。必要な変更を行うのに必要な最小時間だけは使用できません。

デフォルトは、1 週間のうちのランダムな日に起こる、Amazon Web Services リージョンあたり 8 時間の範囲からランダムに選択された 30 分のウィンドウです。

メンテナンスウィンドウを変更するには、「」を参照してくださいElastic クラスター設定の変更。

Elastic クラスターのオペレーティングシステムの更新

Amazon DocumentDB エラスティッククラスターでは、オペレーティングシステムの更新が必要になる場合があります。Amazon DocumentDB は、データベースパフォーマンスと顧客の全体的なセキュリティ体制改善のために、オペレーティングシステムを新しいバージョンにアップグレードします。オペレーティングシステムの更新では、Amazon DocumentDB エラスティッククラスターのクラスターエンジンバージョンは変更されません。

Amazon DocumentDB エラスティッククラスターのほとんどのオペレーティングシステムの更新はオプションであり、適用する日付が設定されていません。ただし、これらの更新をしばらく適用しない場合、最終的に必要になり、クラスターのメンテナンスウィンドウ中に自動的に適用される可能性があります。これは、データベースのセキュリティ体制を維持するために行われます。予期しないダウンタイムを避けるため、Amazon DocumentDB エラスティッククラスターが使用可能になったらすぐにオペレーティングシステムの更新を適用し、ビジネスニーズに応じて都合の良いときにクラスターのメンテナンスウィンドウを設定することをお勧めします。

Amazon DocumentDB エラスティッククラスターで保管中のデータ暗号化

以下のトピックは、Amazon DocumentDB エラスティッククラスターの AWS Key Management Service 暗号化キーについて学習、作成、およびモニタリングするのに役立ちます。

トピック

- Amazon DocumentDB エラスティッククラスターが で許可を使用する方法 AWS KMS
- カスタマーマネージドキーを作成する
- Amazon DocumentDB エラスティック クラスター の暗号化キーのモニタリング
- 詳細

Amazon DocumentDB エラスティッククラスターは、キー管理のために AWS Key Management Service （ AWS KMS) と自動的に統合され、エンベロープ暗号化と呼ばれる方法を使用してデータを保護します。エンベロープ暗号化の仕組みの詳細については、AWS Key Management Service デベロッパーガイドの「エンベロープ暗号化」を参照してください。

AWS KMS key はキーの論理表現です。KMS キーには、キー ID、作成日、説明、キーステータスなどのメタデータが含まれます。KMS キーには、データの暗号化と復号に使用されるキーマテリアルも含まれています。KMS キーの詳細については、AWS Key Management Service デベロッパーガイドの「AWS KMS keys 」を参照してください。

Amazon DocumentDB エラスティッククラスターは、次の 2 種類のキーによる暗号化をサポートしています。

- AWS 所有キー — Amazon DocumentDB エラスティッククラスターは、デフォルトでこれらのキーを使用して、個人を特定できるデータを自動的に暗号化します。 AWS所有キーを表示、管理、使用したり、その使用を監査したりすることはできません。ただし、データを暗号化するキーを保護するためのアクションの実施やプログラムの変更を行う必要はありません。詳細については、AWS Key Management Service デベロッパーガイドの「AWS 所有キー」を参照してください。
- カスタマーマネージドキー — AWS KMS keys ユーザーが作成、所有、管理する対称 。この暗号化層はユーザーが完全に制御できるため、次のようなタスクを実行できます。
  - キーポリシーの策定と維持
  - IAM ポリシーとグラントの策定と維持
  - キーポリシーの有効化と無効化
  - キー暗号化マテリアルのローテーション
  - タグの追加
  - キーエイリアスの作成
  - 削除のためのキースケジューリング

  詳細については、「AWS Key Management Service デベロッパーガイド」の「カスタマーマネージドキー」を参照してください。

> **Important**
>
> Amazon DocumentDB がサポートしているのは対称 KMS キーのみであるため、クラスターの暗号化には対称暗号化 KMS キーを使用して必要があります。Amazon DocumentDB エラスティッククラスター内のデータの暗号化に非対称 KMS キー を使用しないでください。詳細については、AWS Key Management Service デベロッパーガイドの「AWS KMSの非対称キーの」を参照してください。
> Amazon DocumentDB がクラスターの暗号化キーにアクセスできなくなった場合、例えばキーへのアクセス権が失効した場合など、暗号化されたクラスターは終了状態になります。この場合、クラスターはバックアップからのみ復元できます。Amazon DocumentDB では、バックアップは常に 1 日間有効になります。また、暗号化された Amazon DocumentDB クラスターのキーを無効にすると、最終的にそのクラスタに対する読み取りおよび書き込みのアクセス権は失われます。Amazon DocumentDB で、アクセスできないキーで暗号化され

たクラスターが発生すると、クラスターが終了状態になります。この場合、クラスターは使用できなくなり、データベースの現在の状態を復元することはできません。クラスターを復元するには、Amazon DocumentDB の暗号化キーへのアクセスを再び有効にした後、バックアップからクラスターを復元します。

> **Important**
>
> 暗号化されたクラスターの KMS キーを作成した後は、そのクラスターを変更することはできません。暗号化されたエラスティッククラスターを作成する前に、暗号化キーの要件を確認してください。

Amazon DocumentDB エラスティッククラスターが で許可を使用する方法 AWS KMS

Amazon DocumentDB エラスティック クラスター には、カスタマーマネージドキーを使用するためのグラントが必要です。

カスタマーマネージドキーで暗号化されたクラスターを作成すると、Amazon DocumentDB エラスティッククラスターは`CreateGrant`リクエストを送信してユーザーに代わってグラントを作成します AWS KMS。の許可 AWS KMS は、Amazon DocumentDB エラスティッククラスターに顧客アカウントの KMS キーへのアクセスを許可するために使用されます。

Amazon DocumentDB エラスティッククラスターでは、以下の内部操作にカスタマーマネージドキーを使用するためのグラントが必要です。

- トラッカーまたはジオフェンスコレクションの作成時に入力された対称カスタマーマネージド KMS キー ID が有効であることを確認する AWS KMS リクエストを `DescribeKey`に送信します。
- カスタマーマネージドキーで暗号化されたデータキーを生成する AWS KMS には、に`GenerateDataKey`リクエストを送信します。
- `Decrypt` リクエストを に送信 AWS KMS して、暗号化されたデータキーを復号し、データの暗号化に使用できます。
- グラントへのアクセスの取り消しや、カスタマーマネージドキーに対するサービスのアクセスの取り消しは、いつでもできます。これを行うと、Amazon DocumentDB エラスティッククラスター

はカスタマーマネージドキーによって暗号化されたすべてのデータにアクセスできなくなり、そのデータに依存しているオペレーションが影響を受けます。

カスタマーマネージドキーを作成する

AWS Management Console または AWS KMS API を使用して、対称カスタマーマネージドキーを作成できます。

対称カスタマー管理キーの作成

AWS Key Management Service デベロッパーガイド にある 対称カスタマーマネージドキーの作成 ステップに従います。

キーポリシー

キーポリシーは、カスタマーマネージドキーへのアクセスを制御します。すべてのカスタマーマネージドキーには、キーポリシーが 1 つだけ必要です。このポリシーには、そのキーを使用できるユーザーとその使用方法を決定するステートメントが含まれています。キーポリシーは、カスタマーマネージドキーの作成時に指定できます。詳細については、AWS Key Management Service デベロッパーガイドの「AWS Key Management Service 概要」にあるKMS キーアクセス情報を参照してください。

Amazon DocumentDB エラスティック クラスター で顧客管理キーを使用するには、キーポリシーで次の API オペレーションを許可する必要があります。

- `kms:CreateGrant` - カスタマーマネージドキーに許可を追加します。この権限は、指定された KMS キーへのアクセスを制御します。これにより、必要な許可の付与オペレーションに対し Amazon Location Service がアクセスできるようにします。許可の詳細については、AWS Key Management Service デベロッパーガイドの「許可の使用 AWS KMS」 を参照してください。
- `kms:DescribeKey` — お客様が管理するキーの詳細を提供し、DocDB-Elastic がキーを検証できるようにします。
- `kms:Decrypt` – 暗号化されたデータにアクセスするために、Docdb-Elastic が保存された暗号化データキーを使用できるようにします。
- `kms:GenerateDataKey` – Docdb-Elastic が暗号化されたデータキーを生成して保存できるようにします。データキーはすぐに暗号化に使用されないからです。

詳細については、「 AWS Key Management Service デベロッパーガイド」の「キーポリシーでの AWS サービスのアクセス許可」および「キーアクセスのトラブルシューティング」を参照してください。

IAM ポリシーによるカスタマーマネージドキーアクセスの制限

KMS キーポリシーに加えて、IAM ポリシーで KMS キー権限を制限することもできます。

さまざまな方法で、IAM ポリシー をより厳しくすることができます。例えば、Amazon DocumentDB エラスティッククラスターを起点とするリクエストにのみカスタマーマネージドキーを使用できるようにするには、`docdb-elastic.<region-name>.amazonaws.com` 値で `kms:ViaService` 条件キーを使用します。

詳細については、AWS Key Management Service デベロッパーガイドの他のアカウントのユーザーに KMS キーの使用を許可するをご参照ください。

Amazon DocumentDB エラスティック クラスター の暗号化キーのモニタリング

Docdb Elastic リソースで AWS KMS key カスタマーマネージドキーを使用する場合、 AWS CloudTrail または Amazon CloudWatch Logs を使用して、Docdb Elastic が送信するリクエストを追跡できます AWS KMS。

次の例は`CreateGrant`、Amazon DocumentDB エラスティッククラスターによって呼び出される AWS KMS key オペレーションをモニタリングして、カスタマーマネージドキーによって暗号化されたデータにアクセス`DescribeKey`するための `GenerateDataKeyWithoutPlainText`、`Decrypt`、、および の AWS CloudTrail イベントです。

CreateGrant

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
```

```
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-05-09T23:04:20Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "docdb-elastic.amazonaws.com"
    },
    "eventTime": "2023-05-09T23:55:48Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "docdb-elastic.amazonaws.com",
    "userAgent": "docdb-elastic.amazonaws.com",
    "requestParameters": {
        "retiringPrincipal": "docdb-elastic.us-east-1.amazonaws.com",
        "granteePrincipal": "docdb-elastic.us-east-1.amazonaws.com",
        "operations": [
            "Decrypt",
            "Encrypt",
            "GenerateDataKey",
            "GenerateDataKeyWithoutPlaintext",
            "ReEncryptFrom",
            "ReEncryptTo",
            "CreateGrant",
            "RetireGrant",
            "DescribeKey"
        ],
        "keyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
```

```
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

GenerateDataKey

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-05-10T18:02:59Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "docdb-elastic.amazonaws.com"
    },
    "eventTime": "2023-05-10T18:03:25Z",
    "eventSource": "kms.amazonaws.com",
```

```
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "docdb-elastic.amazonaws.com",
    "userAgent": "docdb-elastic.amazonaws.com",
    "requestParameters": {
        "keySpec": "AES_256",
        "keyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Decrypt

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
```

```
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-05-10T18:05:49Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "docdb-elastic.amazonaws.com"
    },
    "eventTime": "2023-05-10T18:06:19Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "docdb-elastic.amazonaws.com",
    "userAgent": "docdb-elastic.amazonaws.com",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

DescribeKey

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
```

```
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-05-09T23:04:20Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "docdb-elastic.amazonaws.com"
    },
    "eventTime": "2023-05-09T23:55:48Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "docdb-elastic.amazonaws.com",
    "userAgent": "docdb-elastic.amazonaws.com",
    "requestParameters": {
        "keyId": "alias/SampleKmsKey"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
```

```
    "eventCategory": "Management"
}
```

詳細

次のリソースは、保管時のデータ暗号化についての詳細を説明しています。

- AWS KMS 概念の詳細については、「 AWS Key Management Service デベロッパーガイド」のAWS Key Management Service 「 の基本概念」を参照してください。
- AWS KMS セキュリティの詳細については、「 AWS Key Management Service デベロッパーガイド」の「 のセキュリティのベストプラクティス AWS Key Management Service」を参照してください。

エラスティッククラスター上のサービスにリンクされたロール

Amazon DocumentDB エラスティッククラスターは AWS Identity and Access Management 、(IAM) サービスにリンクされたロールを使用します。サービスにリンクされているロールは、Amazon DocumentDB エラスティッククラスターに直接リンクされれている特殊なタイプの IAM ロールです。サービスにリンクされたロールは Amazon DocumentDB エラスティッククラスターによって事前定義されており、サービスがユーザーに代わって他の AWS サービスを呼び出すために必要なすべてのアクセス許可が含まれています。

サービスにリンクされているロールを使用すると、必要な権限を手動で追加する必要がないため、Amazon DocumentDB の使用が簡単になります。Amazon DocumentDB エラスティッククラスターには、サービスにリンクされているロールのアクセス許可が定義されます。特に定義されていない限り、Amazon DocumentDB のみがそのロールを引き受けることができます。定義されたアクセス許可には、信頼ポリシーとアクセス許可ポリシーが含まれ、そのアクセス許可ポリシーを他の IAM エンティティにアタッチすることはできません。ロールを削除するには、まず関連リソースを削除します。これにより、リソースにアクセスするためのアクセス許可を誤って削除することができないため、Amazon DocumentDB エラスティッククラスターが保護されます。

サービスにリンクされているロールをサポートするその他のサービスについては、「IAM と連携するAWS のサービス」を参照し、[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探してください。サービスリンク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、リンクで [はい] を選択します。

エラスティッククラスターに対するサービスにリンクされたロールのアクセス許可

Amazon DocumentDB エラスティッククラスターは、 という名前のサービスにリンクされたロールを使用してAWS ServiceRoleForDocDB-Elastic、Amazon DocumentDB エラスティッククラスターがクラスターに代わって AWS サービスを呼び出すことを許可します。

このサービスにリンクされたロールには、アカウントで操作するためのアクセス許可を付与する AmazonDocDB-ElasticServiceRolePolicy というアクセス許可ポリシーがアタッチされています。ロールのアクセス許可ポリシーは、Amazon DocumentDB エラスティッククラスターが指定リソースに対して以下のアクションを実行することを 許可します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

> **Note**
>
> サービスリンク役割の作成、編集、削除を IAM エンティティ (ユーザー、グループ、役割など) に許可するにはアクセス許可を設定する必要があります。次のエラーメッセージ、「リソースを作成できません。サービスにリンクされたロールを作成するために必要なアクセス許可があることを確認します。それ以外の場合は、時間をおいてからもう一度お試しください。」が表示された場合は、以下のアクセス許可が有効であることを確認します。

```
{
"Action": "iam:CreateServiceLinkedRole",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/aws-service-role/docdb-elastic.amazonaws.com/
AWSServiceRoleForDocDB-Elastic",
    "Condition": {
"StringLike": {
"iam:AWSServiceName":"docdb-elastic.amazonaws.com"
        }
    }
}
```

詳細については、『AWS ID とアクセス管理ユーザーガイド』の「サービスにリンクされたロールのアクセス許可」を参照してください。

Amazon DocumentDB エラスティッククラスター用のサービスにリンクされたロールの作成

サービスリンクロールを手動で作成する必要はありません。DB インスタンスが作成されると、Amazon DocumentDB の Elastic クラスターはサービスにリンクされたロールを自動的に作成します。

Amazon DocumentDB エラスティッククラスター用のサービスにリンクされたロールの編集

Amazon DocumentDB のエラスティッククラスターでは、AWS ServiceRoleForDocDB-Elastic サービスにリンクされたロールを編集することはできません。サービスリンクロールの作成後は、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。詳細については、『AWS ID とアクセス管理ユーザーガイド』の「サービスにリンクされたロールの編集」を参照してください。

Amazon DocumentDB エラスティッククラスター用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、使用していないエンティティがアクティブにモニタリングされたり、メンテナンスされたりすることがなくなります。ただし、サービスにリンクされたロールを削除する前に、すべての クラスターを削除する必要があります。

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、まずそのロールにアクティブなセッションがないことを確認し、そのロールで使用されているリソースをすべて削除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 Management Console にサインインし、IAM コンソール 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] を選択します。次に、AWS ServiceRoleForDocDB-Elastic 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。

> **Note**
>
> Amazon DocumentDB が AWS ServiceRoleForDocDB-Elastic ロールを使用しているかどうか不明な場合は、ロールの削除を試みることができます。サービスがロールを使用している場合、削除は失敗し、ロールが使用されている AWS リージョン を表示できます。ロールが使用されている場合は、ロールを削除する前にセッションが終了するのを待つ必要があります。サービスにリンクされたロールのセッションを取り消すことはできません。AWS ServiceRoleForDocDB-Elastic ロールを削除する場合は、最初にすべてのクラスターを削除する必要があります。

すべてのクラスターの削除

Amazon DocumentDB コンソールからクラスターを削除する

1. AWS Management Console にサインインし、Amazon DocumentDB コンソール を開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。
3. 削除するクラスターを選択します。
4. [アクション] で、[削除] を選択します。
5. [最終スナップショットを作成しますか?] が表示されたら、[はい] または [いいえ] を選択します。
6. 前のステップで [はい] を選択した場合は、[最終スナップショット名] に最終スナップショットの名前を入力します。
7. [削除] を選択します。

> **Note**
>
> `AWS ServiceRoleForDocDB-Elastic` サービスにリンクされたロールは、IAM コンソール、IAM CLI、または IAM API を使用して削除することができます。詳細については、『AWS ID とアクセス管理ユーザーガイド』の「サービスにリンクされたロールの削除」を参照してください。

Amazon DocumentDB のモニタリング

AWS サービスのモニタリングは、システムを正常かつ最適に機能させる上で重要な部分です。AWS ソリューションのすべての部分からモニタリングデータを収集し、障害や劣化が発生した場合に、より簡単にデバッグと修正を行えるようにすることが賢明です。 AWS ソリューションのモニタリングを開始する前に、以下の質問について検討し、回答を作成することをお勧めします。

- どのような目的でモニタリングしますか?
- モニタリングの対象となるリソースは?
- これらのリソースをモニタリングする頻度は?
- どのモニタリングツールを利用しますか?
- モニタリングを実行するための責任者は?
- 問題が発生した場合に、だれにどのような方法で通知しますか?

現在のパフォーマンスパターンを理解するには、パフォーマンスの異常を識別し、問題に対処する方法を策定します。さまざまな時間帯に、さまざまなロード条件下でパフォーマンスメトリクスのベースラインを確立する必要があります。 AWS ソリューションをモニタリングするときは、今後の参照用、およびベースラインを確立するため、過去のモニタリングデータを保存することをお勧めします。

一般的に、パフォーマンスメトリクスの許容値は、ベースラインがどのようになっているか、アプリケーションによって何が実行されているかによって異なります。ベースラインからの一貫した差異またはトレンドになっている差異を調べます。メトリクスのタイプごとのアドバイスは以下のとおりです。

- CPU または RAM の大量使用 - CPU または RAM の値が高くても、アプリケーションの目標 (スループットや同時実行数など) に沿った想定内の値であれば、それは適切であると見なされます。
- ストレージボリュームの消費量 - 使用領域が一貫してストレージボリューム領域全体の 85% 以上である場合は、ストレージの消費量 (`VolumeBytesUsed`) を調査します。ストレージボリュームからデータを削除するか、別のシステムにデータをアーカイブすることで、領域を解放できないかどうかを判断します。詳細については、「Amazon DocumentDB のストレージ」および「Amazon DocumentDB のクォータと制限事項」を参照してください。
- ネットワークトラフィック - ネットワークトラフィックについてシステム管理者に問い合わせて、ドメインネットワークとインターネット接続に対する想定スループットを把握します。スループットが一貫して想定よりも低い場合は、ネットワークトラフィックを調べます。

- データベース接続数 - ユーザー接続数が多いことが、インスタンスのパフォーマンスが下がっていること、応答時間が長くなっていることに関連しているとわかった場合、データベース接続数を制限することを検討します。インスタンスの最適なユーザー接続数は、インスタンスのクラスと実行中のオペレーションの複雑さによって異なります。
- IOPS メトリクス - IOPS メトリクスの想定値はディスクの仕様とサーバーの設定によって異なるため、ベースラインを使用して一般的な値を把握します。値とベースラインとの差が一貫しているかどうかを調べます。最適な IOPS パフォーマンスを得るには、読み取りおよび書き込みオペレーションが最小限になるように、一般的な作業セットがメモリに収まることを確認してください。

Amazon DocumentDB (MongoDB 互換) は、Amazon DocumentDB クラスターとインスタンスの健全性とパフォーマンスを判断するためにモニタリングできる、さまざまな Amazon CloudWatch メトリクスを提供します。Amazon DocumentDB コンソール、CloudWatch API、Performance Insights など AWS CLI、さまざまなツールを使用して Amazon DocumentDB メトリクスを表示できます。

トピック

- Amazon DocumentDB クラスターのステータスのモニタリング
- Amazon DocumentDB インスタンスのステータスのモニタリング
- Amazon DocumentDB に関する推奨事項の表示
- Amazon DocumentDB イベントサブスクリプションを使用する
- Amazon DocumentDB と CloudWatch のモニタリング
- を使用した Amazon DocumentDB API コールのログ記録 AWS CloudTrail
- Amazon DocumentDB オペレーションのプロファイリング
- Performance Insights を使用したモニタリング

Amazon DocumentDB クラスターのステータスのモニタリング

クラスターのステータスは、クラスターの状態を示します。Amazon DocumentDB コンソールまたは AWS CLI `describe-db-clusters` コマンドを使用して、クラスターのステータスを表示できます。

トピック

- クラスターステータスの値
- クラスターのステータスのモニタリング

クラスターステータスの値

次の表は、クラスターのステータスの有効な値の一覧です。

クラスターステータス	説明
`active`	クラスターはアクティブです。このステータスは Elastic クラスターにのみ適用されます。
`available`	クラスターは正常で、使用可能です。このステータスはインスタンスベースのクラスターにのみ適用されます。
`backing-up`	クラスターは現在バックアップ中です。
`creating`	クラスターを作成しています。作成中はクラスターにアクセスできません。
`deleting`	クラスターを削除しています。削除中はクラスターにアクセスできません。
`failing-over`	プライマリインスタンスから Amazon DocumentDB レプリカへのフェイルオーバーが実行されています。
`inaccessible-encryption-credentials`	クラスターの暗号化または復号に使用される AWS KMS キーにはアクセスできません。
`maintenance`	クラスターにメンテナンス更新を適用しています。このステータスは、Amazon DocumentDB が事前に十分スケジュールした

クラスターステータス	説明
	クラスターレベルのメンテナンスに使用されます。
migrating	クラスタースナップショットは、クラスターに復元されています。
migration-failed	移行に失敗しました。
modifying	お客様がクラスターの変更をリクエストしたため、クラスターは変更中です。
renaming	お客様が名前の変更をリクエストしたため、クラスターの名前を変更中です。
resetting-master-credentials	お客様がリセットをリクエストしたため、クラスターのマスター認証情報をリセット中です。
upgrading	クラスターエンジンのバージョンをアップグレード中です。

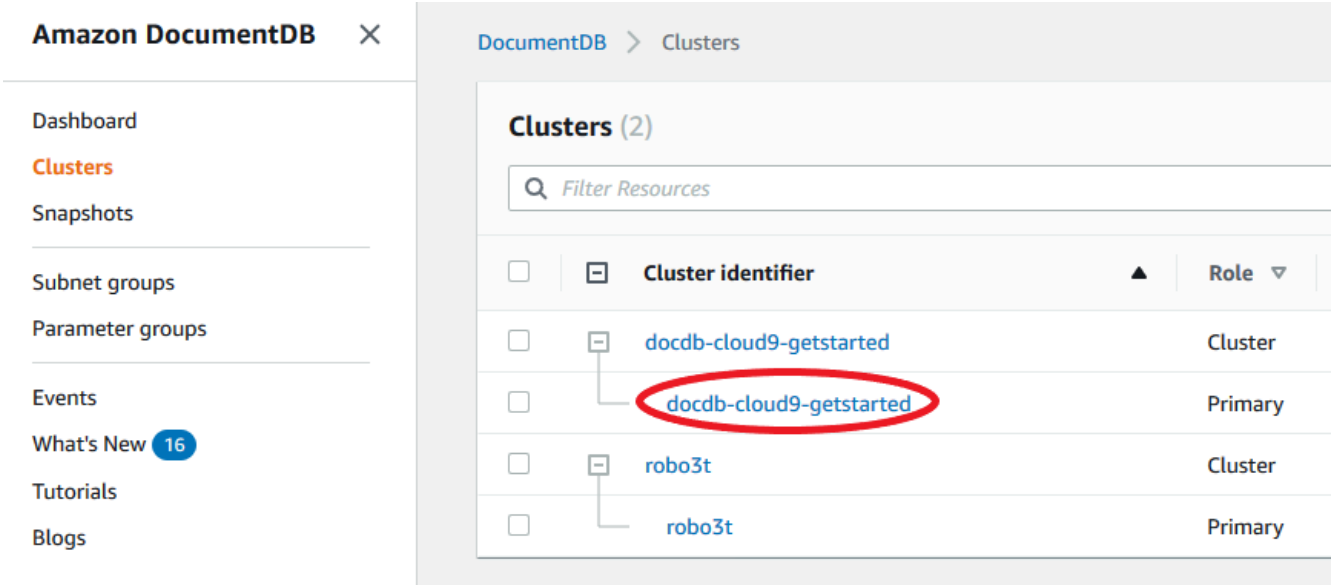
クラスターのステータスのモニタリング

Using the AWS Management Console

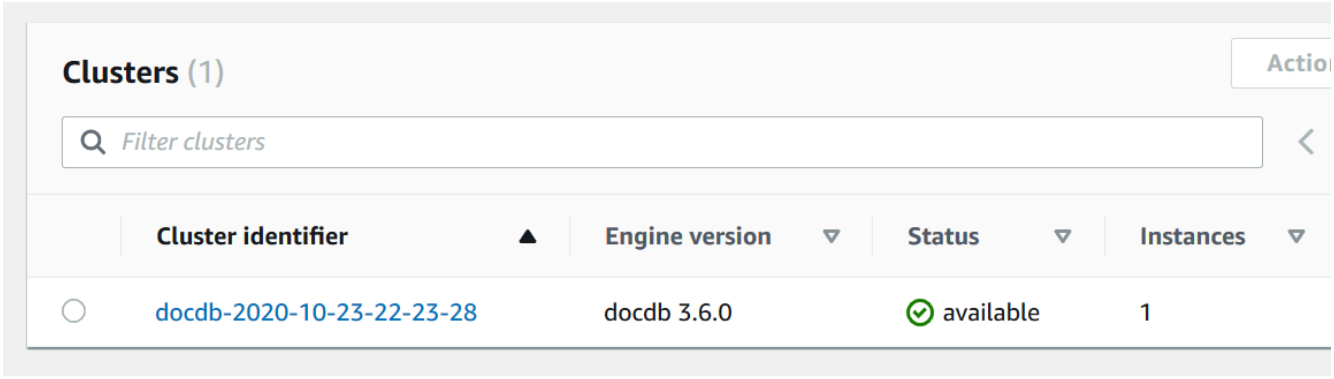
を使用してクラスターのステータス AWS Management Console を判断する場合は、次の手順を使用します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

3. Clusters ナビゲーションボックスで、[クラスター識別子] という列が表示されます。インスタンスは、以下のスクリーンショットのように、クラスターの下に表示されます。



4. [クラスター識別子] 列で、興味のあるインスタンスの名前を探します。次に、インスタンスのステータスを見つけるために、次に示すように行全体を参照して [ステータス] 列を見つけます。



Using the AWS CLI

を使用してクラスターのステータス AWS CLI を判断する場合は、 `describe-db-clusters`オペレーションを使用します。次のコードでは、クラスター `sample-cluster` のステータスを確認します。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster  \
```

```
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Windows の場合:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster  ^
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

このオペレーションによる出力は、次のようになります。

```
[
    [
        "sample-cluster",
        "available"
    ]
]
```

Amazon DocumentDB インスタンスのステータスのモニタリング

Amazon DocumentDB は、データベースに設定されている各インスタンスの現在の状態に関する情報を提供します。

Amazon DocumentDB インスタンスについて表示できるステータスには 3 つのタイプがあります。

- インスタンスのステータス: このステータスは、 のクラスターテーブルのステータス列に表示され AWS Management Console 、インスタンスの現在のライフサイクル条件が表示されます。[ステータス] 列に表示される値は `DescribeDBCluster` API レスポンスの `Status` フィールドから取得されます。
- インスタンスのヘルスステータス: このステータスは、 のクラスターテーブルのインスタンスのヘルス列に表示され、データの管理と取得を担当するコンポーネントであるデータベースエンジンが実行されているかどうか AWS Management Console を示します。[インスタンスヘルス] 列に表示される値は、Amazon CloudWatch `EngineUptime` システムメトリックスに基づいています。
- メンテナンスステータス: このステータスは、 のクラスターテーブルのメンテナンス列に表示され AWS Management Console 、インスタンスに適用する必要があるメンテナンスイベントのステータスを示します。メンテナンスステータスは他のインスタンスのステータスとは独立して、`PendingMaintenanceAction` API から取得されます。メンテナンスステータスの詳細については、「Amazon DocumentDB のメンテナンス」を参照してください。

トピック

- インスタンスのステータス値
- AWS Management Console または を使用したインスタンスステータスのモニタリング AWS CLI
- インスタンスのヘルスステータス値
- を使用したインスタンスのヘルスステータスのモニタリング AWS Management Console

インスタンスのステータス値

次の表は、インスタンスのステータス値と、ステータス別の請求方法の一覧です。請求対象がインスタンスとストレージであるか、ストレージのみであるか、または請求なしであるかを示します。インスタンスのすべてのステータスで、バックアップの使用は常に請求されます。

インスタンスの状態	請求対象	説明
`available`	請求対象	インスタンスは正常で、使用可能です。
`backing-up`	請求対象	インスタンスは現在バックアップ中です。
`configuring-log-exports`	請求対象	Amazon CloudWatch Logs へのログファイルの公開は、このインスタンスに対して有効または無効になっています。
`creating`	非請求対象	インスタンスは現在作成中です。作成中はインスタンスにアクセスできません。
`deleting`	非請求対象	インスタンスを削除中です。
`failed`	非請求対象	インスタンスに障害が発生し、Amazon DocumentDB は復旧できませんでした。データを復旧するには、インスタンスの復元可能な直近の時間までポイントインタイムリカバリを実行します。
`inaccessible-encryption-credentials`	非請求対象	インスタンスの暗号化または復号に使用される AWS KMS キーにアクセスできませんでした。

インスタンスの状態	請求対象	説明
incompatible-network	非請求対象	Amazon DocumentDB はインスタンスに対してリカバリーアクションを実行しようとしていますが、VPC がアクションの完了を妨げている状態であるため、実行することができません。この状態は、たとえば、サブネット内の利用可能なすべての IP アドレスが使用中で、Amazon DocumentDB がインスタンスの IP アドレスを取得できない場合に発生する可能性があります。
maintenance	請求対象	Amazon DocumentDB は、インスタンスにメンテナンス更新を適用しています。このステータスは、Amazon DocumentDB が事前に十分なスケジュールを組んでいるインスタンスレベルのメンテナンスに使用されます。このステータスを通じてお客様に追加のメンテナンスアクションを提示する方法を評価中です。
modifying	請求対象	インスタンスの変更をリクエストしたため、インスタンスを変更中です。
rebooting	請求対象	Amazon DocumentDB は、インスタンスの再起動を必要とするリクエスト、または Amazon DocumentDB プロセスのために、インスタンスを再起動しています。
renaming	請求対象	名前の変更をリクエストしたため、インスタンスの名前を変更中です。
resetting-master-credentials	請求対象	リセットをリクエストしたため、インスタンスのマスター認証情報をリセット中です。

インスタンスの状態	請求対象	説明
restore-error	請求対象	特定時点への復元またはスナップショットからの復元を実行する際に、インスタンスでエラーが発生しました。
starting	ストレージが請求対象	インスタンスを起動中です。
stopped	ストレージが請求対象	インスタンスが停止されました。
stopping	ストレージが請求対象	インスタンスを停止中です。
storage-full	請求対象	インスタンスが、ストレージ容量の割り当て分に到達しました。これは非常に重要なステータスであり、すぐに修正する必要があります。インスタンスを変更して、ストレージをスケールアップしてください。このような状況にならないように、ストレージスペースが少なくなってきたときに警告するよう Amazon CloudWatch のアラームを設定します。

AWS Management Console または を使用したインスタンスステータスのモニタリング AWS CLI

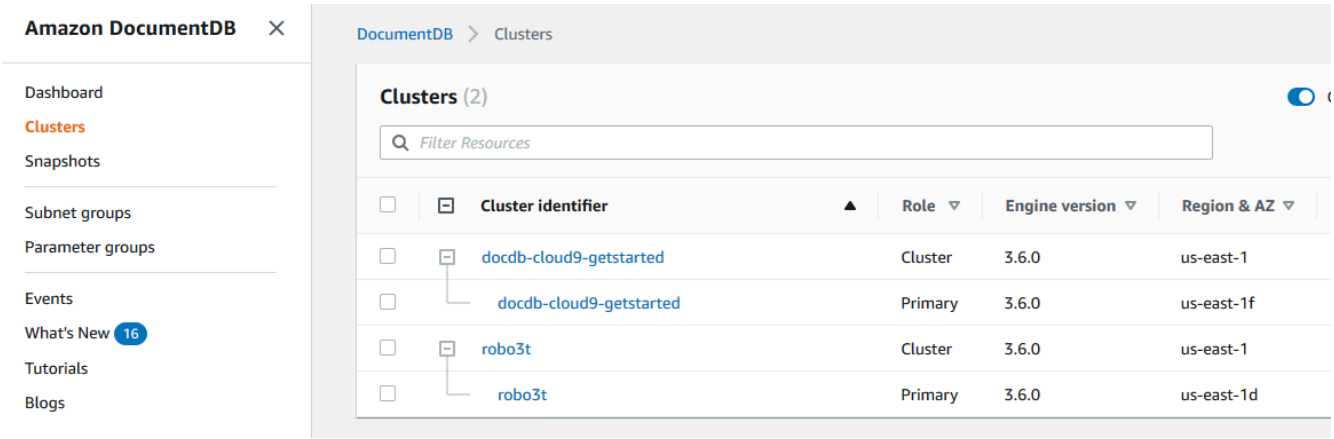
AWS Management Console または を使用して AWS CLI 、インスタンスのステータスをモニタリングします。

Using the AWS Management Console

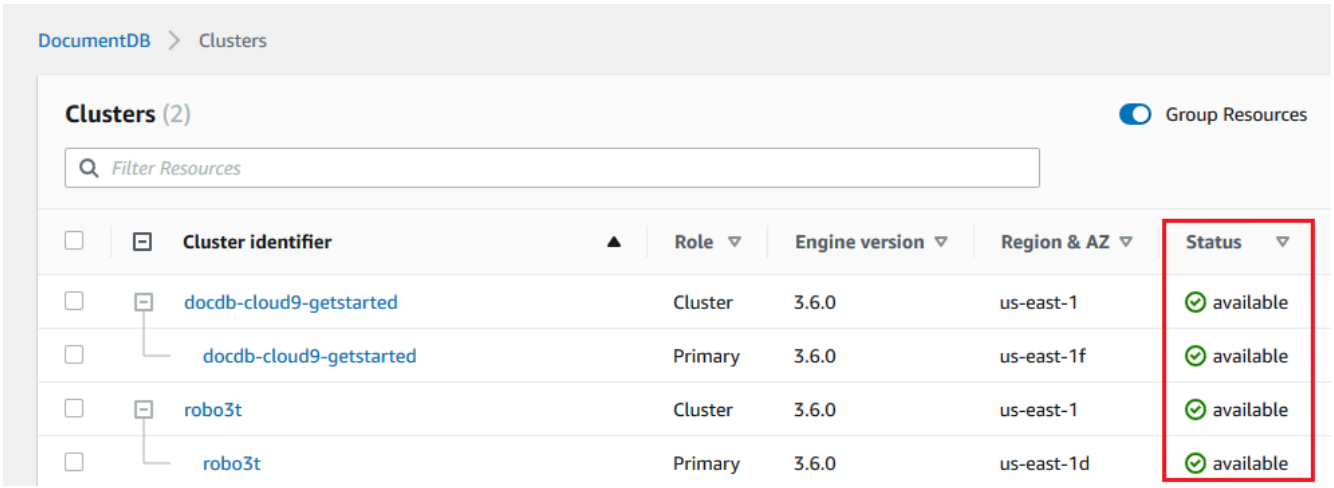
を使用してクラスターのステータス AWS Management Console を判断する場合は、次の手順を使用します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。

> **Note**
>
> Clusters ナビゲーションボックスで、[クラスター識別子] 列にはクラスターとインスタンスの両方が表示されることに注意してください。インスタンスは、以下のスクリーンショットのように、クラスターの下に表示されます。



3. 興味のあるインスタンスの名前を探します。次に、インスタンスのステータスを見つけるには、次に示すように [ステータス] 列まで行全体を読みます。



Using the AWS CLI

を使用してクラスターのステータス AWS CLI を判断する場合は、 `describe-db-instances`オペレーションを使用します。次のコードでは、インスタンス `sample-cluster-instance-01` のステータスを確認します。

Linux、macOS、Unix の場合:

```
aws docdb describe-db-instances \
          --db-instance-identifier sample-cluster-instance-01  \
          --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

Windows の場合:

```
aws docdb describe-db-instances ^
          --db-instance-identifier sample-cluster-instance-01  ^
          --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

このオペレーションによる出力は、次のようになります。

```
[
          [
              "sample-cluster-instance-01",
              "available"
          ]
      ]
```

インスタンスのヘルスステータス値

DB インスタンスに可能なステータス値を以下の表に示します。のクラスターテーブルにあるインスタンスヘルス列には AWS Management Console、データの保存、管理、取得を担当するコンポーネントであるデータベースエンジンが正常に動作しているかどうかが表示されます。この列には、CloudWatch で使用できる `EngineUptime` システムメトリックスに各インスタンスのヘルスステータスが表示されているかどうかも表示されます。

インスタンスのヘルスステータス	説明
正常	データベースエンジンは Amazon DocumentDB インスタンスで実行されています。
異常	データベースエンジンが実行されていないか、1 分以内に再起動しました。

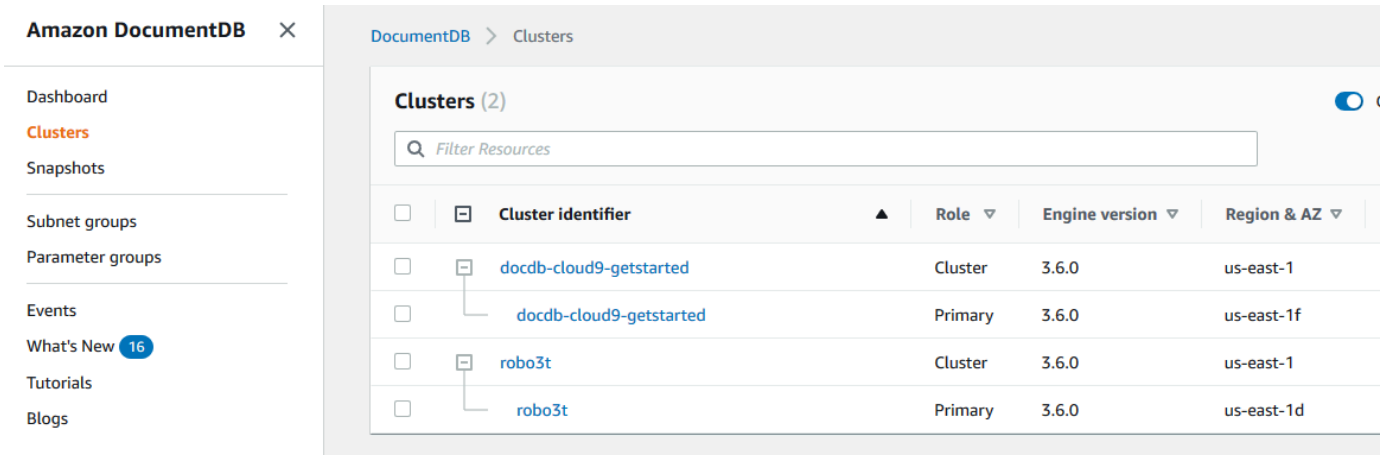
を使用したインスタンスのヘルスステータスのモニタリング AWS Management Console

を使用して AWS Management Console 、インスタンスのヘルスステータスをモニタリングします。

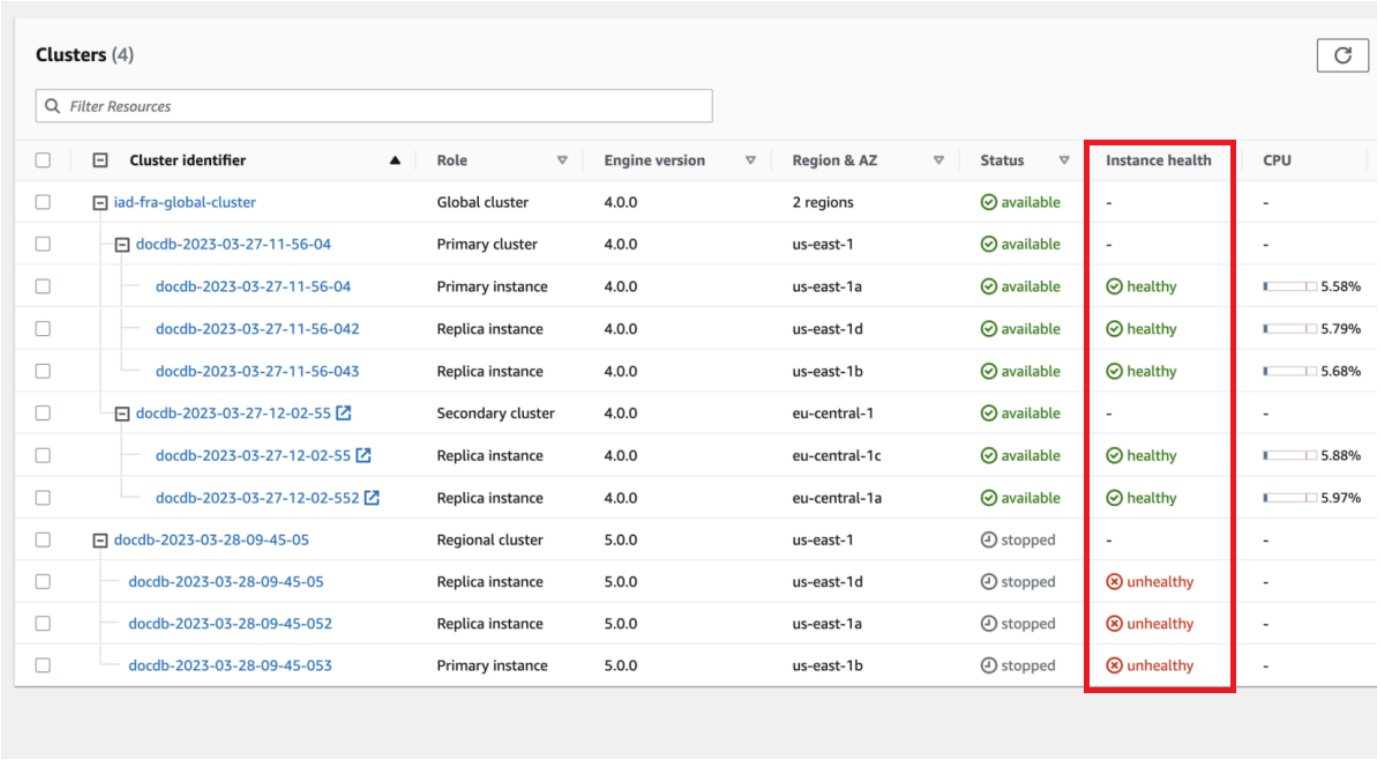
を使用する場合は AWS Management Console、次の手順を使用してインスタンスのヘルスステータスを理解します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで [Clusters] (クラスター) を選択してください。

> **Note**
>
> [クラスター] ナビゲーションボックスの [クラスター識別子] 列にはクラスターとインスタンスの両方が表示されます。インスタンスは、以下のスクリーンショットのように、クラスターの下に表示されます。



3. 興味のあるインスタンスの名前を探します。次に、インスタンスのステータスを見つけるには、次に示すように [インスタンスヘルス] 列まで行全体を読みます。



> **Note**
>
> インスタンスヘルスステータスのポーリングは 60 秒ごとに発生し、CloudWatch `EngineUptime` システムメトリックスに基づいています。[インスタンスヘルス] 列の値は自動的に更新されます。

Amazon DocumentDB に関する推奨事項の表示

Amazon DocumentDB では、インスタンスやクラスターなどのデータベースリソースについての推奨事項が自動で表示されます。これらの推奨事項では、クラスターとインスタンスの設定を分析して、ベストプラクティスガイダンスを提供します。

これらの推奨事項の例については、以下を参照してください。

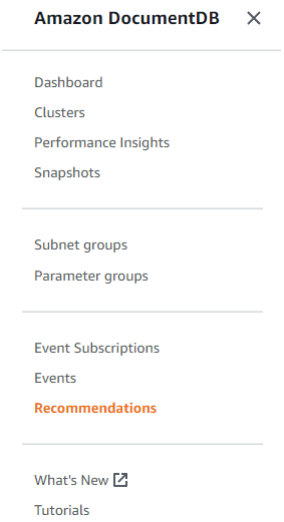
タイプ	説明	推奨事項	追加情報
1 個の インスタンス	クラスターにはインスタンスが 1 つだけ含まれています。	パフォーマンスと可用性: 同じ DB インスタンスクラスを持つ	Amazon DocumentDB 高可用性とレプリケーション

タイプ	説明	推奨事項	追加情報
		別の DB インスタンスを別の可用性ゾーンに追加することをお勧めします。	

Amazon DocumentDB は、リソースが作成または変更されると、リソースの推奨事項を生成します。また、Amazon DocumentDBは定期的にリソースをスキャンして、推奨事項を生成します。

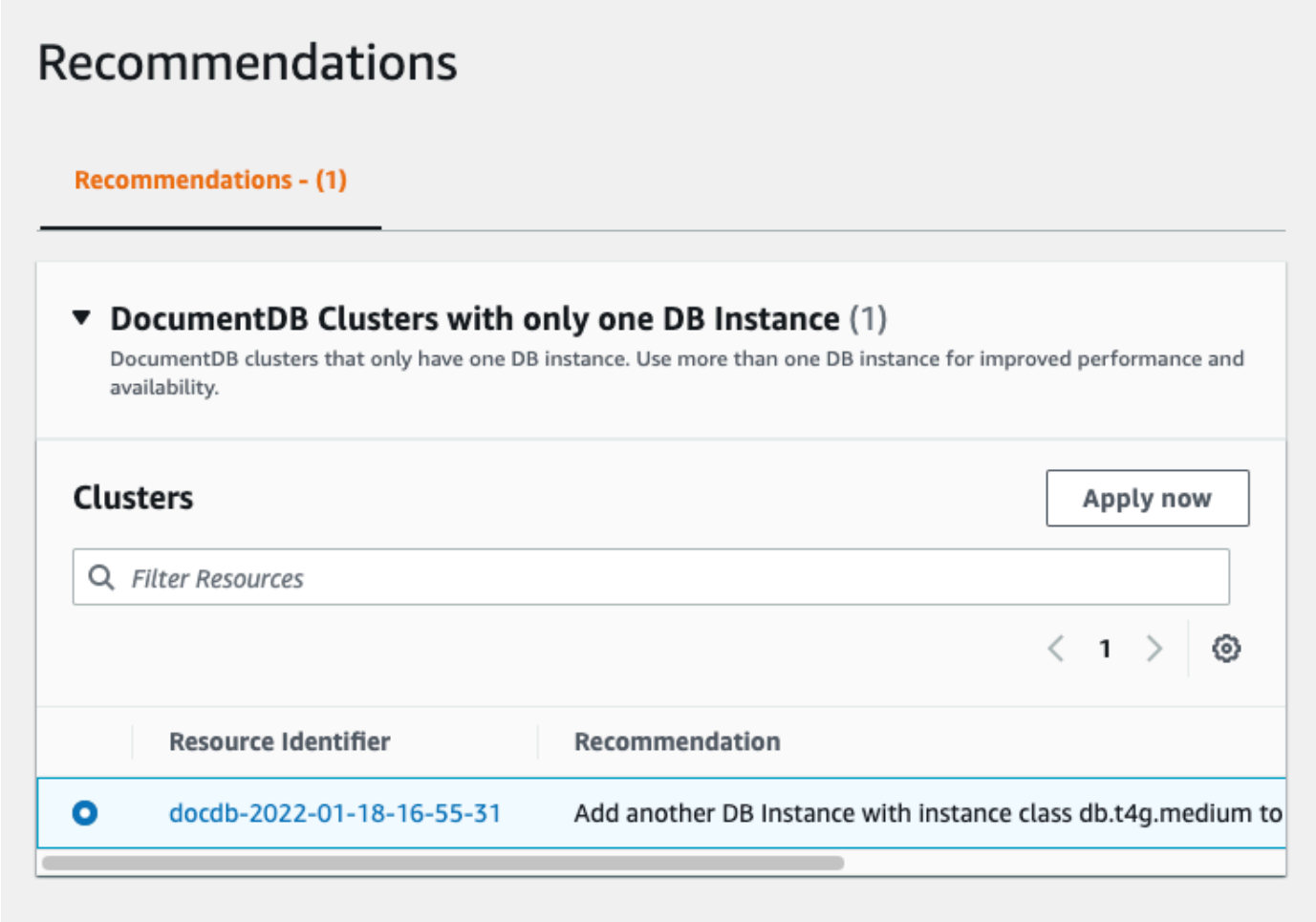
Amazon DocumentDB の推奨事項を確認してアクションを実行するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[推奨事項] を選択します。



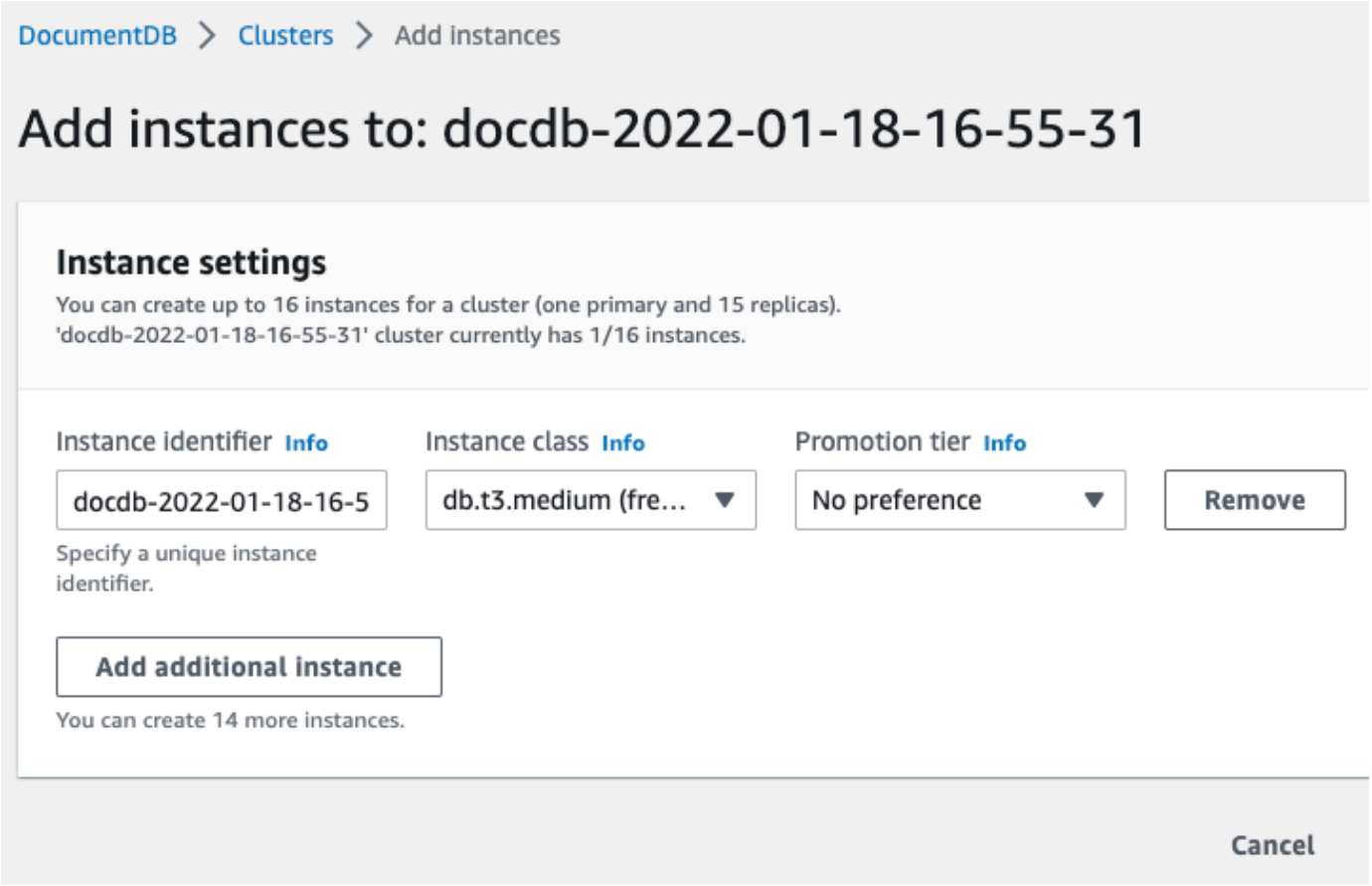
3. [推奨事項] ダイアログで、関心のあるセクションを展開し、推奨タスクを選択します。

   以下の例では、推奨タスクはインスタンスが 1 つしかない Amazon DocumentDB クラスターに適用されます。パフォーマンスと可用性を改善するために、別のインスタンスを追加することをお勧めします。



4. [今すぐ適用] をクリックします。

   この例では、[インスタンスの追加] ダイアログが表示されます。



5. 新しいインスタンスの設定を変更し、[作成] をクリックします。

Amazon DocumentDB イベントサブスクリプションを使用する

Amazon DocumentDB では、Amazon DocumentDB のイベントが発生したときに、Amazon Simple Notification Service (Amazon SNS) を使用して通知を送信します。これらの通知は、E メール AWS リージョン、テキストメッセージ、HTTP エンドポイントへの呼び出しなど、 の Amazon SNS でサポートされている任意の形式で送信できます。

Amazon DocumentDB は、サブスクライブ可能なカテゴリにイベントをグループ分けします。これにより、そのカテゴリのイベントが発生すると、通知を受け取ることができます。インスタンス、クラスター、スナップショット、クラスタースナップショット、または パラメータグループのイベントカテゴリをサブスクライブできます。例えば、特定のインスタンスのバックアップカテゴリをサブスクライブした場合、インスタンスに影響するバックアップ関連のイベントが発生するたびに通知が送信されます。また、イベントサブスクリプションが変更されても、通知を受け取ります。

イベントは、クラスターレベルとインスタンスレベルの両方で発生するため、クラスターまたはインスタンスをサブスクライブするとイベントを受信できます。

イベントサブスクリプションは、サブスクリプションを作成するときに指定したアドレスに送信されます。いくつかの異なるサブスクリプションを作成することもできます。例えば、すべてのイベント通知を受信するサブスクリプションと、本番稼働用のインスタンスに関する重要なイベントのみを含むサブスクリプションを作成できます。サブスクリプションを削除せずに通知を簡単にオフにできます。これを行うには、Amazon DocumentDB コンソールで [Enabled (有効)] ラジオボタンを [No (なし)]に設定します。

> **Important**
>
> Amazon DocumentDB はイベントストリームのイベントの順番を保証しません。イベントの順番は変わる場合があります。

Amazon DocumentDB では、Amazon SNS トピックの Amazon リソースネーム (ARN) を使用して、各サブスクリプションを識別します。Amazon DocumentDB コンソールでは、サブスクリプションの作成時に ARN が作成されます。

Amazon DocumentDB イベントサブスクリプションの請求は、Amazon SNS を通じて行われます。使用したイベント通知に対して、Amazon SNS 料金が適用されます。詳細については、「Amazon Simple Notification Service の料金」を参照してください。Amazon SNS の料金以外では、Amazon DocumentDB はイベントサブスクリプションの請求は行いません。

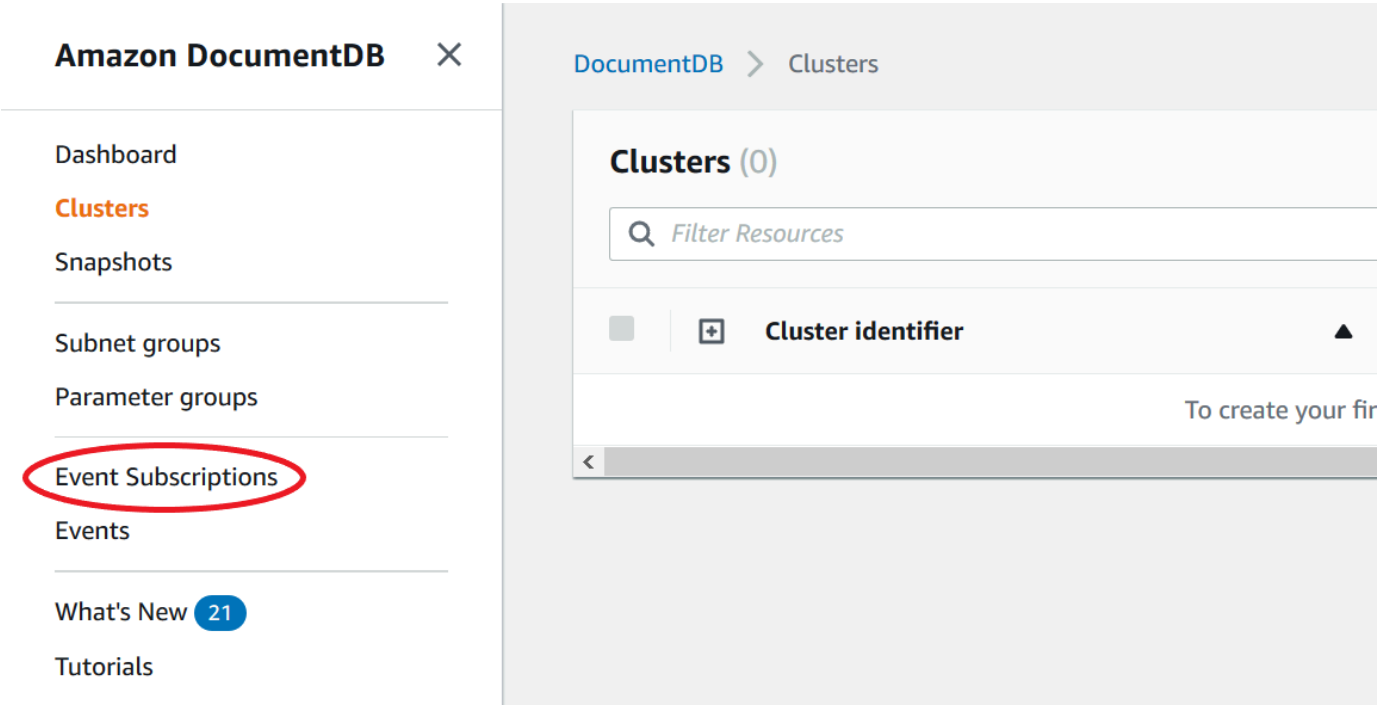
トピック

- Amazon DocumentDB イベントにサブスクライブする
- Amazon DocumentDB イベント通知サブスクリプションを管理する
- Amazon DocumentDB イベントのカテゴリとイベントメッセージ

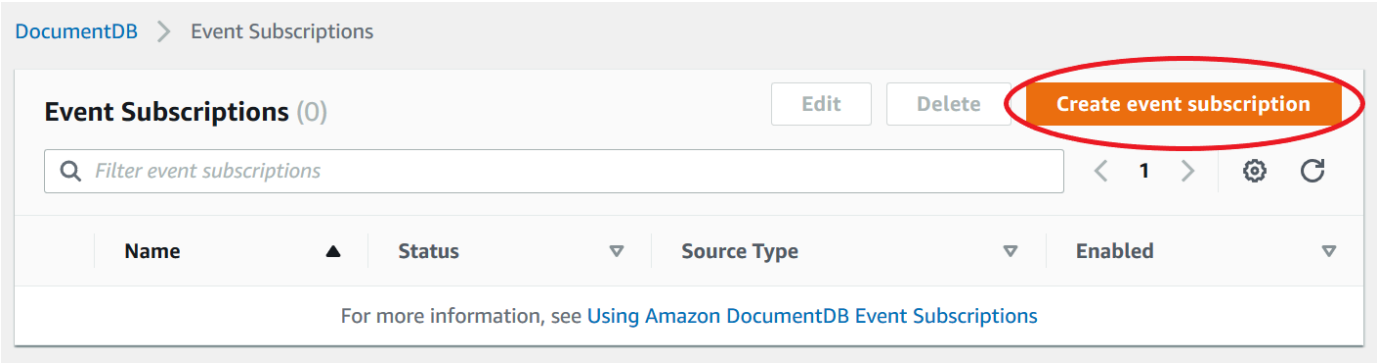
Amazon DocumentDB イベントにサブスクライブする

Amazon DocumentDB コンソールを使用して、次のようにイベントサブスクリプションをサブスクライブできます。

1. AWS Management Console で にサインインしますhttps://console.aws.amazon.com/docdb。
2. ナビゲーションペインで、[Event subscriptions] (イベントサブスクリプション) を選択します。

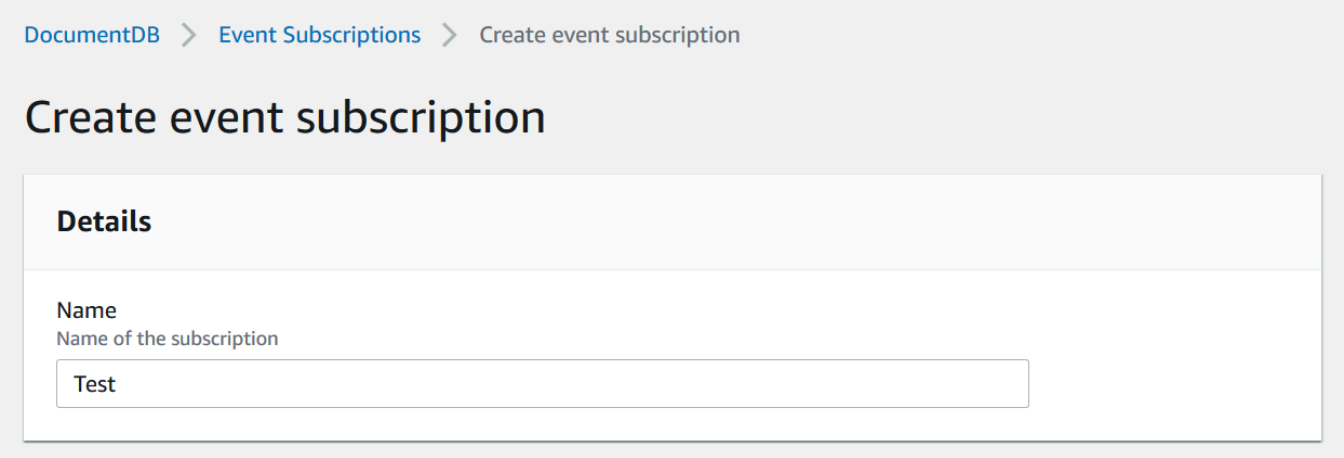


3. [イベントサブスクリプション] ページで、[イベントサブスクリプションの作成] を選択します。

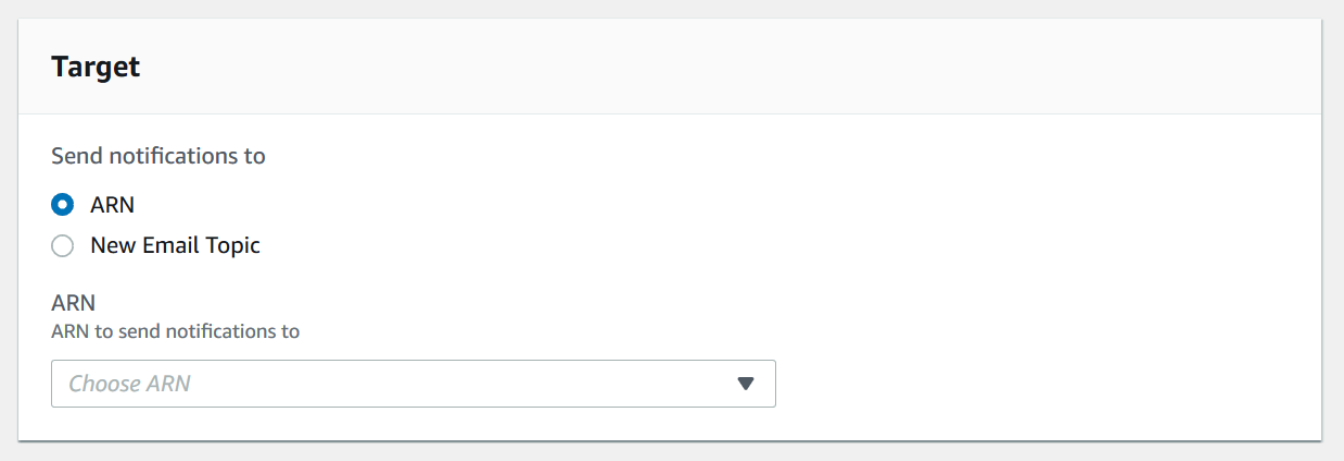


4. [イベントサブスクリプションの作成] ダイアログボックスで、次の操作を行います。

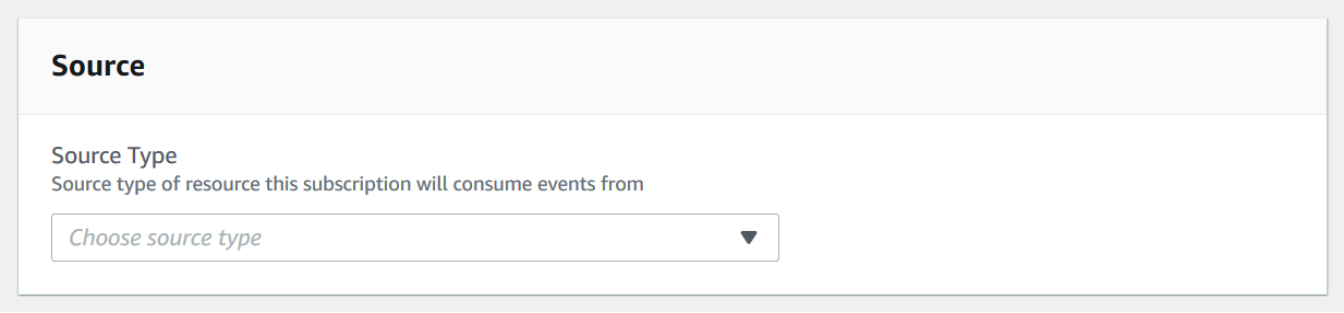
   - [名前] に、イベント通知サブスクリプションの名前を入力します。



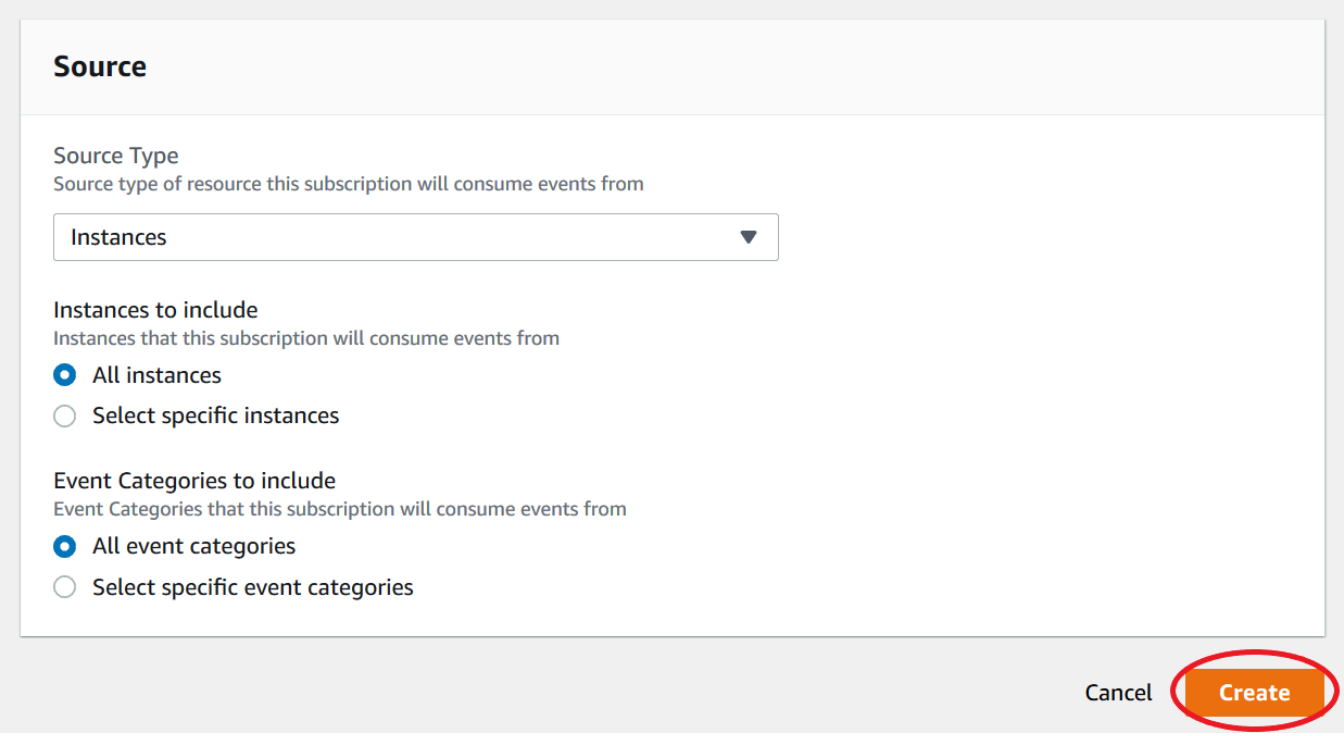
- [ターゲット]で、通知を送信する場所を選択します。既存の ARN を選択できます。または [新しい電子メールトピック] を選択して、トピックの名前と受取人のリストを入力します。



- [ソース] で、ソースタイプを選択します。選択したソースタイプに応じて、イベント通知を受け取る対象のイベントカテゴリとソースを選択します。



- [Create] (作成) を選択します。

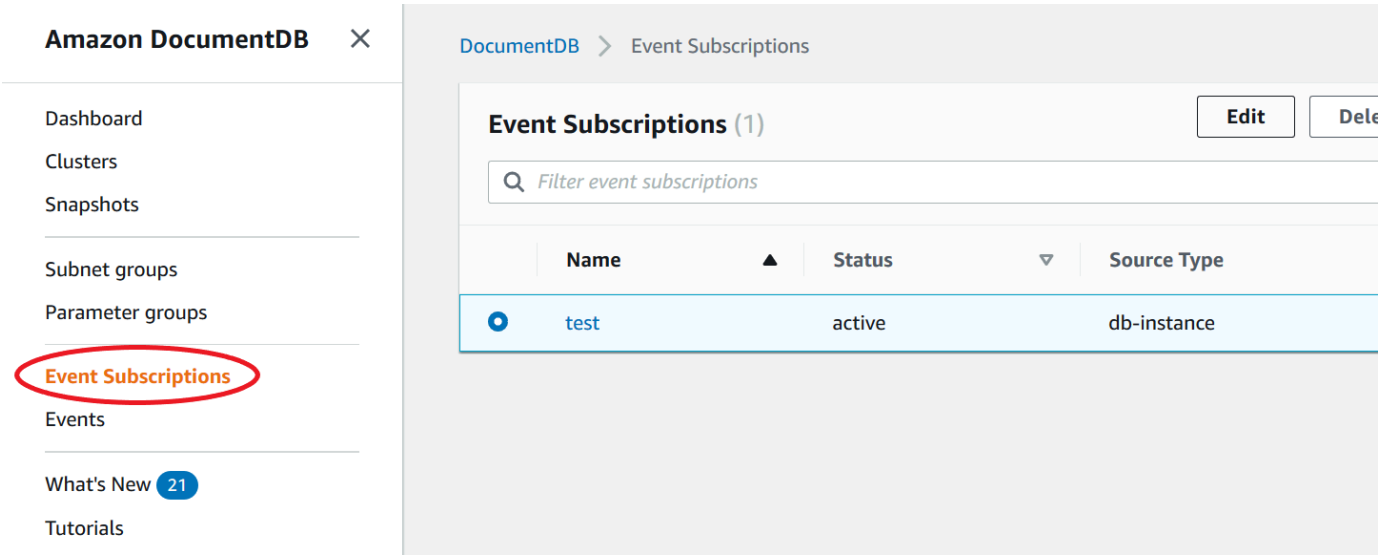


Amazon DocumentDB イベント通知サブスクリプションを管理する

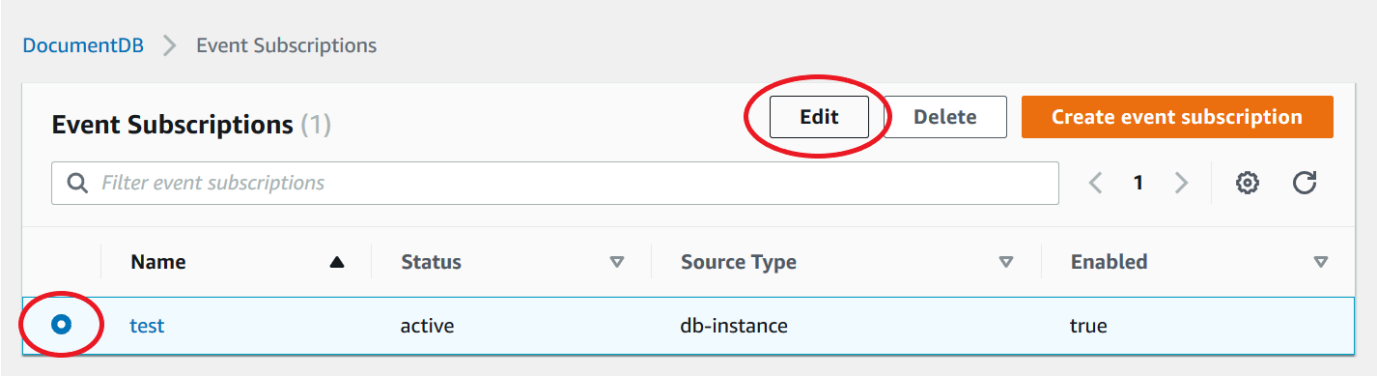
Amazon DocumentDB コンソールのナビゲーションペインで [イベントサブスクリプション] を選択すると、サブスクリプションカテゴリと現在のサブスクリプションのリストを表示できます。特定のサブスクリプションを変更または削除することもできます。

現在の Amazon DocumentDB イベント通知サブスクリプションを変更するには

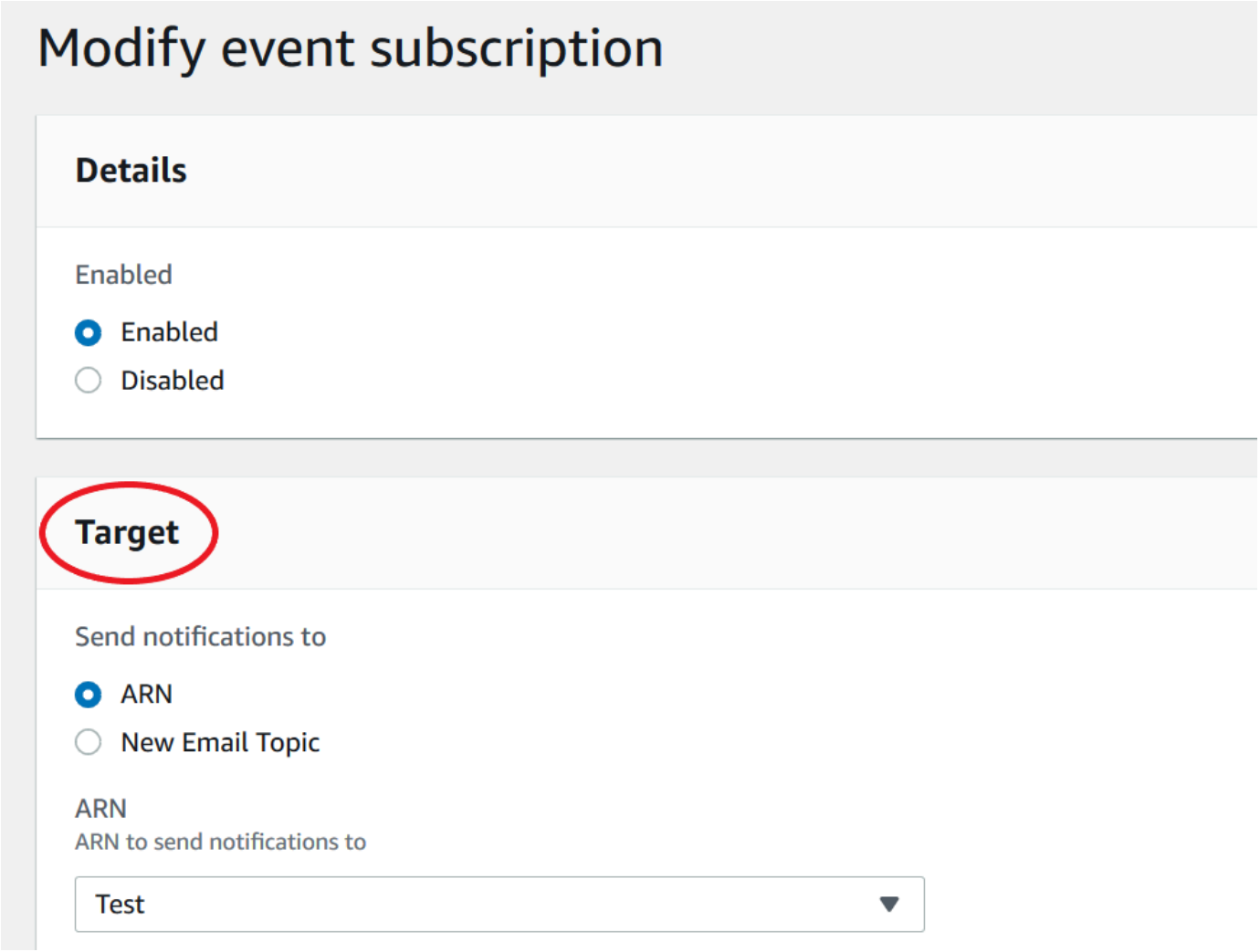
1. AWS Management Console で にサインインしますhttps://console.aws.amazon.com/docdb。
2. ナビゲーションペインで、[イベントサブスクリプション] を選択します。[イベントサブスクリプション] ペインにイベント通知サブスクリプションが一覧表示されます。



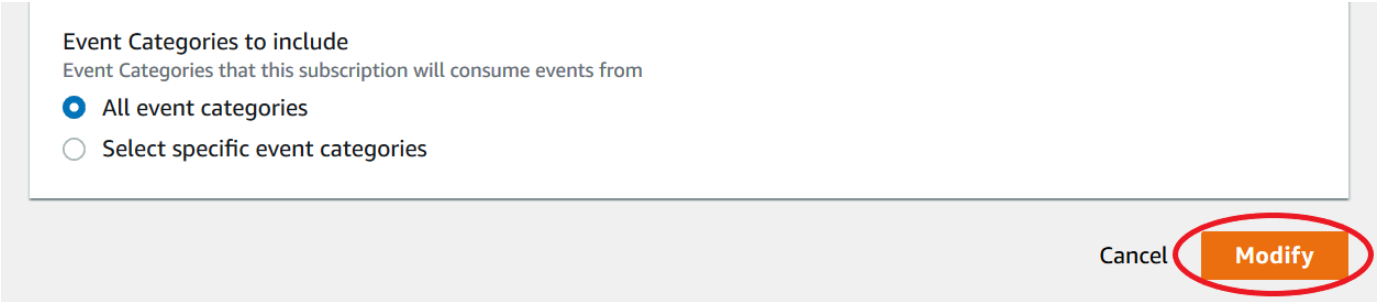
3. [イベントサブスクリプション] ペインで、変更するサブスクリプションを選択し、[編集] をクリックします。



4. [ターゲット] セクションまたは [ソース] セクションのいずれかでサブスクリプションを変更します。[ソース] セクションで選択または選択解除することで、ソース識別子を追加または削除できます。



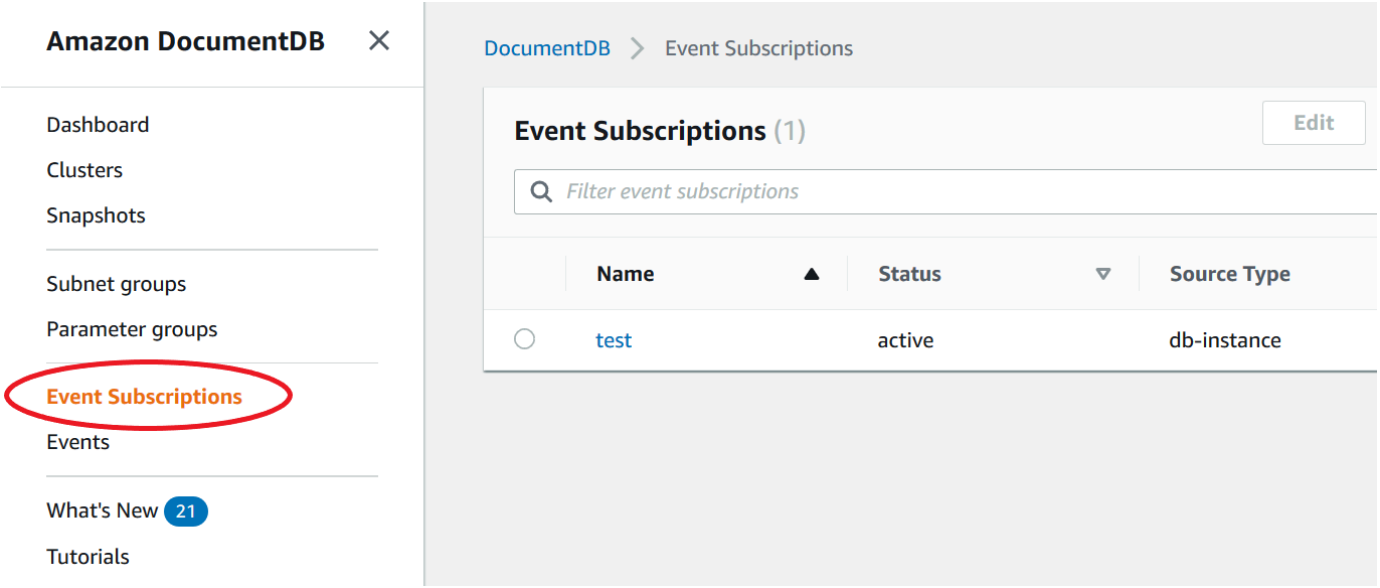
5. Modify を選択します。Amazon DocumentDB コンソールでは、サブスクリプションが変更されることが示されます。



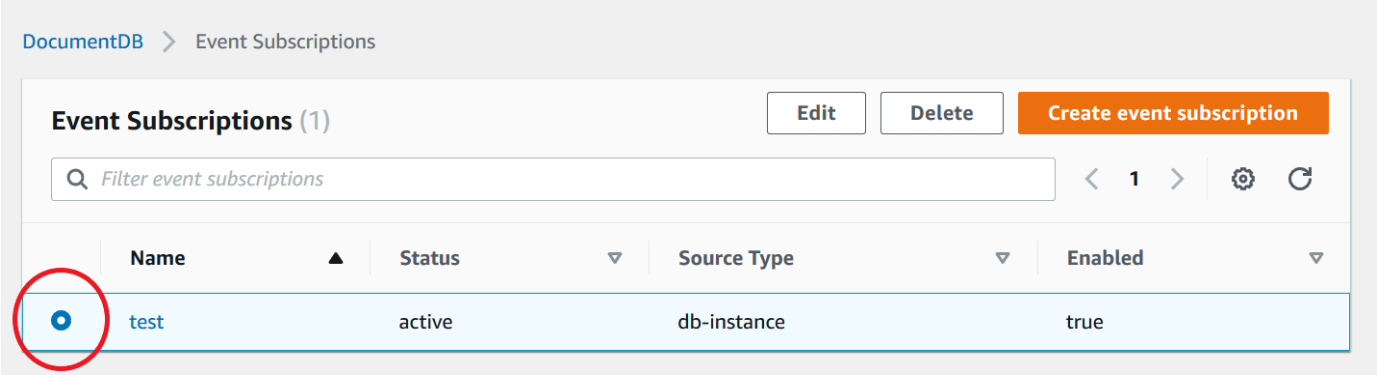
Amazon DocumentDB イベント通知サブスクリプションの削除

不要になったサブスクリプションは削除できます。トピックへのすべてのサブスクライバは、サブスクリプションにより指定されたイベント通知を受け取らなくなります。

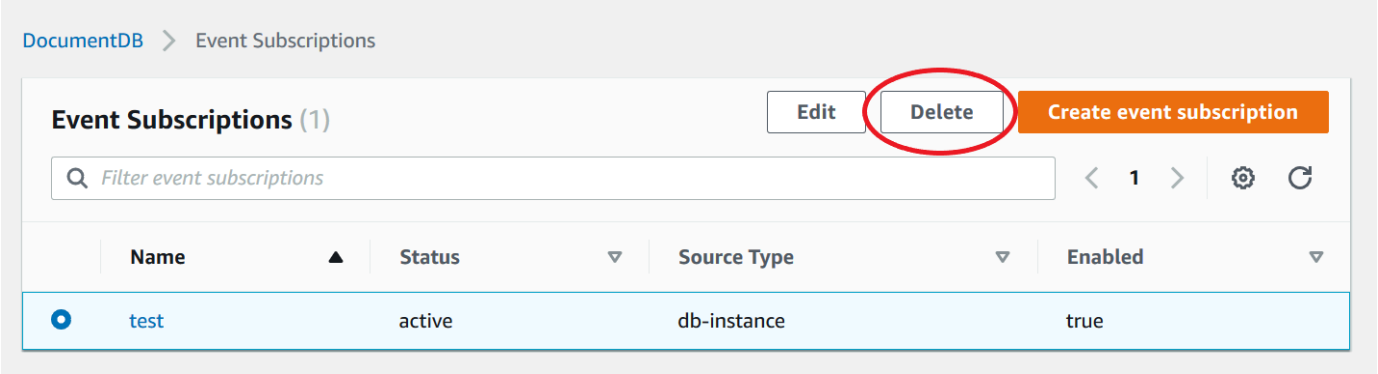
1. AWS Management Console で にサインインしますhttps://console.aws.amazon.com/docdb。
2. ナビゲーションペインで、[Event subscriptions] (イベントサブスクリプション) を選択します。



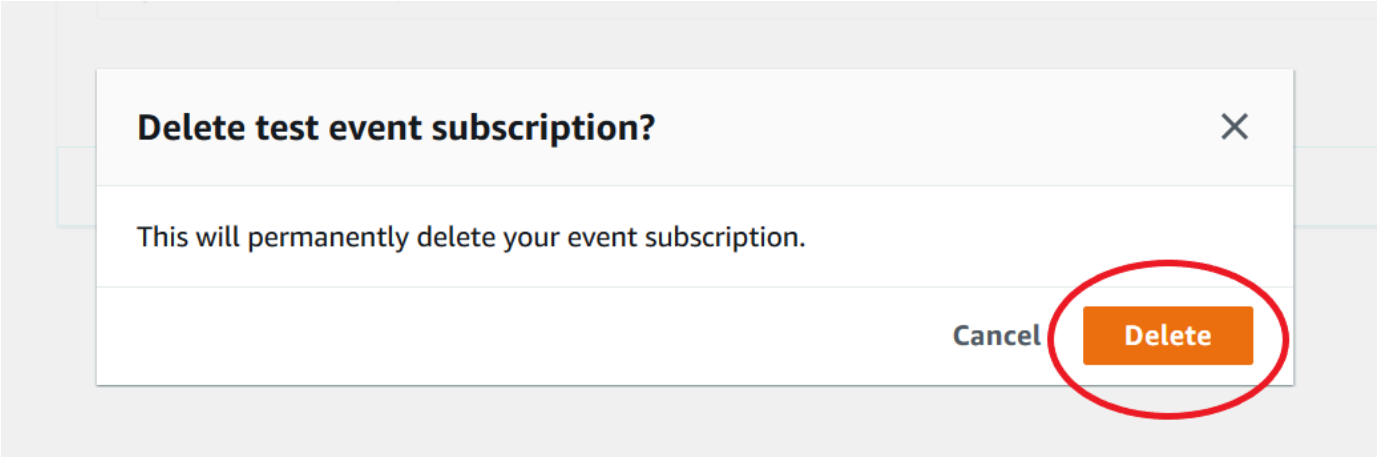
3. [イベントサブスクリプション] ペインで、削除するサブスクリプションを選択します。



4. [削除] を選択します。



5. この通知を完全に削除するかどうかを尋ねるポップアップウィンドウが表示されます。[削除] を選択します。



Amazon DocumentDB イベントのカテゴリとイベントメッセージ

Amazon DocumentDB では、 コンソールを使用してサブスクライブできるカテゴリ内で多数のイベントが生成されます。各カテゴリは、インスタンス、クラスター、スナップショット、またはパラメータグループであるソースタイプに適用されます。

> **Note**
>
> Amazon DocumentDB では既存の Amazon RDS イベント定義と ID が使用されます。

インスタンスから発生する Amazon DocumentDB イベント

カテゴリ	説明
高可用性	インスタンスが再起動しました。
可用性	インスタンスがシャットダウンしました。
設定変更	変更をインスタンスクラスに適用します。
設定変更	変更をインスタンスクラスに適用しました。
設定変更	プライマリ認証情報をリセットします。
作成	インスタンスが作成されました。

カテゴリ	説明
削除	インスタンスが削除されました
失敗	インスタンスは、互換性のない設定または基本的なストレージの問題により失敗しました。インスタンスのポイントインタイムの復元を開始します。
通知	インスタンスが停止しました。
通知	インスタンスが起動しました。
通知	停止中の最大許容時間を超えたため、インスタンスが起動されています。
リカバリ	インスタンスの復旧が開始しました。復旧時間は、復旧するデータの量に応じて変わります。
リカバリ	インスタンスの復旧が完了しました。
セキュリティパッチ	ご使用のインスタンスに対するオペレーティングシステムアップデートがあります。アップデートの適用方法については、「Amazon DocumentDB のメンテナンス」を参照してください。

クラスターから発生する Amazon DocumentDB イベント

カテゴリ	説明
作成	クラスターが作成されました
削除	クラスターが削除されました。
フェイルオーバー	以前のプライマリを再度プロモートします。

カテゴリ	説明
フェイルオーバー	インスタンスへのフェイルオーバーが完了しました。
フェイルオーバー	DB インスタンスへのフェイルオーバーを開始しました: %s
フェイルオーバー	DB インスタンスへの同じ AZ フェイルオーバーを開始しました: %s
フェイルオーバー	DB インスタンスへのクロスAZ フェイルオーバーを開始しました: %s
メンテナンス	クラスターにパッチが適用されました。
メンテナンス	データベースクラスターはアップグレードできない状態です: %s
通知	クラスターが停止しました。
通知	クラスターが開始しました。
通知	クラスターの停止が失敗しました。
通知	停止中の最大許容時間を超えたため、クラスターが開始されています。
通知	クラスターの名前を %s から %s に変更しました。

クラススナップショットから発生する Amazon DocumentDB イベント

次の表は、Amazon DocumentDB クラスターのスナップショットがソースタイプである場合のイベントカテゴリとイベントのリストを示しています。

カテゴリ	説明
バックアップ	手動クラススナップショットの作成

カテゴリ	説明
バックアップ	手動クラスタースナップショットが作成されました。
バックアップ	自動クラスタースナップショットを作成しています。
バックアップ	自動クラスタースナップショットが作成されました。

パラメータグループから発生する Amazon DocumentDB イベント

次の表は、パラメータグループがソースタイプである場合のイベントカテゴリとイベントのリストを示します。

カテゴリ	説明
設定変更	適用メソッド %s でパラメータ %s を %s に更新しました

Amazon DocumentDB と CloudWatch のモニタリング

Amazon DocumentDB (MongoDB 互換) は Amazon CloudWatch と統合することで、クラスターのオペレーションメトリクスを収集および分析できます。これらのメトリクスは、CloudWatch コンソール、Amazon DocumentDB コンソール、 AWS Command Line Interface （ AWS CLI ）、または CloudWatch API を使用してモニタリングできます。

CloudWatch では、メトリクスが指定したしきい値を超えた場合に通知を受け取ることができるように、アラームを設定することもできます。超過が発生した場合に修正作業を行うことができるように、Amazon CloudWatch Events を設定することもできます。CloudWatch とアラームの使用方法の詳細については、Amazon CloudWatch のドキュメント を参照してください。

トピック

- Amazon DocumentDB のメトリクス
- CloudWatch データの表示

- Amazon DocumentDB ディメンション
- Opcounter メトリクスのモニタリング
- データベース接続のモニタリング

Amazon DocumentDB のメトリクス

Amazon DocumentDB クラスターとインスタンスの状態とパフォーマンスをモニタリングするために、 Amazon DocumentDB コンソールで、次のメトリクスを表示できます。

> **Note**
>
> 以下の表のメトリクスは、インスタンスベースのクラスターと Elastic クラスターの両方に適用されます。

トピック

- リソース使用率メトリクス
- レイテンシーメトリクス
- NVMe-backed インスタンスメトリクス
- オペレーションメトリクス
- スループットメトリクス
- システムメトリクス
- T3 インスタンスメトリクス

リソース使用率メトリクス

メトリクス	説明	
`BackupRetentionPeriodStorageUsed`	Amazon DocumentDB の保持期間内のpoint-in-time復元機能をサポートするために使用されるバックアップストレージの合計量をバイト単位で表します。`TotalBack`	

メトリクス	説明	
	upStorageBilled メトリクスによって報告される合計に含まれます。各 Amazon DocumentDB クラスターに対して個別に計算されます。	
ChangeStreamLogSize	変更ストリームログを保存するためにクラスターが使用するストレージの容量 (MB)。この値は、クラスターの合計ストレージのサブセット (VolumeBytesUsed) であり、クラスターのコストに影響します。ストレージの料金については、Amazon DocumentDB 製品ページ を参照してください。変更ストリームのログサイズは、クラスターで発生している変更の量と、変更ストリームの長期保存期間の関数です。変更ストリームの詳細については、[Amazon DocumentDB を用いた変更ストリームの使用] を参照してください。	
CPUUtilization	インスタンスによって使用される CPU のパーセント。	
DatabaseConnections	1 分間隔で実行されたインスタンスで開いている接続 (アクティブおよびアイドル) の数。	

メトリクス	説明
DatabaseConnectionsMax	1 分間にインスタンスで開いているデータベース接続 (アクティブおよびアイドル) の最大数。
DatabaseConnectionsLimit	任意の時点でインスタンスで許可される同時データベース接続 (アクティブおよびアイドル) の最大数。
DatabaseCursors	1 分間隔で取得されたインスタンスで開いているカーソルの数。
DatabaseCursorsMax	1 分間にインスタンスで開いているカーソルの最大数。
DatabaseCursorsLimit	任意の時点でインスタンスで許可されるカーソルの最大数。
DatabaseCursorsTimedOut	1 分間にタイムアウトしたカーソルの数。
FreeableMemory	使用可能な RAM の容量 (バイト単位)。

メトリクス	説明
FreeLocalStorage	このメトリクスでは、各インスタンスの一時テーブルとログで使用できるストレージの量が報告されます。この値は、インスタンスクラスによって異なります。インスタンスに対してより大きなインスタンスクラスを選択することで、インスタンス用の空きストレージ容量を増やすことができます。
LowMemThrottleQueueDepth	1 分間隔で取得される使用可能なメモリが少ないためにスロットリングされたリクエストのキュー深度。
LowMemThrottleMaxQueueDepth	1 分間に使用可能なメモリが少ないためにスロットリングされたリクエストの最大キュー深度。
LowMemNumOperationsThrottled	1 分間に使用可能なメモリが少ないためにスロットリングされたリクエストの数。

メトリクス	説明	
SnapshotStorageUsed	バックアップ保持期間外に特定の Amazon DocumentDB クラスターのすべてのスナップショットによって消費されたバックアップストレージのバイト単位の合計量。TotalBackupStorageBilled メトリクスによって報告される合計に含まれます。各 Amazon DocumentDB クラスターに対して個別に計算されます。	
SwapUsage	インスタンスで使用するスワップ領域の量。	
TotalBackupStorageBilled	特定の Amazon DocumentDB クラスターに対して請求されるバックアップストレージのバイト単位の合計量。BackupRetentionPeriodStorageUsed メトリクスおよび SnapshotStorageUsed メトリクスによって測定されるバックアップストレージが含まれます。各 Amazon DocumentDB クラスターに対して個別に計算されます。	
TransactionsOpen	1 分間隔で実行されたインスタンスで開いているトランザクションの数。	

メトリクス	説明
TransactionsOpenMax	1 分間にインスタンスで開いているトランザクションの最大数。
TransactionsOpenLimit	任意の時点でインスタンスで許可される同時トランザクションの最大数。
VolumeBytesUsed	クラスターで使用されたストレージ容量 (バイト単位)。この値は、クラスターのコストに影響します。料金情報については、Amazon DocumentDB 製品ページ を参照してください。

レイテンシーメトリクス

メトリクス	説明
DBClusterReplicaLagMaximum	クラスター内のプライマリインスタンスと Amazon DocumentDB インスタンス間の最大遅延時間 (ミリ秒単位)。
DBClusterReplicaLagMinimum	クラスター内のプライマリインスタンスと各レプリカインスタンス間の最小遅延時間 (ミリ秒単位)。
DBInstanceReplicaLag	プライマリインスタンスからレプリカインスタンスにアップデートをレプリケートする際の遅延時間 (ミリ秒単位)。

メトリクス	説明
ReadLatency	1 回のディスク I/O オペレーションにかかる平均時間。
WriteLatency	1 回のディスク I/O オペレーションにかかる平均時間 (ミリ秒単位)。

NVMe-backed インスタンスメトリクス

メトリクス	説明
NVMeStorageCacheHitRatio	階層型キャッシュによって処理されるリクエストの割合。
FreeNVMeStorage	使用できる Ephemeral NVMe ストレージの容量。
ReadIOPSNVMeStorage	Ephemeral NVMe ストレージへのディスク読み取り I/O オペレーションの平均数。
ReadLatencyNVMeStorage	Ephemeral NVMe ストレージのディスク読み取り I/O オペレーションごとにかかる平均時間。
ReadThroughputNVMeStorage	Ephemeral NVMe ストレージ用にディスクから読み取られた 1 秒あたりの平均バイト数。
WriteIOPSNVMeStorage	Ephemeral NVMe ストレージへのディスク書き込み I/O オペレーションの平均数。

メトリクス	説明
WriteLatencyNVMeStorage	Ephemeral NVMe ストレージのディスク書き込み I/O オペレーションごとにかかる平均時間。
WriteThroughputNVMeStorage	Ephemeral NVMe ストレージ用にディスクに書き込まれる 1 秒あたりの平均バイト数。

オペレーションメトリクス

メトリクス	説明
DocumentsDeleted	1 分間に削除されたドキュメントの数。
DocumentsInserted	1 分間に挿入されたドキュメントの数。
DocumentsReturned	1 分間に返されたドキュメントの数。
DocumentsUpdated	1 分間に更新されたドキュメントの数。
OpcountersCommand	1 分間に発行されたコマンドの数。
OpcountersDelete	1 分間に発行された削除オペレーションの数。
OpcountersGetmore	1 分間に発行された getmore の数。
OpcountersInsert	1 分間に発行された挿入オペレーションの数。

メトリクス	説明
OpcountersQuery	1 分間に発行されたクエリの数。
OpcountersUpdate	1 分間に発行された更新オペレーションの数。
TransactionsStarted	1 分間にインスタンスで開始されたトランザクションの数。
TransactionsCommitted	1 分間にインスタンスでコミットされたトランザクションの数。
TransactionsAborted	1 分間にインスタンスで中止されたトランザクションの数。
TTLDeletedDocuments	1 分間に TTLMonitor によって削除されたドキュメントの数。

スループットメトリクス

メトリクス	説明
NetworkReceiveThroughput	クラスター内の各インスタンスが各クライアントから受信したネットワークスループットの量 (バイト/秒単位)。クラスターとクラスターボリューム内のインスタンス間のネットワークトラフィックは、このスループットに含まれません。

メトリクス	説明	
NetworkThroughput	Amazon DocumentDB クラスター内の各インスタンスがクライアントで送受信したネットワークスループットの量 (バイト/秒単位)。クラスターとクラスターボリューム内のインスタンス間のネットワークトラフィックは、このスループットに含まれません。	
NetworkTransmitThroughput	クラスター内の各インスタンスが各クライアントに対して送信したネットワークスループットの量 (バイト/秒単位)。クラスターとクラスターボリューム内のインスタンス間のネットワークトラフィックは、このスループットに含まれません。	
ReadIOPS	1 秒あたりのディスク読み取り I/O オペレーションの平均回数。Amazon DocumentDB は、読み取り IOPS と書き込み IOPS を個別に 1 分間隔でレポートします。	
ReadThroughput	1 秒あたりのディスクからの平均読み取りバイト数。	

メトリクス	説明	
StorageNetworkReceiveThroughput	Amazon DocumentDB クラスター内の各インスタンスがクラスターストレージボリュームから受信したネットワークスループットの量 (バイト/秒単位)。	
StorageNetworkTransmitThroughput	Amazon DocumentDB クラスター内の各インスタンスがクラスターストレージボリュームに対して送信したネットワークスループットの量 (バイト/秒単位)。	
StorageNetworkThroughput	Amazon DocumentDB クラスター内の各インスタンスがクラスターストレージボリュームとの間で送受信したネットワークスループットの量 (バイト/秒単位)。	

メトリクス	説明	
VolumeReadIOPs	課金読み取り I/O オペレーションの平均回数は 5 分間隔で報告されます。課金読み取りオペレーションはクラスターボリュームレベルで計算され、クラスター内のすべてのインスタンスから集計された後、5 分おきに報告されます。この値は読み取りオペレーションのメトリクスを 5 分間受け取ることによって計算されます。課金読み取りオペレーションのメトリクスを受け取って 300 秒で割ることで、1 秒あたりの課金読み取りオペレーションの回数を決定できます。 例えば、VolumeReadIOPs が 13,686 を返す場合、1 秒あたりの課金読み取りオペレーションは 45 (13,686 / 300 = 45.62) です。 バッファキャッシュに存在しないデータベースのページをリクエストするクエリの課金読み取りオペレーションが発生するため、ストレージからロードする必要があります。課金読み取りオペレーションはストレージからクエリの結果が読み取られるのと同様に急増することがありますが、	

メトリクス	説明	
	その後バッファキャッシュにロードされます。	

メトリクス	説明	
VolumeWriteIOPs	課金書き込み I/O オペレーションの平均回数は 5 分間隔で報告されます。課金書き込みオペレーションはクラスターボリュームレベルで計算され、クラスター内のすべてのインスタンスから集計された後、5 分おきに報告されます。この値は、5 分間にわたる書き込みオペレーションメトリクスの値を受け取ることによって計算されます。課金書き込みオペレーションメトリクスの値を受け取って 300 秒で割ることで、1 秒あたりの課金書き込みオペレーションの回数を決定できます。 例えば、VolumeWriteIOPs が 13,686 を返す場合、1 秒あたりの課金書き込みオペレーション回数は 45 (13,686 / 300 = 45.62) です。 なお、VolumeReadIOPs と VolumeWriteIOPs メトリクスは DocumentDB ストレージレイヤーによって計算され、プライマリインスタンスとレプリカインスタンスによって実行される IO が含まれます。データは 20 ~ 30 分ごとに集計され、5 分間隔でレポートされるため、その時間帯のメトリクスには同じデー	

メトリクス	説明	
	タポイントが出力されます。1 分間隔で挿入オペレーションと相関するメトリクスを探している場合は、インスタンスレベルの WriteIOPS メトリクスを使用できます。メトリクスは、Amazon DocumentDB プライマリインスタンスの モニタリングタブで使用できます。	
WriteIOPS	1 秒あたりのディスク書き込み I/O オペレーションの平均回数。クラスターレベルで使用すると、WriteIOPs はクラスター内のすべてのインスタンスで評価されます。読み取りおよび書き込み IOPS は個別に 1 分間隔で報告されます。	
WriteThroughput	1 秒あたりのディスクへの平均書き込みバイト数。	

システムメトリクス

メトリクス	説明	
BufferCacheHitRatio	バッファキャッシュから提供されたリクエストの割合 (パーセント)。	
DiskQueueDepth	ディスクへの書き込みまたはディスクからの読み取りを	

メトリクス	説明
	待っている I/O オペレーションの数。
EngineUptime	インスタンスの実行時間 (秒単位)。
IndexBufferCacheHitRatio	バッファキャッシュから提供されたインデックスリクエストの割合 (パーセント)。インデックス、コレクション、またはデータベースを削除すると、メトリクスのスパイクが 100% を超えることがあります。これは 60 秒後に自動的に修正されます。この制限は、今後のパッチアップデートで修正される予定です。

T3 インスタンスメトリクス

メトリクス	説明
CPUCreditUsage	測定期間に消費された CPU クレジットの数。
CPUCreditBalance	インスタンスが蓄積する CPU クレジット数。CPU がバーストし、CPU クレジットが獲得するよりも速い速度で使用される際に、このバランスは枯渇します。
CPUSurplusCreditBalance	CPUCreditBalance の値がゼロになった時に CPU パフォーマンスを保持するために消費さ

メトリクス	説明	
	れる、余剰 CPU クレジットの数。	
CPUSurplusCreditsCharged	24 時間で獲得できる CPU クレジットの最大数を越えた、追加料金が発生する分の余剰 CPU クレジットの数。詳細については、[CPU クレジットのモニタリング] を参照してください。	

CloudWatch データの表示

Amazon CloudWatch データは、CloudWatch コンソール、Amazon DocumentDB コンソール、AWS Command Line Interface（ AWS CLI ）、または CloudWatch API を使用して表示できます。

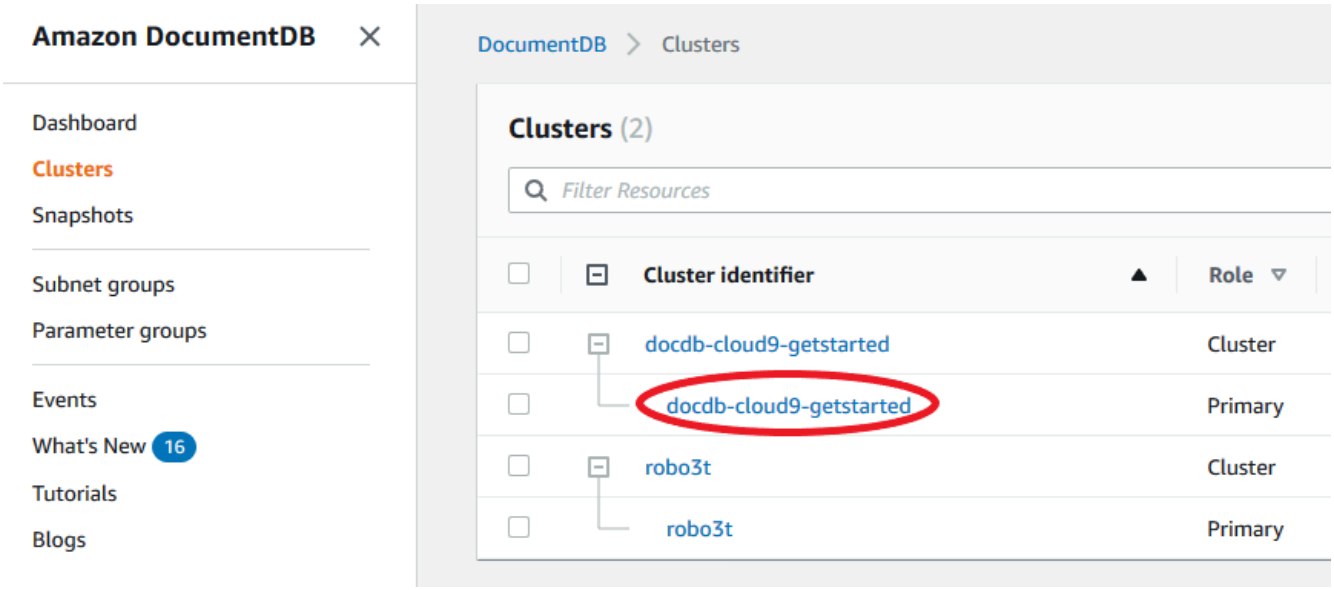
Using the AWS Management Console

Amazon DocumentDB マネジメントコンソール を使用して CloudWatch メトリクスを表示するには、次のステップを完了します。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB 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。

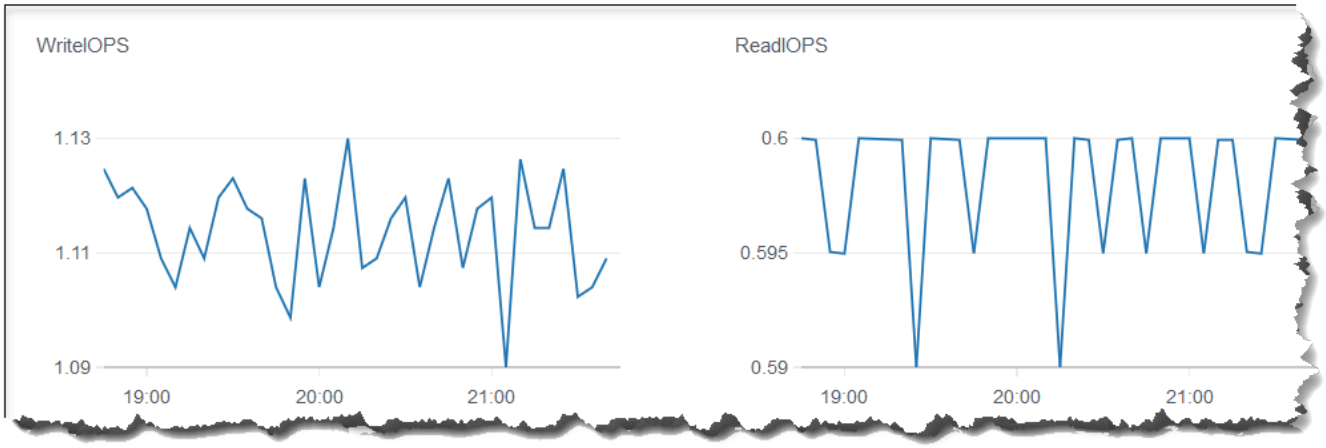
> **Tip**
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

3. クラスターナビゲーションボックスに、[Cluster Identifier]の列が表示されます。インスタンスは、以下のスクリーンショットのように、クラスターの下に表示されます。



4. インスタンスの一覧から、メトリクスを表示するインスタンスの名前を選択します。
5. インスタンスサマリーページで、モニタリング タブを選択すると、Amazon DocumentDB インスタンスのメトリクスをグラフィカルに表示することができます。各メトリクスに対してグラフを生成する必要があるため、[CloudWatch] グラフにデータが入力されるまでに数分かかる場合があります。

   次の図は、Amazon DocumentDB コンソールでの `WriteIOPS` と `ReadIOPS` の 2 つの CloudWatch メトリクスをグラフィカルに表示しています。



Using the CloudWatch Management Console

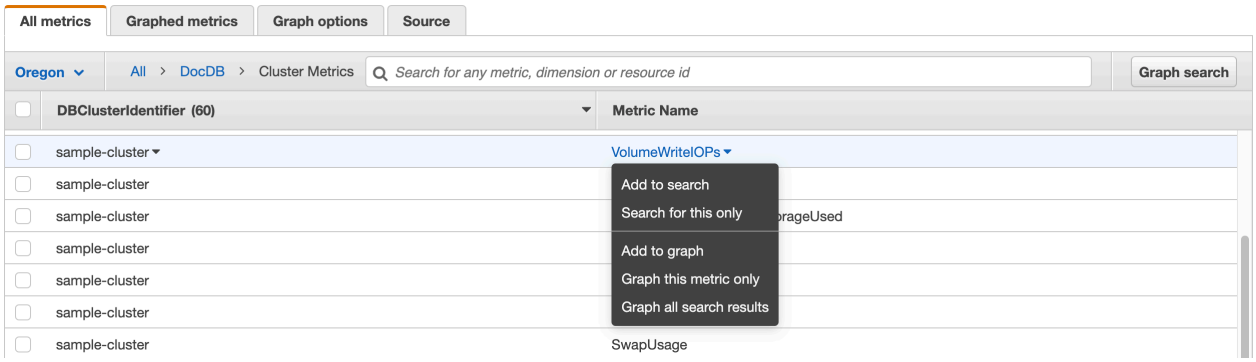
Amazon DocumentDB マネジメントコンソール を使用して、CloudWatch メトリクスを表示するには、次のステップを完了します。

1. にサインインし AWS Management Console、 で Amazon DocumentDB コンソールを開きますhttps://console.aws.amazon.com/cloudwatch。
2. ナビゲーションペインで [Metrics (メトリクス)] を選択してください。次に、サービス名のリストで、[DocDB] を選択します。
3. メトリクスのディメンション (例: クラスターメトリクス) を選択します。
4. 全てのメトリクス タブには、DocDB にあるディメンションの全てのメトリクスが表示されます。

    a. テーブルを並べ替えるには、列見出しを使用します。

    b. メトリクスをグラフ表示するには、メトリクスの横にあるチェックボックスを選択します。すべてのメトリクスを選択するには、テーブルの見出し行にあるチェックボックスを選択します。

    c. メトリクスでフィルタリングするには、メトリクス名にカーソルを合わせ、メトリクス名の横にあるドロップダウン矢印を選択します。次に、以下のイメージに示すように、検索に追加 を選択します。



Using the AWS CLI

Amazon DocumentDB の CloudWatch データを表示するには、以下のパラメータで CloudWatch `get-metric-statistics` オペレーションを使用します。

パラメータ

- **`--namespace`** — 必須。CloudWatch メトリクスを表示するサービス名前空間。Amazon DocumentDB の場合、これは `AWS/DocDB` でなければなりません。
- **`--metric-name`** — 必須。データを表示するメトリクスの名前。
- **`--start-time`** — 必須。最初に返すデータポイントを決定するタイムスタンプ。

指定された値は含まれます。つまり、結果には指定されたタイムスタンプのデータポイントが含まれます。タイムスタンプは ISO 8601 UTC 形式である必要があります (例: 2016-10-03T23:00:00Z)。

- **--end-time** — 必須。最後に返すデータポイントを決定するタイムスタンプ。

  指定された値は含まれます。つまり、結果には指定されたタイムスタンプのデータポイントが含まれます。タイムスタンプは ISO 8601 UTC 形式である必要があります (例: 2016-10-03T23:00:00Z)。

- **--period** — 必須。返されるデータポイントの詳細度 (秒)。通常の解像度のメトリクスについては、期間は最短 1 分 (60 秒) で、60 の倍数である必要があります。1 分未満の間隔で収集される高解像度メトリクスについては、期間は 1、5、10、30、60、または 60 の倍数にできます。

- **--dimensions** — オプション。メトリクスに複数のディメンションが含まれている場合は、各ディメンションの値を含める必要があります。CloudWatch は、ディメンションの一意の組み合わせをそれぞれ別のメトリクスとして扱います。ディメンションの特定の組み合わせが発行されていない場合は、その統計を取得することはできません。メトリクス作成時に使用した同じディメンションを指定する必要があります。

- **--statistics** — オプション。パーセンタイル以外のメトリクス統計。パーセンタイル統計の場合は、ExtendedStatistics を使用します。GetMetricStatistics を呼び出すときは、Statistics または ExtendedStatistics のどちらかを指定する必要があります。両方を指定することはできません。

  許可される値:

  - SampleCount
  - Average
  - Sum
  - Minimum
  - Maximum

- **--extended-statistics** — オプション。percentile の統計情報。p0.0 と p100 の間の値を指定します。GetMetricStatistics を呼び出すときは、Statistics または ExtendedStatistics のどちらかを指定する必要があります。両方を指定することはできません。

- **--unit** — オプション。特定のメトリクスの単位。メトリクスが複数の単位で報告される場合があります。単位を指定しない結果、すべての単位が返されます。メトリクスが報告しない単位のみを指定した場合、呼び出しの結果は null になります。

  使用できる値:

  - Seconds
  - Microseconds
  - Milliseconds
  - Bytes
  - Kilobytes
  - Megabytes
  - Gigabytes
  - Terabytes
  - Bits
  - Kilobytes
  - Megabits
  - Gigabits
  - Terabits
  - Percent
  - Count
  - Bytes/Second
  - Kilobytes/Second
  - Megabytes/Second
  - Gigabytes/Second
  - Terabytes/Second
  - Bits/Second
  - Kilobits/Second
  - Megabits/Second
  - Gigabits/Second
  - Terabits/Second
  - Count/Second

- None

Example

次の例では、60 秒ごとにサンプルを取得し、2 時間の期間の最大 CPUUtilization を見つけます。

Linux、macOS、Unix の場合:

```
aws cloudwatch get-metric-statistics \
       --namespace AWS/DocDB \
       --dimensions \
           Name=DBInstanceIdentifier,Value=docdb-2019-01-09-23-55-38 \
       --metric-name CPUUtilization \
       --start-time 2019-02-11T05:00:00Z \
       --end-time 2019-02-11T07:00:00Z \
       --period 60 \
       --statistics Maximum
```

Windows の場合:

```
aws cloudwatch get-metric-statistics ^
       --namespace AWS/DocDB ^
       --dimensions ^
           Name=DBInstanceIdentifier,Value=docdb-2019-01-09-23-55-38 ^
       --metric-name CPUUtilization ^
       --start-time 2019-02-11T05:00:00Z ^
       --end-time 2019-02-11T07:00:00Z ^
       --period 60 ^
       --statistics Maximum
```

このオペレーションの出力は、次のようになります。

```
{
       "Label": "CPUUtilization",
       "Datapoints": [
           {
               "Unit": "Percent",
               "Maximum": 4.49152542374361,
               "Timestamp": "2019-02-11T05:51:00Z"
           },
```

```
            {
                "Unit": "Percent",
                "Maximum": 4.25000000000485,
                "Timestamp": "2019-02-11T06:44:00Z"
            },

            ********* some output omitted for brevity *********

            {
                "Unit": "Percent",
                "Maximum": 4.33333333331878,
                "Timestamp": "2019-02-11T06:07:00Z"
            }
        ]
    }
```

Amazon DocumentDB ディメンション

Amazon DocumentDB のメトリクスは、アカウントまたはオペレーションの値で分類されます。CloudWatch コンソールを使用して、以下の表のいずれかのディメンションによってフィルタリングされた Amazon DocumentDB データを取得できます。

ディメンション	説明
DBClusterIdentifier	特定の Amazon DocumentDB クラスターに対してリクエストするデータをフィルタリングします。
DBClusterIdentifier, Role	特定の Amazon DocumentDB クラスターに対してリクエストするデータをフィルタリングして、インスタンスロール (WRITER/READER) 別にメトリクスを集計します。例えば、クラスターに属するすべての READER インスタンスのメトリクスを集計できます。
DBInstanceIdentifier	特定のデータベースインスタンスに対してリクエストするデータをフィルター処理します。

Opcounter メトリクスのモニタリング

Opcounter メトリクスは、アイドルクラスターに対してゼロ以外の値 (通常は 50 まで) を持ちます。これは、Amazon DocumentDB が定期的にヘルスチェック、内部オペレーション、およびメトリクス収集タスクを実行するためです。

データベース接続のモニタリング

`db.runCommand( { serverStatus: 1 })` などのデータベースエンジンコマンドを使用して接続数を確認すると、CloudWatch を通じて `DatabaseConnections` で確認した接続数よりも最大 10 個多い数が表示される場合があります。これは、 Amazon DocumentDB が定期的にヘルスチェックとメトリクス収集タスクを実行するためで、`DatabaseConnections` では考慮されません。`DatabaseConnections` は、お客様が開始した接続のみを表します。

を使用した Amazon DocumentDB API コールのログ記録 AWS CloudTrail

Amazon DocumentDB (MongoDB 互換) は AWS CloudTrail、Amazon DocumentDB のユーザー、ロール、または サービス (MongoDB 互換) によって実行されたアクションを記録する AWS サービスである と統合されています。CloudTrail は、Amazon DocumentDB コンソールからの呼び出しや Amazon DocumentDB SDK へのコード呼び出しを含む、Amazon DocumentDB のすべての AWS CLI API コールをイベントとしてキャプチャします。追跡を作成する場合は、Amazon DocumentDB のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [Event history (イベント履歴)] で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、Amazon DocumentDB に対するリクエスト、リクエスト元の IP アドレス、リクエストした者、リクエスト日時などのリクエストを確認できます。

> **Important**
>
> 特定の管理機能では、Amazon DocumentDB は Amazon Relational Database Service (Amazon RDS) と共有の運用テクノロジーを使用します。Amazon DocumentDB コンソール、、および API コールは AWS CLI、Amazon RDS API に対する呼び出しとして記録されます。

詳細については AWS CloudTrail、AWS CloudTrail 「 ユーザーガイド」を参照してください。

CloudTrail 内の Amazon DocumentDB 情報

アカウントを作成する AWS アカウント と、 で CloudTrail が有効になります。Amazon DocumentDB (MongoDB 互換) でアクティビティが発生すると、そのアクティビティはイベント履歴の他の AWS サービスイベントとともに CloudTrail イベントに記録されます。で最近のイベントを表示、検索、ダウンロードできます AWS アカウント。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。

Amazon DocumentDB (MongoDB 互換) 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。追跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。証跡は、 AWS パーティションのすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをより詳細に分析し、それに基づいて行動するように、他の AWS サービスを設定できます。詳細については、『AWS CloudTrail ユーザーガイド:』の以下のトピックを参照してください。

- 追跡作成の概要
- CloudTrail がサポートするサービスと統合
- CloudTrail の Amazon SNS 通知の設定
- CloudTrail ログファイルの複数のリージョンからの受け取り
- 複数のアカウントから CloudTrailログファイルを受け取る

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。ID 情報は次の判断に役立ちます。

- リクエストが、ルートと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、ロールと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のどちらを使用して送信されたか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、CloudTrail userIdentity 要素を参照してください。

Amazon DocumentDB オペレーションのプロファイリング

Amazon DocumentDB (with MongoDB compatibility) でプロファイラーを使用して、クラスターで実行されたオペレーションの実行時間と詳細をログに記録することができます。プロファイラーは、クラスターで最も遅いオペレーションをモニタリングし、個々のクエリパフォーマンスとクラスター全体のパフォーマンスを向上させるのに役立ちます。

デフォルトでは、プロファイラー機能は無効になっています。有効にすると、プロファイラーは、お客様定義のしきい値 (100 ms など) の値より時間がかかっているオペレーションを Amazon CloudWatch Logs に記録します。ログに記録される詳細には、プロファイリングされたコマンド、時間、プランの概要、クライアントメタデータが含まれます。操作が CloudWatch Logs に記録された後、CloudWatch Logs インサイトを使用して Amazon DocumentDB プロファイリングデータを分析、モニタリング、アーカイブすることができます。一般的なクエリについては、「一般的なクエリ」セクションを参照してください 。

有効にすると、プロファイラーはクラスター内の追加のリソースを利用します。高いしきい値 (500 ms など) から開始し、値を徐々に小さくして、低速なオペレーションを識別することをお勧めします。しきい値 50 ms から開始すると、高スループットのアプリケーションのクラスターでパフォーマンスの問題が発生する可能性があります。プロファイラーはクラスターレベルで有効であり、クラスター内のすべてのインスタンスとデータベースで動作します。Amazon DocumentDB は、ベストエフォート方式で Amazon CloudWatch Logs にオペレーションを記録します。

Amazon DocumentDB はプロファイラーを有効にするための追加料金を課しませんが、CloudWatch Logs の使用料は標準料金で請求されます。CloudWatch Logs の料金の詳細については、Amazon CloudWatch の料金 をご覧ください。

トピック

- サポートされているオペレーション
- 制限
- Amazon DocumentDB プロファイラーの有効化
- Amazon DocumentDB プロファイラーの無効化
- プロファイラーログのエクスポートの無効化
- Amazon DocumentDB プロファイラーログにアクセスする
- 一般的なクエリ

サポートされているオペレーション

Amazon DocumentDB プロファイラーでは、以下の操作をサポートしています。

- `aggregate`
- `count`
- `delete`
- `distinct`
- `find` (OP_QUERY およびコマンド)
- `findAndModify`
- `insert`
- `update`

制限

スロークエリプロファイラーは、クエリの結果セット全体が 1 つのバッチに収まることができ、結果セットが 16 MB 未満 (最大 BSON サイズ) の場合のみプロファイラーログを出力できます。16 MB を超える結果セットは、自動的に複数のバッチに分割されます。

ほとんどのドライバまたはシェルでは、デフォルトのバッチサイズが小さく設定されている場合があります。バッチサイズは、クエリの一部として指定することができます。低速クエリログを取得するために、予想される結果セットのサイズを超えるバッチサイズをお勧めします。結果セットのサイズが不明な場合や変動する場合は、バッチサイズを大きな数値 (100k など) に設定することもできます。

ただし、バッチサイズを大きくすると、クライアントにレスポンスを送信する前にデータベースから取得しなければならない結果も多くなります。一部のクエリでは、結果を得るまでに長い遅延が発生する可能性があります。結果セット全体を消費する予定がない場合、クエリの処理に多くの I/O を費やして結果を破棄する可能性があります。

Amazon DocumentDB プロファイラーの有効化

クラスターでプロファイラーを有効にするには、3 つのステップを実行します。すべての手順が完了していないと、プロファイリングログが CloudWatch Logs に送信されませんので、ご注意ください。プロファイラーはクラスターレベルで設定され、クラスターのすべてのデータベースとインスタンスで実行されます。

クラスターでプロファイラーを有効にするには

1. デフォルトのクラスターパラメータグループを変更できないため、使用できるカスタムクラスターパラメータグループがあることを確認してください。詳細については、「Amazon DocumentDB クラスターパラメータグループを作成する」を参照してください。
2. 使用可能なカスタムクラスターパラメータグループを使用して、`profiler`、`profiler_threshold_ms` および `profiler_sampling_rate` のパラメータを変更します。詳細については、「Amazon DocumentDB クラスターパラメータグループを変更する」を参照してください。
3. カスタムクラスターパラメータグループを使用し、CloudWatch Logs への `profiler` ログのエクスポートを有効にするために、クラスターを作成または変更します。

以下のセクションでは、 AWS Management Console と AWS Command Line Interface () を使用してこれらのステップを実装する方法を示しますAWS CLI。

Using the AWS Management Console

1. 開始する前に、Amazon DocumentDB クラスターと、まだ持っていない場合はカスタムクラスターパラメータグループを作成します。詳細については、Amazon DocumentDB クラスターパラメータグループを作成するおよびAmazon DocumentDB クラスターの作成を参照してください。
2. 使用可能なカスタムクラスターパラメータグループを使用して、以下のパラメータを変更します。詳細については、「Amazon DocumentDB クラスターパラメータグループを変更する」を参照してください。

   - `profiler` - クエリプロファイリングを有効化または無効化します。有効な値は、`enabled` および `disabled` です。デフォルト値は `disabled` です。プロファイリングを有効にするには、値を `enabled` に設定します。
   - `profiler_threshold_ms` - `profiler` を `enabled` に設定すると、`profiler_threshold_ms` より時間がかかっているコマンドは全て CloudWatch にログ出力されます。有効な値は `[50-INT_MAX]` です。デフォルト値は `100` です。
   - `profiler_sampling_rate` - プロファイリングやロギングを行うべき低速オペレーションの割合です。有効な値は `[0.0-1.0]` です。デフォルト値は `1.0` です。
3. カスタムクラスターパラメータグループを使用するようにクラスターを修正し、プロファイラーログエクスポートを Amazon CloudWatch に公開するように設定します。

a. ナビゲーションペインで、[Clusters (クラスター)] を選択し、カスタムパラメータグループをクラスターに追加します。

b. パラメータグループを関連付けるクラスター名の左側にあるボタンを選択します。[Actions (アクション)]、[Modify (変更)] の順に選択し、クラスターを変更します。

c. [Cluster options (クラスターオプション)] で、上記のステップからカスタムパラメータグループを選択して、クラスターに追加します。

d. [Log exports (ログのエクスポート)] で [Profiler logs (プロファイラーログ)] をクリックして Amazon CloudWatchを選択します。

e. [続行] を選択して、変更の概要を表示します。

f. 変更を確認したら、直ちに、あるいは、[変更のスケジューリング] の下にある次のメンテナンスウィンドウ中に適用することができます。

g. [クラスターの変更] を選択して、新しいパラメータグループでクラスターを更新します。

Using the AWS CLI

次の手順では、クラスター `sample-cluster` でサポートされているすべてのオペレーションでプロファイラーを有効にします。

1. 開始する前に、次のコマンドを実行し、名前に `default` がなく、パラメータグループファミリーとして `docdb3.6` を持つクラスターパラメータグループの出力を確認して、使用可能なカスタムクラスターパラメータグループがあることを確認します。デフォルト以外のクラスターパラメータグループがない場合は、「Amazon DocumentDB クラスターパラメータグループを作成する」を参照してください。

```
aws docdb describe-db-cluster-parameter-groups \
    --query 'DBClusterParameterGroups[*].
[DBClusterParameterGroupName,DBParameterGroupFamily]'
```

次の出力では、`sample-parameter-group` のみが両方の条件を満たしています。

```
[
       [
           "default.docdb3.6",
           "docdb3.6"
       ],
```

```
        [
            "sample-parameter-group",
            "docdb3.6"
        ]
]
```

2. カスタムクラスターパラメータグループを使用して、以下のパラメータを変更します。

   - profiler - クエリプロファイリングを有効化または無効化します。有効な値は、enabled および disabled です。デフォルト値は disabled です。プロファイリングを有効にするには、値を enabled に設定します。
   - profiler_threshold_ms - profiler を enabled に設定すると、profiler_threshold_ms より長い時間を要するすべてのコマンドが CloudWatch にログ記録されます。有効な値は [50-INT_MAX] です。デフォルト値は 100 です。
   - profiler_sampling_rate - プロファイリングやロギングを行うべき低速オペレーションの割合です。有効な値は [0.0-1.0] です。デフォルト値は 1.0 です。

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 ParameterName=profiler,ParameterValue=enabled,ApplyMethod=immediate \

 ParameterName=profiler_threshold_ms,ParameterValue=100,ApplyMethod=immediate \

 ParameterName=profiler_sampling_rate,ParameterValue=0.5,ApplyMethod=immediate
```

3. Amazon DocumentDB クラスターを修正して、前のステップの sample-parameter-group カスタムクラスターパラメータグループを使用し、パラメータ --enable-cloudwatch-logs-exports を profiler に設定するようにします。

   次のコードは、前のステップから sample-cluster を使用するようにクラスター sample-parameter-group を変更し、有効な CloudWatch Logs のエクスポートに profiler を追加します。

```
aws docdb modify-db-cluster \
       --db-cluster-identifier sample-cluster \
       --db-cluster-parameter-group-name sample-parameter-group \
       --cloudwatch-logs-export-configuration '{"EnableLogTypes":["profiler"]}'
```

このオペレーションによる出力は、次のようになります。


```
{
    "DBCluster": {
        "AvailabilityZones": [
            "us-east-1c",
            "us-east-1b",
            "us-east-1a"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "sample-cluster",
        "DBClusterParameterGroup": "sample-parameter-group",
        "DBSubnetGroup": "default",
        "Status": "available",
        "EarliestRestorableTime": "2020-04-07T02:05:12.479Z",
        "Endpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "MultiAZ": false,
        "Engine": "docdb",
        "EngineVersion": "3.6.0",
        "LatestRestorableTime": "2020-04-08T22:08:59.317Z",
        "Port": 27017,
        "MasterUsername": "test",
        "PreferredBackupWindow": "02:00-02:30",
        "PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
        "DBClusterMembers": [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-instance-2",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
```

```
            }
        ],
        "HostedZoneId": "ABCDEFGHIJKLM",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-
cluster",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2020-01-10T22:13:38.261Z",
        "EnabledCloudwatchLogsExports": [
            "profiler"
        ],
        "DeletionProtection": true
    }
}
```

Amazon DocumentDB プロファイラーの無効化

プロファイラを無効にするには、`profiler` パラメータと `profiler` ログの CloudWatch Logs へのエクスポートの両方を無効にします。

プロファイラーの無効化

`profiler` パラメータは AWS CLI、次のように AWS Management Console または を使用して無効にできます。

Using the AWS Management Console

次の手順では AWS Management Console 、 を使用して Amazon DocumentDB を無効にします`profiler`。

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。次に、プロファイラーを無効にするクラスターパラメータグループの名前を選択します。
3. 結果の [Cluster parameters (クラスターパラメータ)] ページで、`profiler` パラメータの左側にあるボタンを選択し、[Edit (編集)] を選択します。
4. [Modify profiler (プロファイラーの変更)] ダイアログボックスで、リストから [`disabled`] を選択します。

5. [Modify cluster parameter (クラスターパラメータの変更)] を選択します。

Using the AWS CLI

AWS CLIを使用してクラスターで `profiler` を無効にするには、次のようにクラスターを変更します。

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 ParameterName=profiler,ParameterValue=disabled,ApplyMethod=immediate
```

プロファイラーログのエクスポートの無効化

CloudWatch Logs への`profiler`ログのエクスポートを無効にするには AWS CLI、次のように AWS Management Console または を使用します。

Using the AWS Management Console

次の手順では、 を使用してCloudWatch AWS Management Console へのログの Amazon DocumentDB エクスポートを無効にします。

1. Amazon DocumentDB コンソールを、次の場所で開きます。https://console.aws.amazon.com/docdb
2. ナビゲーションペインで クラスター を選択します。ログのエクスポートを無効にするクラスターの名前の左側にあるボタンを選択します。
3. [アクション] メニューから [Modify (変更)] を選択します。
4. [Log exports (ログのエクスポート)] セクションまでスクロールし、[Profiler logs (プロファイラーログ)] を選択解除します。
5. [Continue] (続行) を選択します。
6. 変更を確認してから、この変更をクラスターにいつ適用するかを選択します。
   - Apply during the next scheduled maintenance window (次に予定されているメンテナンス期間中に適用)
   - すぐに適用
7. [クラスタークラスターの変更] を選択します。

Using the AWS CLI

次のコードでは、クラスター sample-cluster を変更し、CloudWatch プロファイラーログを無効にします。

Example

Linux、macOS、Unix の場合:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

Windows の場合:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

このオペレーションによる出力は、次のようになります。

```
{
    "DBCluster": {
        "AvailabilityZones": [
            "us-east-1c",
            "us-east-1b",
            "us-east-1a"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "sample-cluster",
        "DBClusterParameterGroup": "sample-parameter-group",
        "DBSubnetGroup": "default",
        "Status": "available",
        "EarliestRestorableTime": "2020-04-08T02:05:17.266Z",
        "Endpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "MultiAZ": false,
        "Engine": "docdb",
        "EngineVersion": "3.6.0",
        "LatestRestorableTime": "2020-04-09T05:14:44.356Z",
        "Port": 27017,
        "MasterUsername": "test",
```

```
        "PreferredBackupWindow": "02:00-02:30",
        "PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
        "DBClusterMembers": [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-instance-2",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "HostedZoneId": "ABCDEFGHIJKLM",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-cluster",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2020-01-10T22:13:38.261Z",
        "DeletionProtection": true
    }
}
```

Amazon DocumentDB プロファイラーログにアクセスする

Amazon CloudWatch 上のプロファイルログにアクセスするには、次のステップに従います。

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. Amazon DocumentDB クラスターと同じリージョンであることを確認します。
3. ナビゲーションペインで [ログ] を選択します。

4. クラスターのプロファイラーログを確認するには、リストで /aws/docdb/*yourClusterName*/profiler を選択します。

   各インスタンスのプロファイルログは、対応する各インスタンス名の下に表示されます。

一般的なクエリ

以下に、プロファイルされたコマンドの分析に使用できる一般的なクエリをいくつか示します。CloudWatch Logs インサイト の詳細については、CloudWatch Logs インサイトでログデータを分析する と サンプルクエリ を参照してください。

指定されたコレクションの中で最も低速だった 10 件のオペレーションを取得する

```
filter ns="test.foo" | sort millis desc | limit 10
```

特定のコレクション内で 60 ms 以上かかったすべての更新オペレーションを取得する

```
filter millis > 60 and op = "update"
```

先月最も遅かった 10 件のオペレーションを取得する

```
sort millis desc | limit 10
```

プランの概要が COLLSCAN のすべてのクエリを取得する

```
filter planSummary="COLLSCAN"
```

Performance Insights を使用したモニタリング

Performance Insights は、既存の Amazon DocumentDB モニタリング機能を拡張して、クラスターのパフォーマンスを明確にし、これに影響を与えるあらゆる問題を分析しやすくします。Performance Insights ダッシュボードを使用してデータベースロードを視覚化したり、ロードを待機、クエリステートメント、ホスト、アプリケーションでフィルタリングしたりできます。

> **Note**
>
> Performance Insights は Amazon DocumentDB 3.6、4.0、5.0 インスタンスベースのクラスターでのみ使用できます。

これはどのように役立ちますか?

- データベースのパフォーマンスを視覚化: ロードを視覚化して、データベースのロードがいつどこにあるかを判断します
- データベースのロードの原因を特定: インスタンスのロードの原因となっているクエリ、ホスト、アプリケーションを特定します
- データベースにロードがかかるタイミングを特定: Performance Insights ダッシュボードを拡大して特定のイベントに注目したり、縮小して長期間にわたる傾向を確認したりできます
- データベースロードに関するアラート: CloudWatch から新しいデータベースロードメトリクスに自動的にアクセスし、DB ロードメトリクスを他の Amazon DocumentDB メトリクスと一緒にモニタリングし、アラートを設定できます

Amazon DocumentDB Performance Insights にはどのような制限がありますか?

- AWS GovCloud (米国東部) および AWS GovCloud (米国西部) リージョンの Performance Insights は利用できません
- Amazon DocumentDB の パフォーマンスインサイトが保持するパフォーマンスデータは 7 日間までとなります
- 1,024 バイトを超えるクエリは Performance Insights に集約されません

トピック

- Performance Insights の概要
- Performance Insights の有効化と無効化
- Performance Insights 用のアクセスポリシーの設定
- Performance Insights ダッシュボードを使用してメトリクスを分析する
- Performance Insights API によるメトリクスの取得
- Performance Insights の Amazon CloudWatch メトリクス
- カウンターメトリクス用の Performance Insights

Performance Insights の概要

トピック

- 平均アクティブセッション
- ディメンション
- 最大 vCPU

平均アクティブセッション

データベース内のデータベースロード (DB ロード)アクティビティのレベルを測定します。毎秒収集されるPerformance Insights のキーメトリクスは DB Load です。DBLoad メトリクスの単位は、Amazon DocumentDB インスタンスの平均アクティブセッション数 (AAS)です。

アクティブなセッションとは、Amazon DocumentDB インスタンスにワークロードを送信し、レスポンスを待っている状態の接続です。例えば、Amazon DocumentDB インスタンスにクエリを送信すると、インスタンスでのクエリの処理中は、そのデータベースセッションは「アクティブな」状態となります。

平均アクティブセッションを取得するために、Performance Insights は、クエリを同時に実行するセッションの数をサンプリングします。AAS は、セッションの総数をサンプルの総数で割った値です。次の表は、実行中のクエリの連続する 5 つのサンプルを示しています。

例	クエリを実行しているセッション数	AAS	計算
1	2	2	2 セッション/1 サンプル
2	0	1	2 セッション/2 サンプル
3	4	2	6 セッション/3 サンプル
4	0	1.5	6 セッション/4 サンプル

例	クエリを実行しているセッション数	AAS	計算
5	4	2	10 セッション/5 サンプル

前の例では、1~5 の時間間隔の DB ロードは 2 AAS です。DB ロードの増加は、データベースで実行されているセッションが平均して増えることを意味します。

ディメンション

この DB Load メトリクスは、ディメンションと呼ばれるサブコンポーネントに分割できるため、他の時系列メトリクスとは異なります。ディメンションは、DB Load メトリクスのさまざまな特性のカテゴリと考えることができます。パフォーマンスの問題を診断する場合、最も有用なディメンションは待機状態と上位のクエリです。

待機状態

待機状態を指定すると、クエリステートメントは、特定のイベントが発生するまで待機してから、実行を継続できます。例えば、クエリステートメントは、ロック済みのリソースのロックが解除されるまで待機することがあります。DB Load と待機イベントを組み合わせると、セッションの状態の全体像を得ることができます。Amazon DocumentDB には次のようなさまざまな待機状態があります。

Amazon DocumentDB の待機状態	待機状態の説明
ラッチ	ラッチ待機状態は、セッションがバッファープールのページングを待っているときに発生します。システムによる大規模なクエリの処理やコレクションのスキャンが頻繁に行われる場合や、バッファープールが小さすぎてワーキングセットを処理できない場合に、バッファープールのページインとページアウトが頻繁に起こる可能性があります。
CPU	CPU 待機状態は、セッションが CPU 上で待機しているときに発生します。

Amazon DocumentDB の待機状態	待機状態の説明
CollectionLock	CollectionLock 待機状態は、セッションがコレクションのロック取得を待機中の場合に発生します。これらのイベントは、コレクションに対して DDL 操作が行われたときに発生します。
DocumentLock	DocumentLock 待機状態は、セッションがドキュメントのロック取得を待機中の場合に発生します。同じドキュメントへの同時書き込みの数が多いと、そのドキュメントの DocumentLock 待機状態が増えます。
SystemLock	SystemLock 待機状態は、セッションがシステムで待機しているときに発生します。これは、実行時間の長いクエリ、実行時間の長いトランザクション、またはシステムの高い同時実行性が頻繁に場合に発生する可能性があります。
IO	IO 待機状態は、セッションが IO の完了を待っているときに発生します。
BufferLock	BufferLock 待機状態は、セッションがバッファーの共有ページのロック取得を待機中の場合に発生します。BufferLock 待機状態は、他のプロセスが要求されたページのオープンカーソルを保持していると、長くなる可能性があります。
LowMemThrottle	LowMemThrottle 待機状態は、Amazon DocumentDB インスタンスに大量のメモリロードがかかってセッションが待機しているときに発生します。この状態が長時間続く場合は、インスタンスをスケールアップしてメモリを追加することを検討してください。詳細については、「リソースガバナー」を参照してください。

Amazon DocumentDB の待機状態	待機状態の説明
BackgroundActivity	BackgroundActivity 待機状態は、セッションが内部システムプロセスを待機しているときに発生します。
その他	その他の待機状態は内部待機状態です。この状態が長時間続く場合は、このクエリを終了することを検討してください。詳細については、「長時間実行されているクエリやブロックされているクエリを見つけて終了する方法」を参照してください。

上位のクエリ

待機状態はボトルネックを示しますが、上位のクエリは、どのクエリが DB ロードの最も大きな原因になっているかを示します。例えば、多くのクエリが現在データベースで実行されている可能性がありますが、1 つのクエリが DB ロードの 99% を占めている可能性もあります。この場合、ロードが高いと、クエリに問題がある可能性があります。

最大 vCPU

ダッシュボードの [データベースロード] グラフで、セッション情報が収集、集計、表示されます。アクティブなセッションが最大 CPU 容量を超えているかどうかを確認するには、最大 vCPU ラインとの関係を調べます。最大 vCPU 値は、お使いの Amazon DocumentDB インスタンスの vCPU (仮想 CPU) のコア数によって決まります。

DB ロードが [Max vCPU (最大 vCPU)] ラインをしばしば超過し、プライマリ待機状態が CPU である場合、CPU が過ロードになっています。この場合、インスタンスへの接続を抑制したり、CPU ロードの高いクエリを調整したり、より大きなインスタンスクラスを検討する必要があります。待機状態の高い一貫したインスタンスは、解決するボトルネックまたはリソースの競合問題がある可能性があることを示します。これは、DB ロードが最大 vCPU ラインを超えていない場合にも該当します。

Performance Insights の有効化と無効化

Performance Insights を使用するには、DB インスタンスで有効にします。必要に応じて、後で無効にすることができます。Performance Insights を有効または無効にしても、ダウンタイム、再起動、フェイルオーバーが発生することはありません。

Performance Insights エージェントは DB ホストの限られた CPU とメモリを消費します。DB のロードが高い場合、エージェントはデータ収集の頻度を下げることでパフォーマンスへの影響を抑えます。

クラスター作成時の Performance Insights の有効化

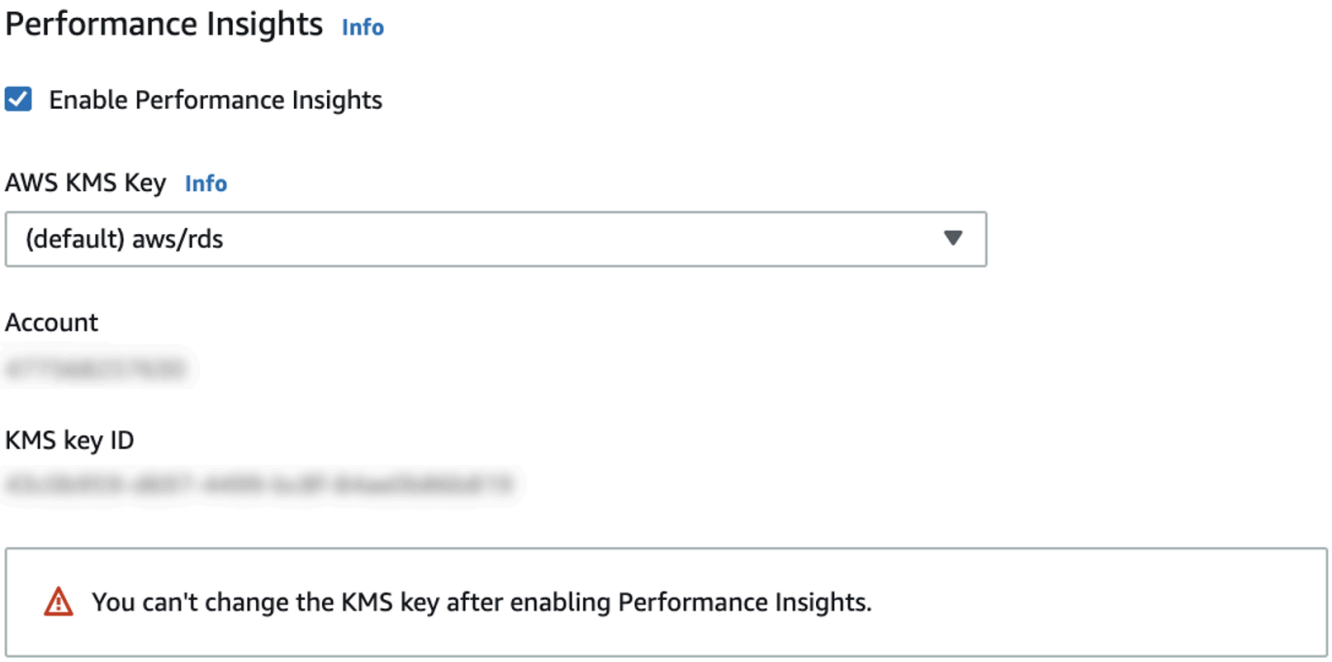
コンソールでは、新しい DB インスタンスを作成または変更するときに、Performance Insights を有効または無効にすることができます。

の使用 AWS Management Console

Amazon DocumentDB クラスターを作成する際に、コンソール上でパフォーマンスインサイトを有効にすることができます。新しい Amazon DocumentDB クラスターを作成する際に、[パフォーマンスインサイト] セクションの [パフォーマンスインサイトを有効化する] を選択することで、パフォーマンスインサイトを有効化できます。

コンソールでの手順

1. クラスターを作成する手順については、「Amazon DocumentDB クラスターの作成」を参照してください。
2. [Performance Insights] セクションで、[Performance Insights の有効化] を選択します。



> **Note**
>
> Performance Insights データ保持期間は 7 日間です。

AWS KMS key — AWS KMS キーを指定します。Performance Insights は、 AWS KMS キーを使用して、潜在的なすべての機密データを暗号化します。データは、転送中と不使用時のいずれも暗号化されます。詳細については、「Performance Insights の AWS AWS KMS ポリシーの設定」を参照してください。

インスタンス変更時の有効化と無効化

コンソールまたは AWS CLIを使用し、DB インスタンスを変更して Performance Insights を有効にできます。

Using the AWS Management Console

コンソールでの手順

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. [クラスター] を選択します。

3. DB インスタンスを選択し、[変更] を選択します。
4. [Performance Insights] セクションで、[Performance Insights の有効化] または [Performance Insightsの無効化] を選択します。

> **Note**
>
> Performance Insights を有効にするを選択した場合は、 AWS AWS KMS キーを指定できます。Performance Insights は、 AWS KMS キーを使用して、潜在的なすべての機密データを暗号化します。データは、転送中と不使用時のいずれも暗号化されます。詳細については、「保管中の Amazon DocumentDB データの暗号化」を参照してください。

5. [続行] をクリックしてください。
6. [Scheduling of Modifications] で、[Apply immediately] を選択します。「次にスケジュールされたメンテナンスウィンドウで適用」 を選択すると、インスタンスではこの設定が無視され、Performance Insights が直ちに有効になります。
7. [インスタンスの変更] を選択します。

Using the AWS CLI

`create-db-instance` または `modify-db-instance` AWS AWS CLI コマンドを使用する場合、 を指定して Performance Insights を有効にするか`--enable-performance-insights`、を指定して無効にすることができます`--no-enable-performance-insights`。

次の手順で、 AWS AWS CLIを使用して DB インスタンスで Performance Insights を有効化または無効化する方法を説明します。

AWS AWS CLI 手順

`modify-db-instance` AWS AWS CLI コマンドを呼び出し、次の値を指定します。

- `--db-instance-identifer` - DB インスタンスの名前
- 有効にする場合は `--enable-performance-insights`、無効にする場合は `--no-enable-performance-insights`

Example

次の例では、`sample-db-instance` の Performance Insights を有効にします。

For Linux, macOS, or Unix:

```
aws docdb modify-db-instance \
    --db-instance-identifier sample-db-instance \
    --enable-performance-insights
```

For Windows:

```
aws docdb modify-db-instance ^
    --db-instance-identifier sample-db-instance ^
    --enable-performance-insights
```

Performance Insights 用のアクセスポリシーの設定

Performance Insights にアクセスするには、 AWS Identity and Access Management (IAM) から適切なアクセス許可を得る必要があります。アクセス権の付与については、次のオプションがあります。

- `AmazonRDSPerformanceInsightsReadOnly` マネージドポリシーをアクセス許可セットまたはロールにアタッチします。
- カスタム IAM ポリシーを作成し、アクセス許可セットまたはロールにアタッチします。

また、Performance Insights を有効にしたときにカスタマーマネージドキーを指定した場合は、アカウント内のユーザーが KMS キーに対する `kms:Decrypt` および `kms:GenerateDataKey` アクセス許可を持っていることを確認します。

> **Note**
>
> AWS KMS キーによるencryption-at-restとセキュリティグループ管理のために、Amazon DocumentDB は Amazon RDS と共有されている運用テクノロジーを活用します。

AmazonRDSPerformanceInsightsReadOnly ポリシーの IAM プリンシパルへのアタッチ

`AmazonRDSPerformanceInsightsReadOnly` は、Amazon DocumentDB Performance Insights API のすべての読み取り専用オペレーションへのアクセスを許可する AWSマネージドポリシーです。現在、この API のすべてのオペレーションは読み取り専用で

す。AmazonRDSPerformanceInsightsReadOnly をアクセス許可セットまたはロールにアタッチすると、受取人は他のコンソール機能とともに Performance Insights を使用できます。

Performance Insights 用のカスタム IAM ポリシーの作成

AmazonRDSPerformanceInsightsReadOnly ポリシーを持たないユーザーの場合は、ユーザーマネージド IAM ポリシーを作成または変更して、Performance Insights に対するアクセス権限を付与できます。ポリシーをアクセス許可セットまたはロールにアタッチすると、受取人は Performance Insights を使用できます。

カスタムポリシーを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで、ポリシー を選択してください。
3. [Create policy] (ポリシーの作成) を選択します。
4. [ポリシーの作成] ページで、[JSON] タブを選択します。
5. 次のテキストをコピーして貼り付け、*us-east-1* を AWS リージョンの名前に置き換え、*111122223333* を顧客アカウント番号に置き換えます。

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds:DescribeDBInstances",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "rds:DescribeDBClusters",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:DescribeDimensionKeys",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": "pi:GetDimensionKeyDetails",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:GetResourceMetadata",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:GetResourceMetrics",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:ListAvailableResourceDimensions",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:ListAvailableResourceMetrics",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        }
    ]
}
```

6. [ポリシーの確認] を選択します。
7. ポリシーの名前と (必要に応じて) 説明を入力し、[ポリシーの作成] を選択します。

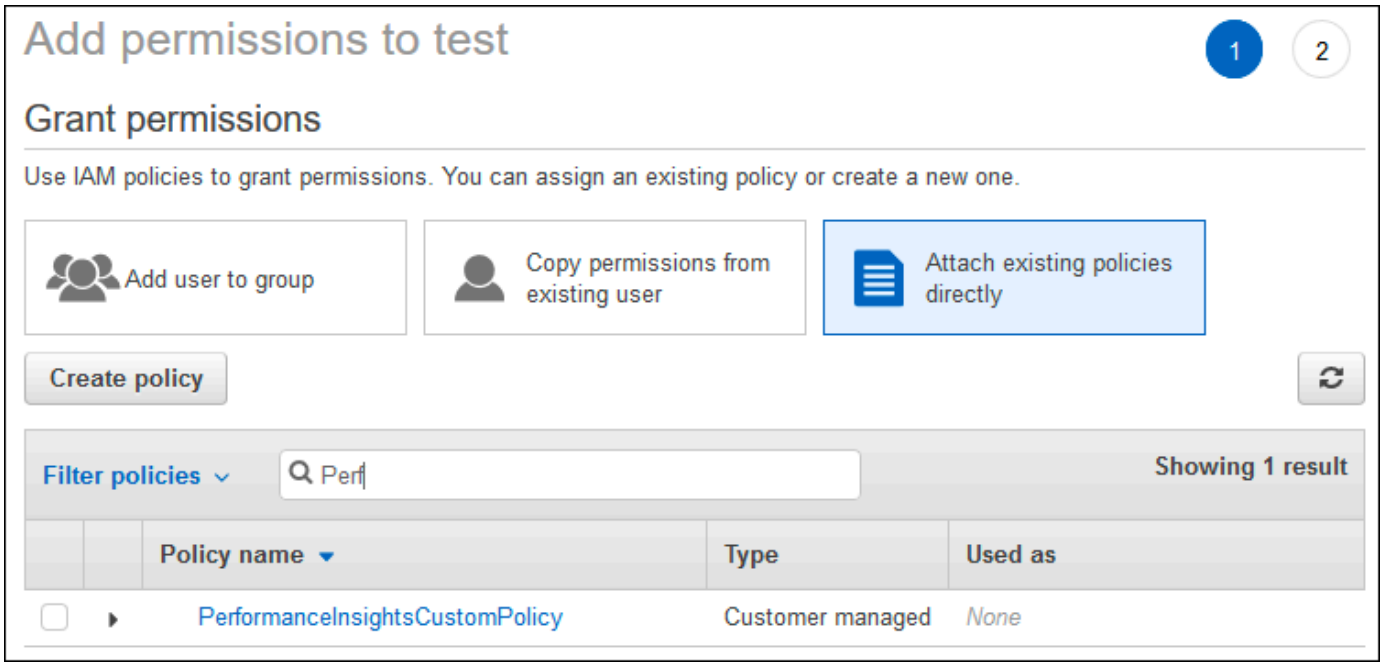
これで、そのポリシーをアクセス許可セットまたはロールにアタッチできます。次の手順では、この目的で使用できるユーザーが既に存在することを前提としています。

ポリシーをユーザーにアタッチするには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで [ユーザー] を選択します。
3. リストから存在するユーザーを 1 人選択します。

> **Important**
>
> Performance Insights を使用するには、カスタムポリシーのほかに別のポリシーで、Amazon DocumentDB へのアクセスを許可されている必要があります。たとえば、事前定義された AmazonDocDBReadOnlyAccess ポリシーは Amazon DocDB への読み取り専用アクセスを許可します。詳細については、「ポリシーを使用したアクセスの管理」を参照してください。

4. [Summary (概要)] ページで、[Add permissions (許可の追加)] を選択します。
5. [既存のポリシーを直接アタッチする] を選択します。検索を行う場合は、次のようにポリシー名の初期の数文字を入力します。



6. ポリシーを選択し、[次へ: レビュー] を選択します。
7. [アクセス権限の追加] を選択します。

Performance Insights の AWS KMS ポリシーの設定

Performance Insights は、 を使用して機密データ AWS KMS key を暗号化します。API またはコンソールを通じて Performance Insights を有効にする場合は、次のオプションを使用できます。

- デフォルトを選択します AWS マネージドキー。

Amazon DocumentDB は、新しい DB インスタンス AWS マネージドキー に を使用します。Amazon DocumentDB は、 AWS アカウントの AWS マネージドキー を作成します。 AWS アカウントは、 AWS リージョンごとに AWS マネージドキー Amazon DocumentDB で異なります。

- カスタマーマネージドキーを選択します。

  カスタマーマネージドキーを指定する場合、Performance Insights API を呼び出すアカウント内のユーザーは、KMS キーに対する `kms:Decrypt` および `kms:GenerateDataKey` アクセス許可が必要です。IAM ポリシーを使用して、これらのアクセス許可を設定できます。ただし、KMS キーポリシーを使用してこれらのアクセス許可を管理することをお勧めします。詳細については、AWS KMS でのキーポリシーの使用を参照してください。

Example

次のサンプルキーポリシーでは、KMS キーポリシーにステートメントを追加する方法が示されています。これらのステートメントは、Performance Insights へのアクセスを許可します。の使用方法によっては AWS KMS、いくつかの制限を変更することもできます。ポリシーにステートメントを追加する前に、すべてのコメントを削除してください。

JSON

```
{
 "Version" : "2012-10-17",
 "Id" : "your-policy",
 "Statement" : [ {
    //This represents a statement that currently exists in your policy.
 }
 ....,
 //Starting here, add new statement to your policy for Performance Insights.
 //We recommend that you add one new statement for every RDS/DocumentDB instance
 {
    "Sid" : "Allow viewing RDS Performance Insights",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            //One or more principals allowed to access Performance Insights
            "arn:aws:iam::444455556666:role/Role1"
        ]
    },
```

```
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition" :{
        "StringEquals" : {
            //Restrict access to only RDS APIs (including Performance Insights).
            //Replace *region* with your AWS Region.
            //For example, specify us-west-2.
            "kms:ViaService" : "rds.*region*.amazonaws.com"
        },
        "ForAnyValue:StringEquals": {
            //Restrict access to only data encrypted by Performance Insights.
            "kms:EncryptionContext:aws:pi:service": "rds",
            "kms:EncryptionContext:service": "pi",

            //Restrict access to a specific DocDB instance.
            //The value is a DbiResourceId.
           "kms:EncryptionContext:aws:rds:db-id": "db-AAAAABBBBBCCCCDDDDDEEEEE"
        }
    }
}
```

Performance Insights ダッシュボードを使用してメトリクスを分析する

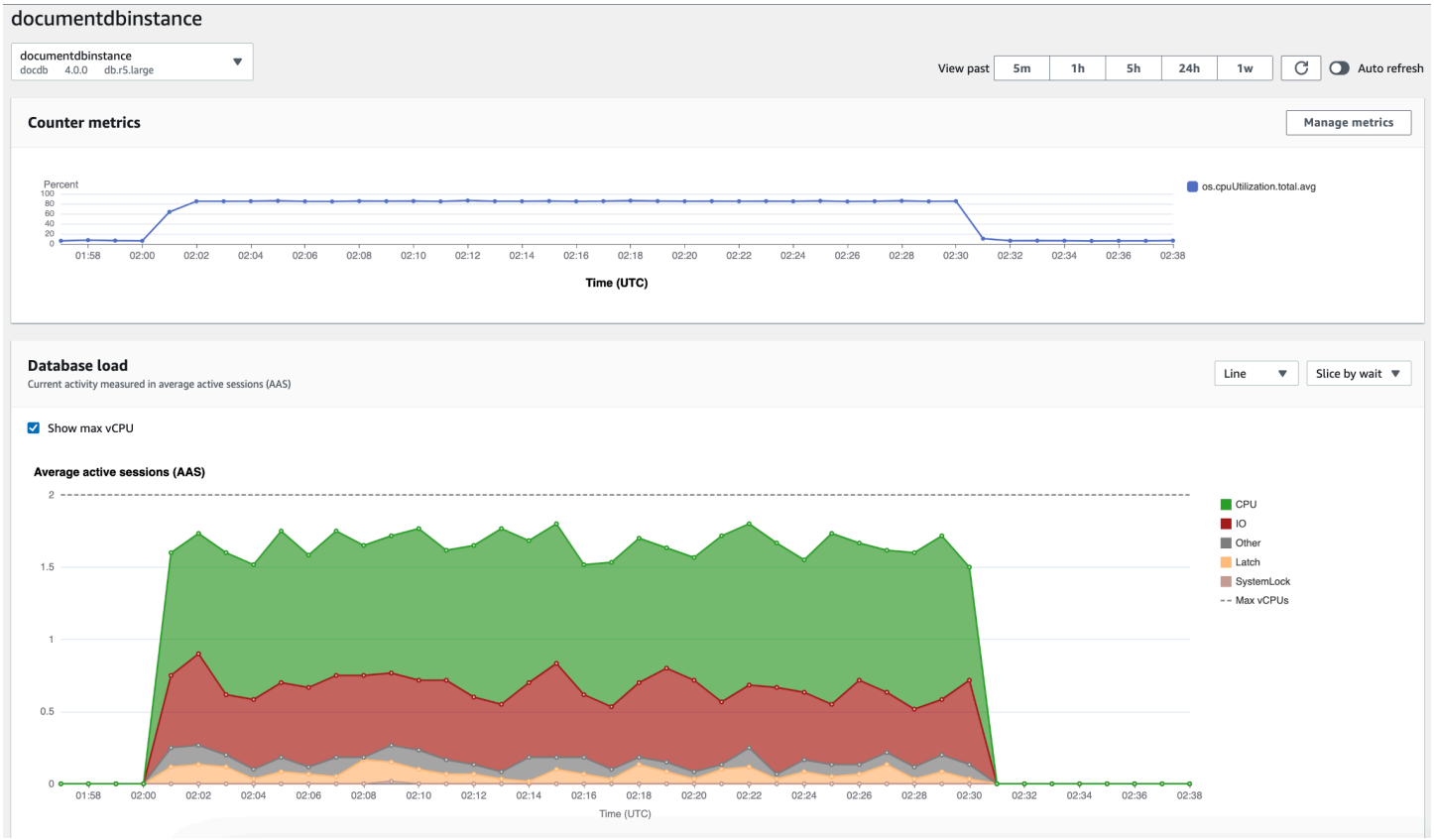
Performance Insights ダッシュボードには、パフォーマンスの問題を分析し、解決するのに役立つ、データベースのパフォーマンス情報が含まれます。ダッシュボードのメインページで、データベースロード (DB ロード) に関する情報を確認できます。待機状態やクエリなどのディメンションによって、DB のロードを「スライス」することが可能です。

トピック

- Performance Insights ダッシュボードの概要
- Performance Insights ダッシュボードを開く
- 待機状態によるデータベースロードの分析
- [上位のクエリ] タブの概要
- データベースロードグラフのズームイン

Performance Insights ダッシュボードの概要

ダッシュボードは、Performance Insights を操作する最も簡単な方法です。次の例では、Amazon DocumentDB インスタンスのダッシュボードを示しています。デフォルトでは、Performance Insights ダッシュボードには過去 60 分間のデータが表示されます。



このダッシュボードは、以下の各パートに分割されています。

1. カウンターメトリクス - 特定のパフォーマンスカウンターメトリクスのデータを示します。
2. データベースロード - DB ロードと DB インスタンスの容量を比較して、最大 vCPU ラインで表します。
3. 上位ディメンション – DB ロードに影響が大きい上位ディメンションを表示します。これらのディメンションには `waits`、`queries`、`hosts`、`databases`、`applications` が含まれます。

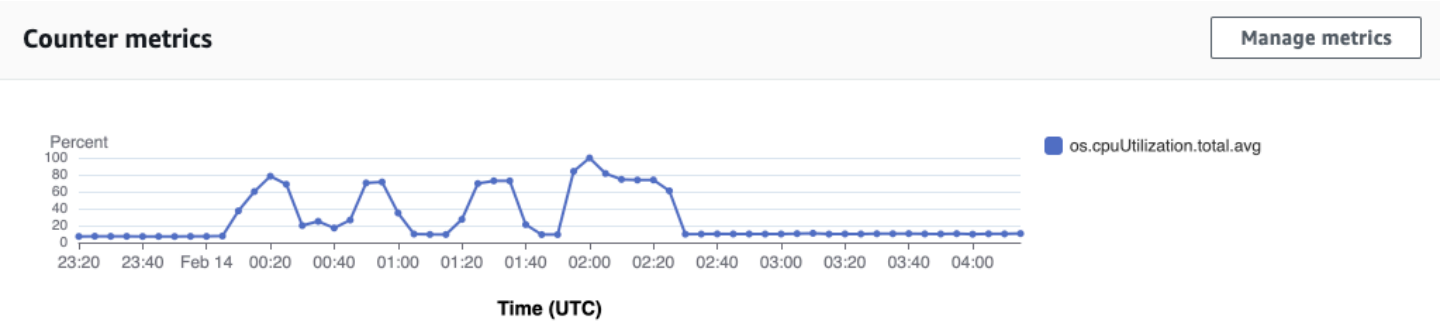
トピック

- カウンターメトリクスグラフ
- データベースロードのグラフ
- 上位のディメンションテーブル

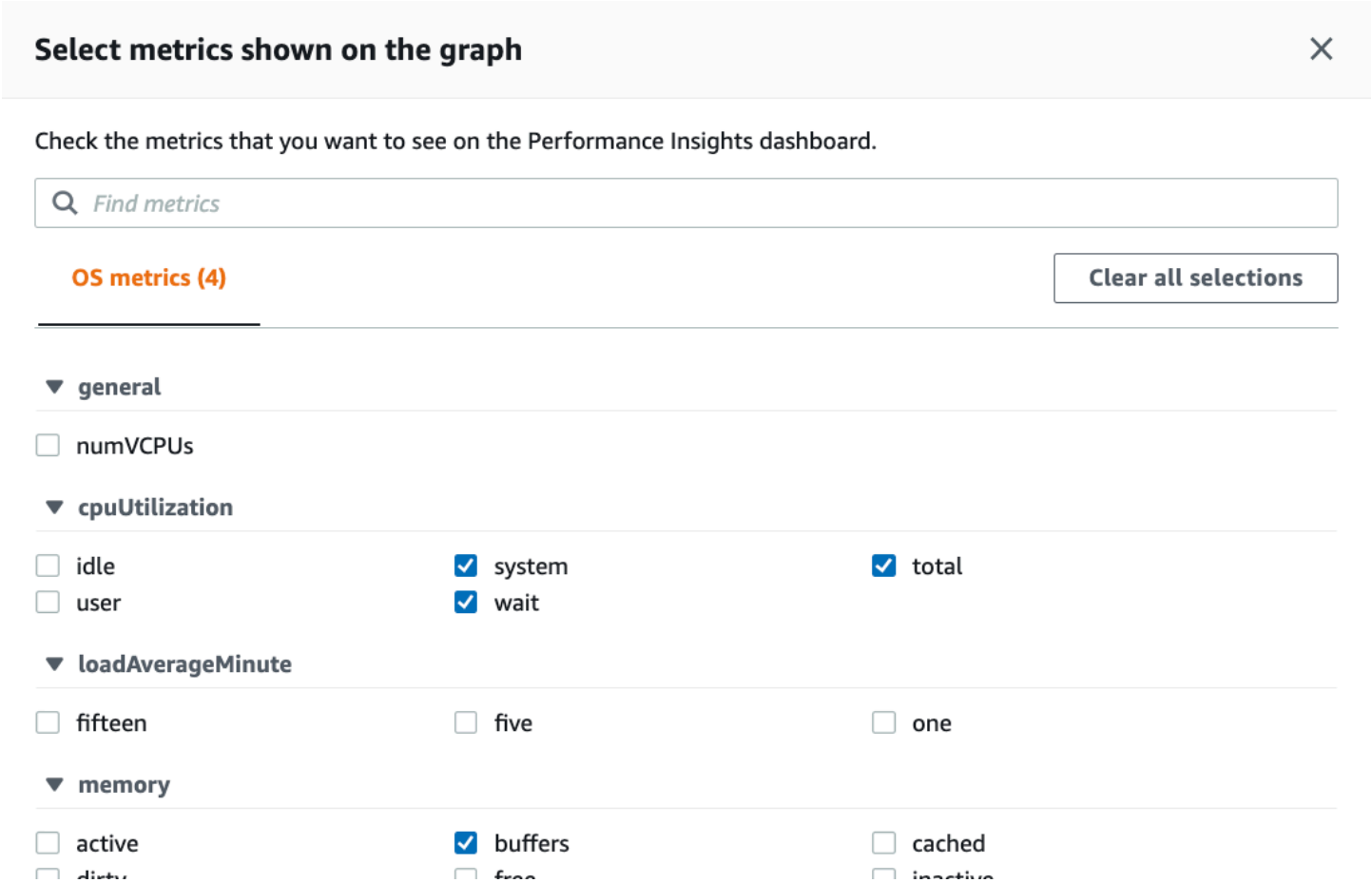
カウンターメトリクスグラフ

カウンターメトリクスを使用すると、Performance Insights ダッシュボードをカスタマイズして最大 10 個の追加グラフを含めることができます。これらのグラフは、数十種類のオペレーティングシステムメトリクスの一部を示しています。この情報をデータベースロードと関連付けることで、パフォーマンスの問題を特定して分析できます。

カウンターメトリクスグラフはパフォーマンスカウンターのデータを表示します。



メトリクスの管理を選択して、パフォーマンスカウンターを変更します。以下のスクリーンショットに示すように、複数の OS メトリクスを選択できます。メトリクスの詳細を表示するには、メトリクス名にカーソルを合わせます。



データベースロードのグラフ

データベースロードは、データベースアクティビティとインスタンス容量の比較結果が 最大 vCPU の折れ線グラフとして表示されます。デフォルトでは、折れ線グラフは DB ロードを単位時間あたりの平均アクティブセッションで表します。DB ロードは、待機状態でスライス (グループ化) されます。



ディメンションでスライスされた DB のロード

サポートされているディメンション別にグループ化された、アクティブなセッションとしてロードを表示するように選択できます。次の図に、Amazon DocumentDB インスタンスのディメンションを示します。

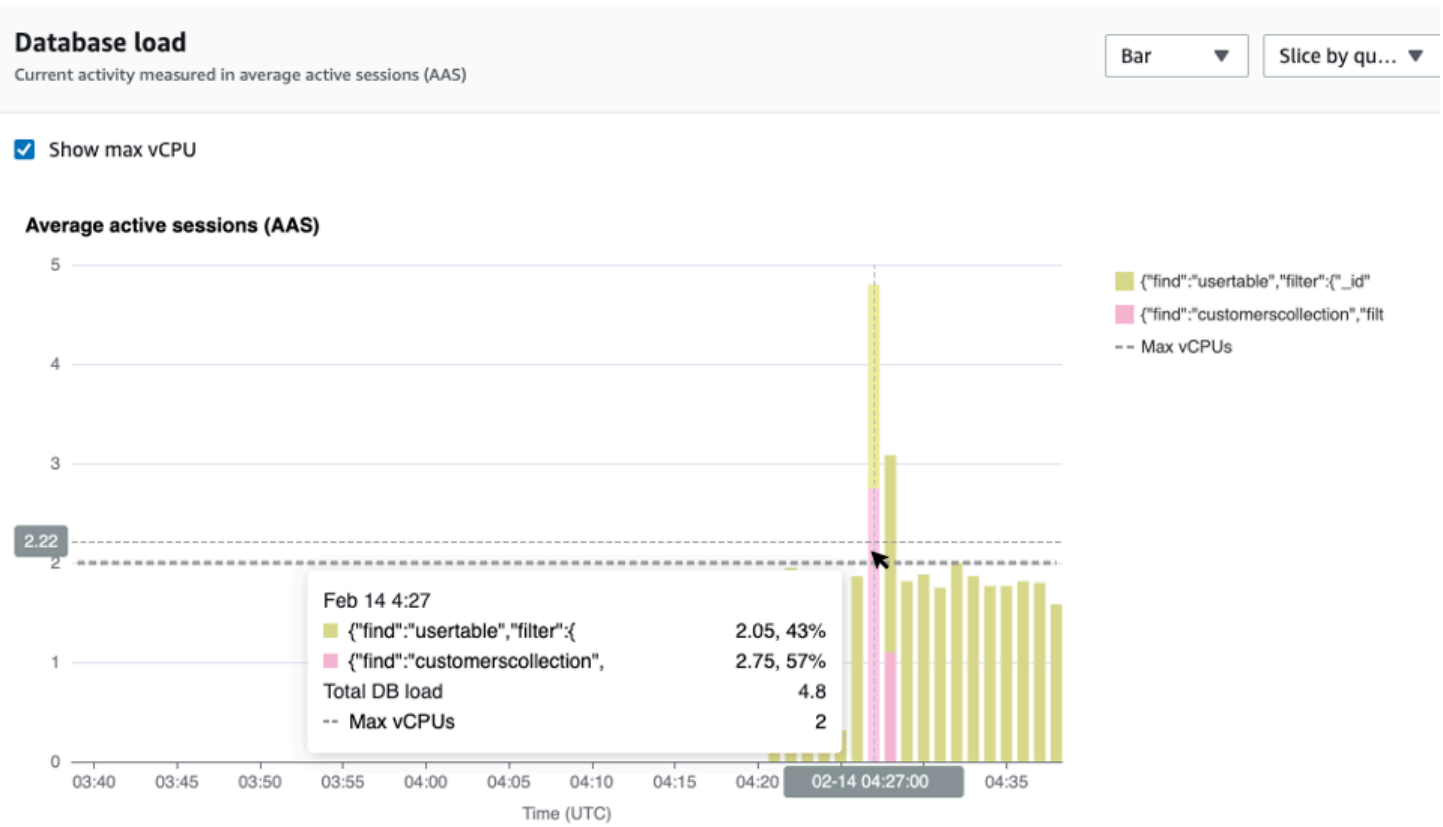


ディメンション項目に関する DB ロードの詳細

ディメンション内の DB ロード項目の詳細を表示するには、項目名にカーソルを合わせます。次の図は、クエリステートメントの詳細を示しています。



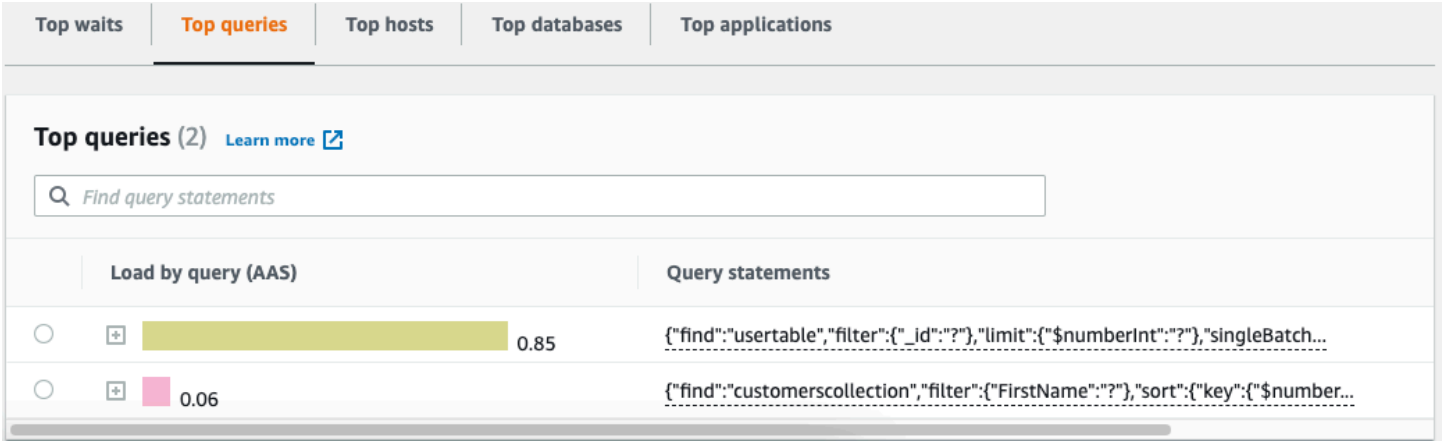
凡例で選択した期間に対する項目の詳細を表示するには、その項目にカーソルを合わせます。



上位のディメンションテーブル

上位ディメンションテーブルは、DB ロードを異なる次元でスライスします。ディメンションとは、DB ロードの異なる特性についてのカテゴリまたは「スライス化」のことです。ディメンションがクエリの場合、上位のクエリは、DB ロードに最も寄与しているクエリステートメントを表示します。

以下のディメンションタブのいずれかを選択します。



次の表では、各タブの簡単な説明を示します。

説明
上位ディメンションテーブルの一覧
待機
バックエンドが待っている

説明
イベント
現在実行中のクエリステートメント

説明
接続されているクライアントのホスト IP

説明
上位データベースユーザーの名前

説明
データベースへの接続を確立したアプリケーションの名前。

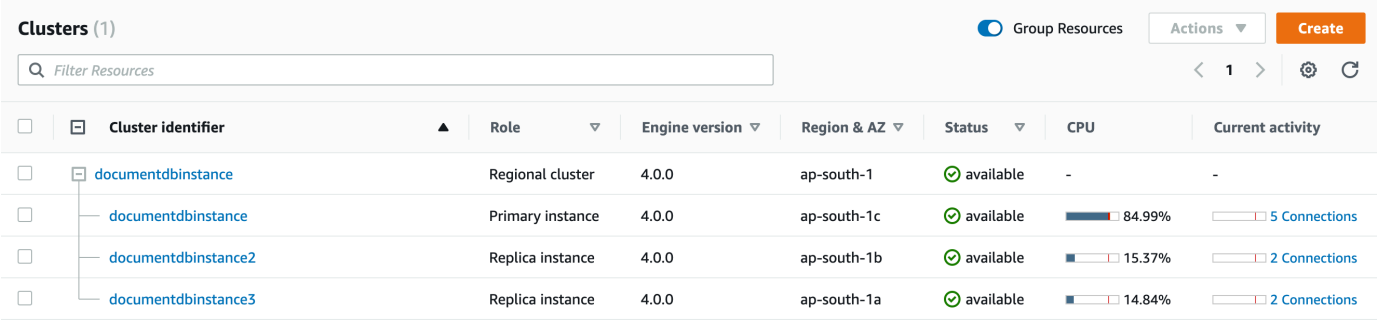
[上位のクエリ] タブを使用してクエリを分析する方法については、「[上位のクエリ] タブの概要」を参照してください。

Performance Insights ダッシュボードを開く

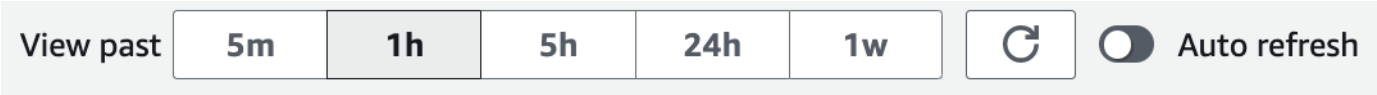
AWS マネジメントコンソールで Performance Insights ダッシュボードを表示するには、次のステップを使用します。

1. https://console.aws.amazon.com/docdb/ で Performance Insights コンソールを開きます。
2. DB インスタンスを選択します。その Amazon DocumentDB インスタンスの Performance Insights ダッシュボードが表示されます。

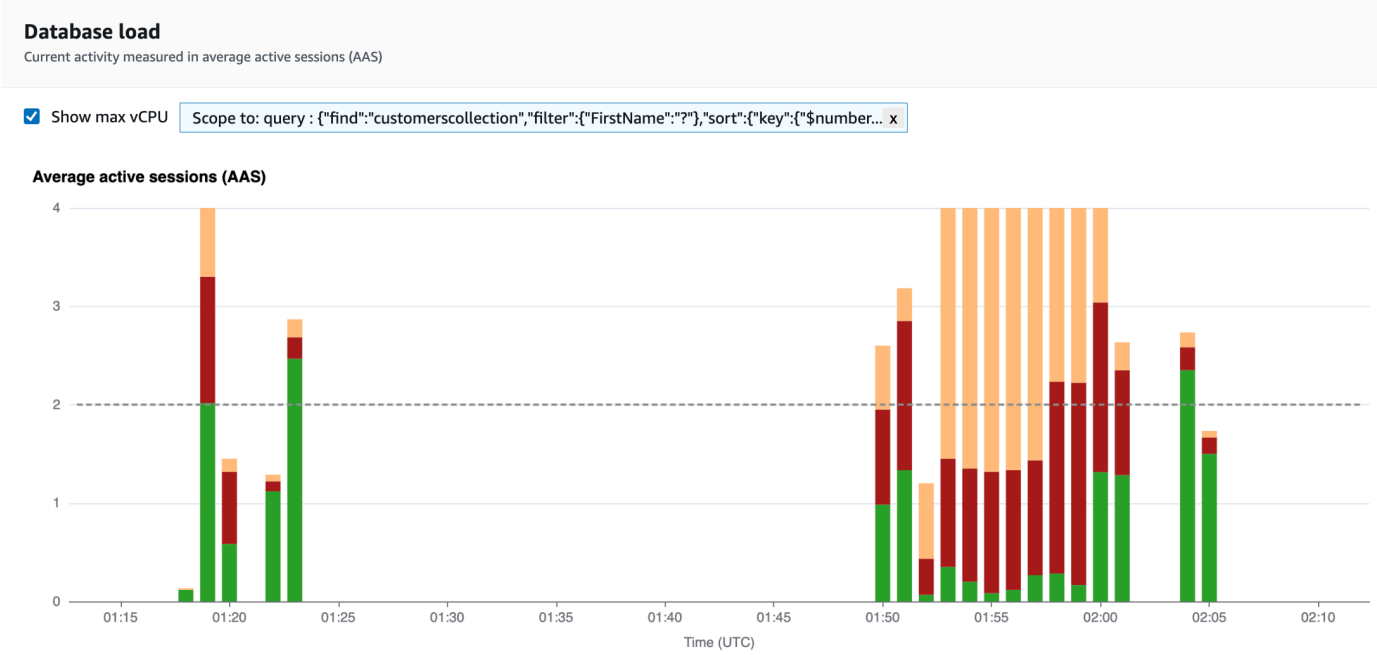
Performance Insights を有効にした Amazon DocumentDB インスタンスでは、インスタンスのリストで [セッション] 項目を選択してダッシュボードにアクセスすることもできます。[現在のアクティビティ] の [セッション] 項目には、直近 5 分間におけるアクティブなセッションの平均データベースロードが表示されます。ロードはバーでグラフィカルに示されます。バーが空の場合、インスタンスはアイドル状態です。ロードが増加すると、バーが青色で塗りつぶされます。ロードがインスタンスクラスの仮想 CPU (vCPU) の数を超えると、バーが赤色になり、ボトルネックとなる可能性があることが示されます。



3. (オプション) 右上のボタンを選択して、別の時間間隔を選択します。例えば、間隔を 1 時間に変更するには、[1 時間] を選択します。



以下のスクリーンショットでは、DB ロードの間隔は 1 時間です。



4. データを自動的に更新するには、[自動更新] を有効にします。



Performance Insights ダッシュボードが自動的に新しいデータで更新されます。更新の頻度は、表示されるデータの量によって異なります。

- 「5 分」は 5 秒ごとに更新されます。
- 「1 時間」は 1 分ごとに更新されます。
- 「5 時間」は 1 分ごとに更新されます。
- 「24 時間」は 5 分ごとに更新されます。
- 「1 週間」は 1 時間ごとに更新されます。

待機状態によるデータベースロードの分析

データベースロード (DB ロード) のグラフにボトルネックが表示される場合、ロードの発生源を確認できます。これを実行するには、データベースロードグラフ下にある[上位ロード項目] テーブルを参照してください。クエリやアプリケーションのような特定の項目を選択すると、その項目をドリルダウンして詳細を表示できます。

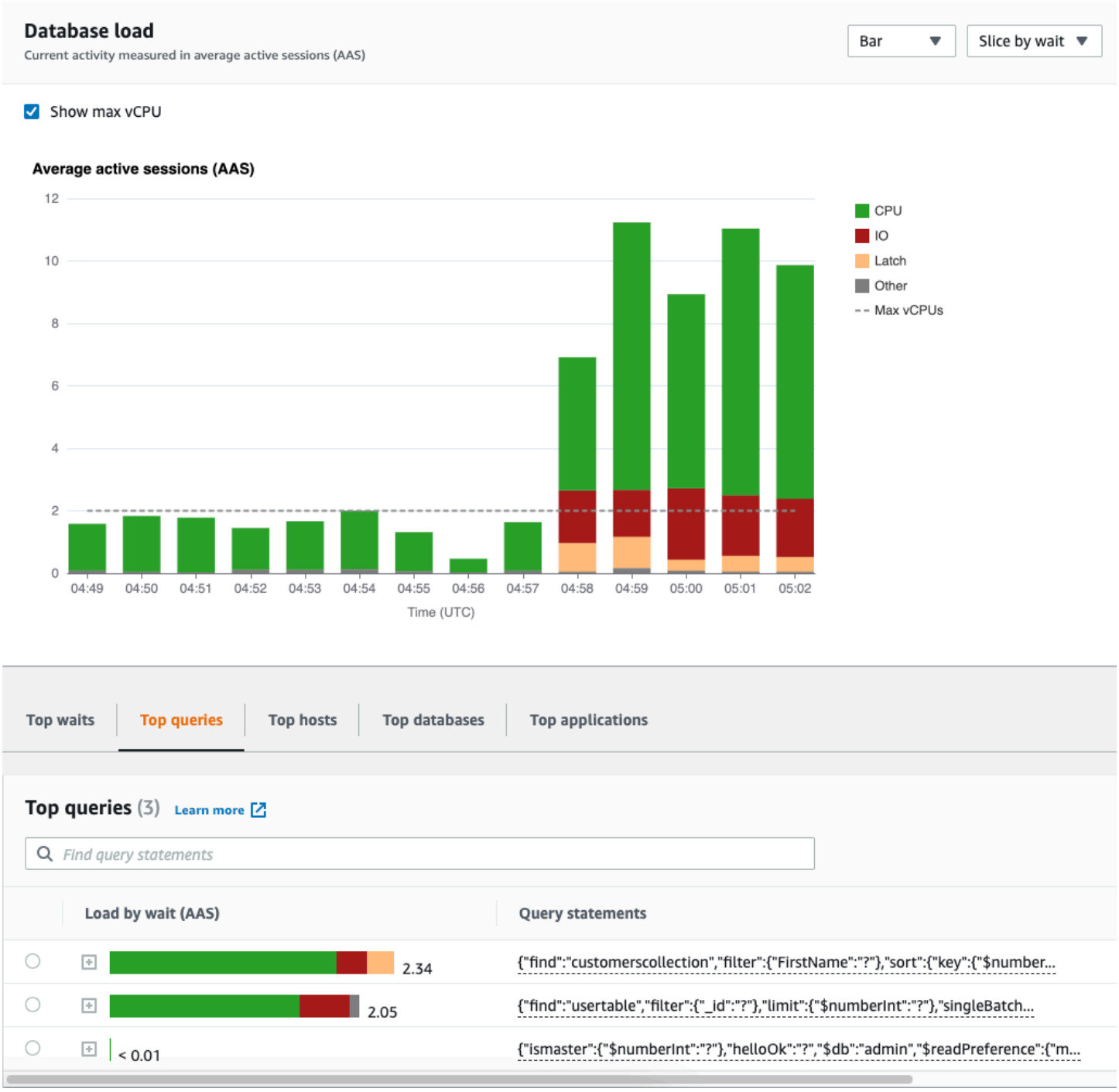
待機および上位クエリによってグループ分けされた DB ロードは、通常、パフォーマンス問題に関する最も正しい情報を提供します。待機でグループ化された DB ロードは、データベースにリソースまたは同時のボトルネックがあるかどうかを示します。この場合、上位ロード項目のテーブルの [上位のクエリ] タブには、どのクエリがそのロードをかけているかが表示されます。

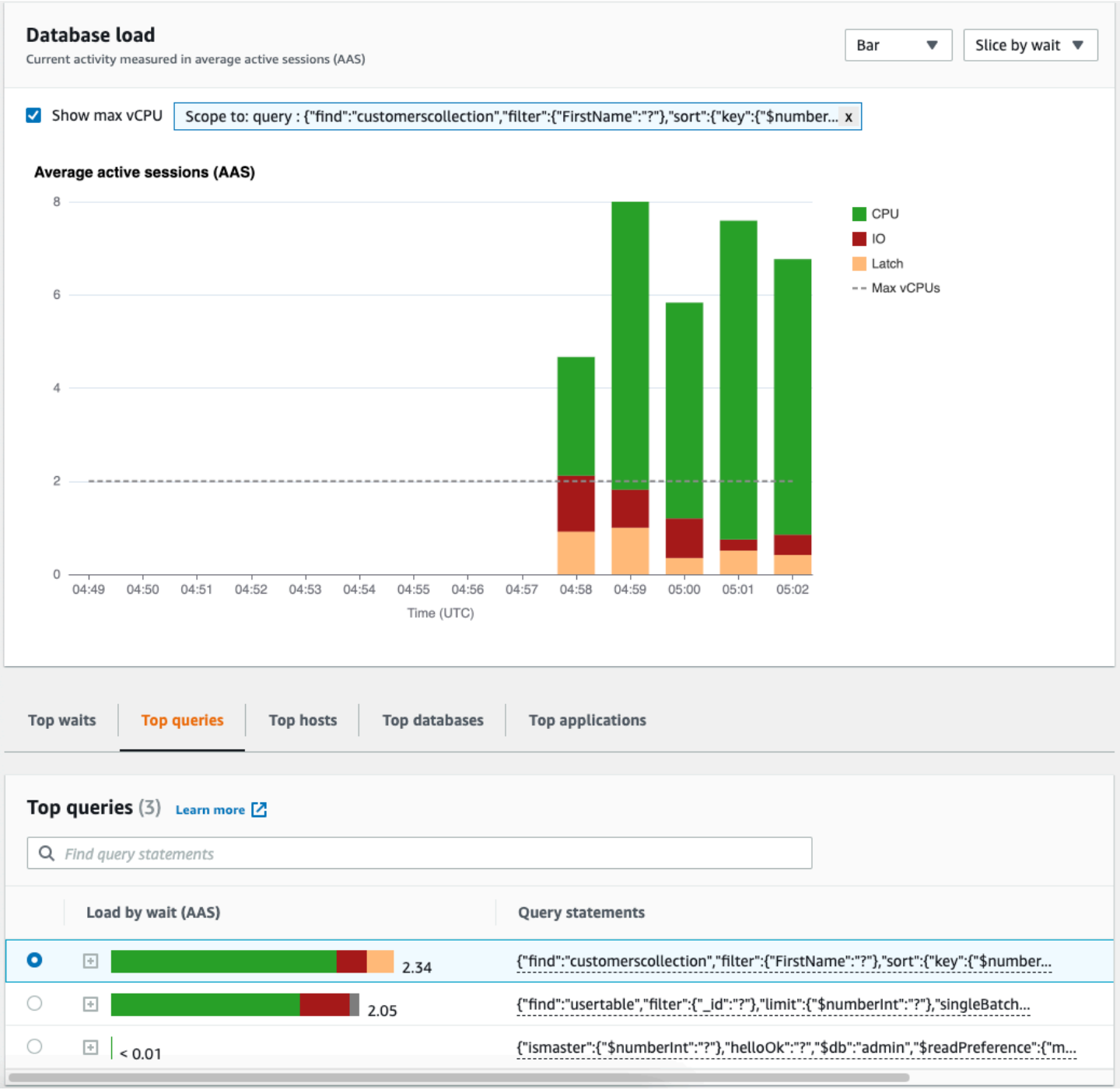
パフォーマンスの問題を診断するための一般的なワークフローは次のとおりです。

1. 「データベースロード」 グラフを確認し、最大 CPU ラインを超えているデータベースロードのインシデントがあるかどうかを確認します。
2. ある場合は、「データベースロード」 グラフを確認して、どの待機状態 (複数) が主に原因であるかを特定します。
3. 上位のロード項目テーブルの [上位のクエリ] タブが待機状態に最も影響しているクエリを確認することによって、ロードを引き起こすダイジェストクエリを特定します。これらは [待機別ロード (AAS)] 列で識別できます。
4. [上位のクエリ] タブでこれらのダイジェストクエリの 1 つを選択して展開し、構成されている子クエリを確認します。

また、[上位のホスト] または [上位のアプリケーション] をそれぞれ選択することで、どのホストまたはアプリケーションが最もロードを発生させているかを確認することもできます。アプリケーション名は Amazon DocumentDB インスタンスへの接続文字列で指定されます。Unknown は、アプリケーションフィールドが指定されなかったことを示します。

例えば、以下のダッシュボードで、CPU の待機はほとんどの DB ロードの主な原因となっています。[上位のクエリ] で上位クエリを選択すると、選択クエリによって発生しているロードが最も大きいことに焦点を当てるようにデータベースロードチャートが絞り込まれます。


Database load
Current activity measured in average active sessions (AAS)
Bar
Slice by wait
Show max vCPU
Average active sessions (AAS)
12
10
8
6
4
2
0
04:49
04:50
04:51
04:52
04:53
04:54
04:55
04:56
04:57
04:58
04:59
05:00
05:01
05:02
Time (UTC)
CPU
IO
Latch
Other
Max vCPUs
Top waits
Top queries
Top hosts
Top databases
Top applications
Top queries (3)
Learn more
Find query statements
Load by wait (AAS)
Query statements
2.34
{"find":"customerscollection","filter":{"FirstName":"?"},"sort":{"key":{"$number...
2.05
{"find":"usertable","filter":{"_id":"?"},"limit":{"$numberInt":"?"},"singleBatch...
< 0.01
{"ismaster":{"$numberInt":"?"},"helloOk":"?","$db":"admin","$readPreference":{"m...



[上位のクエリ] タブの概要

デフォルトでは、[上位のクエリ] タブはデータベースロードに最も貢献しているクエリを表示します。クエリテキストを分析して、クエリの調整に役立てることができます。

トピック

- クエリダイジェスト

- 待機によるロード (AAS)
- 詳細なクエリ情報を表示します
- ステートメントのクエリテキストへのアクセス
- ステートメントのクエリテキストの表示とダウンロード

クエリダイジェスト

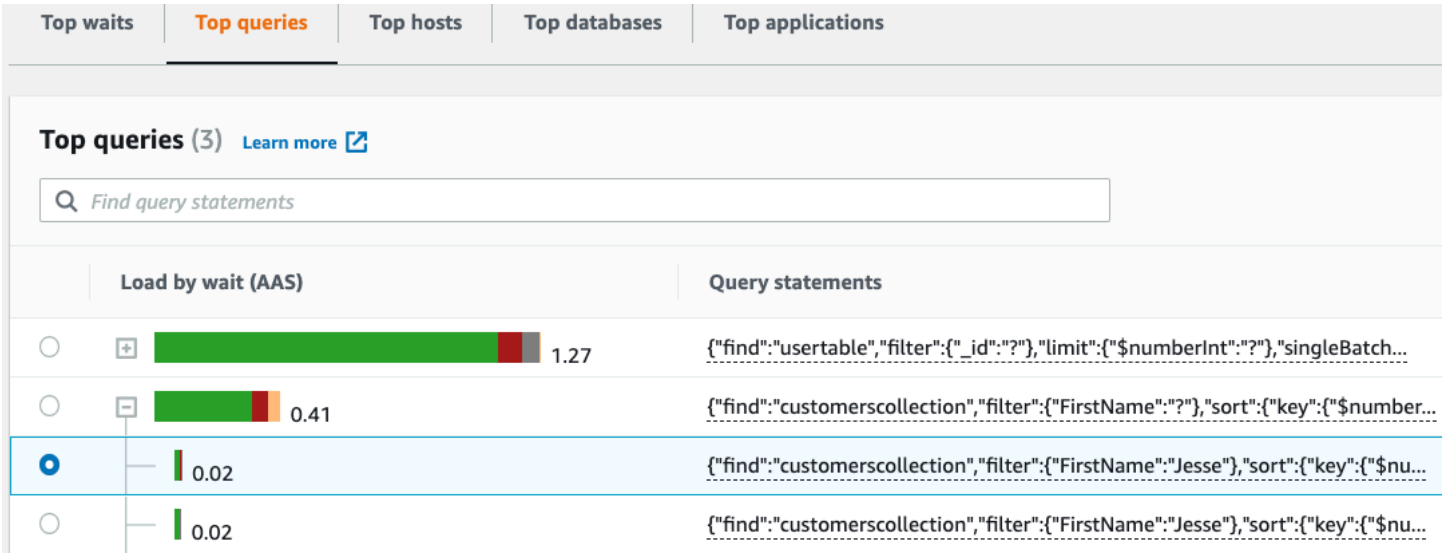
クエリダイジェストは、構造的には類似しているが、異なるリテラル値を含む可能性の高い、複数の実際のクエリの複合体です。ダイジェストは、ハードコードされた値を疑問符に置き換えます。クエリダイジェストの例は次のようになります。

```
{"find":"customerscollection","filter":{"FirstName":"?"},"sort":{"key":
{"$numberInt":"?"}},"limit":{"$numberInt":"?"}}
```

このダイジェストには、次の子クエリが含まれます。

```
{"find":"customerscollection","filter":{"FirstName":"Karrie"},"sort":{"key":
{"$numberInt":"1"}},"limit":{"$numberInt":"3"}}
{"find":"customerscollection","filter":{"FirstName":"Met"},"sort":{"key":
{"$numberInt":"1"}},"limit":{"$numberInt":"3"}}
{"find":"customerscollection","filter":{"FirstName":"Rashin"},"sort":{"key":
{"$numberInt":"1"}},"limit":{"$numberInt":"3"}}
```

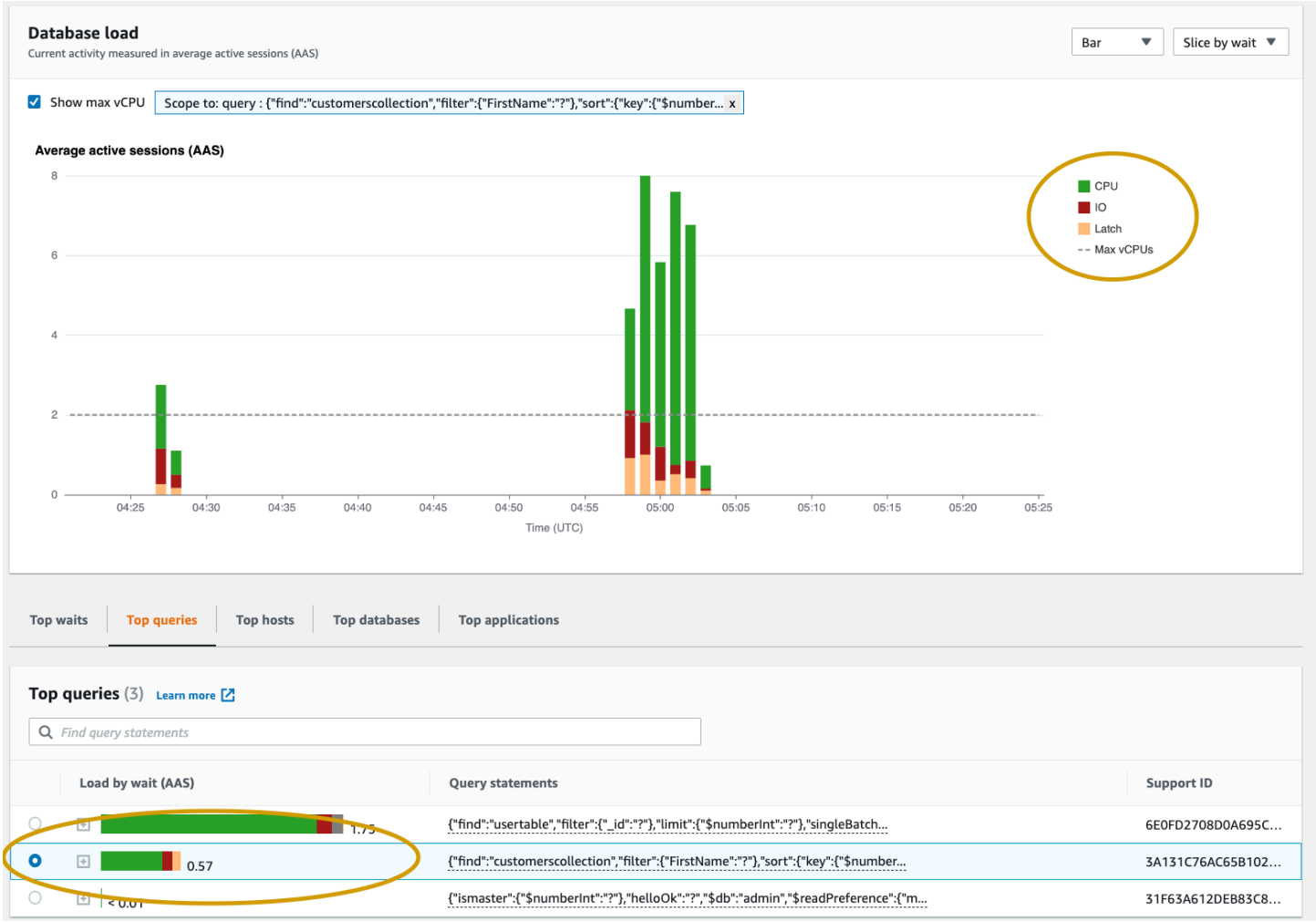
ダイジェスト内でリテラルクエリステートメントを表示するには、クエリを選択してからプラス記号 (+) を選択します。以下のスクリーンショットでは、選択されたクエリはダイジェストです。



> **Note**
>
> クエリダイジェストでは、類似したクエリステートメントがグループ化されますが、機密情報は編集されません。

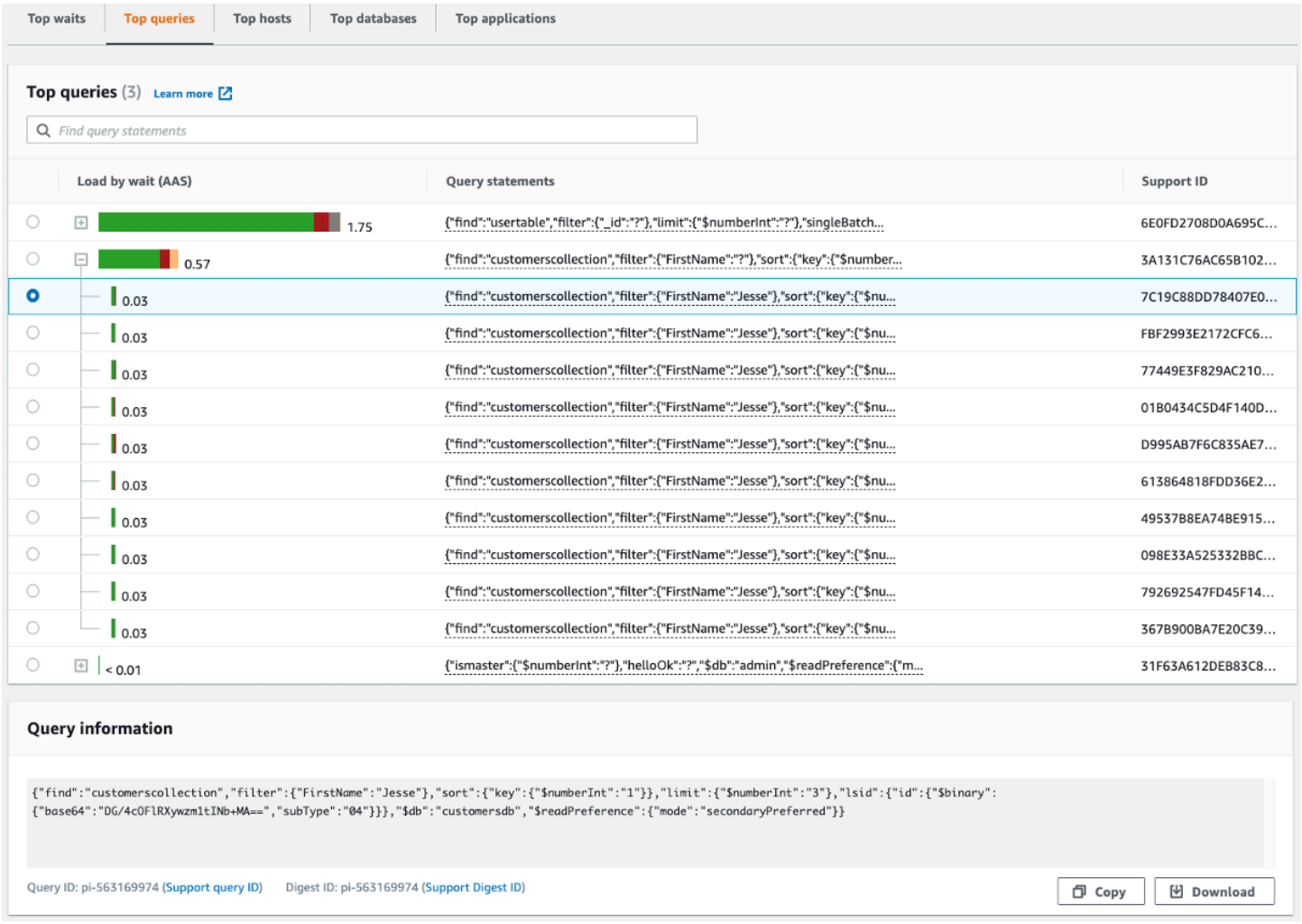
待機によるロード (AAS)

[上位のクエリ] の [待機別のロード (AAS)] 列は、上位の各ロード項目に関連付けられているデータベースロードの割合を示しています。この列には、[DB ロードグラフ] で現在選択されているグループ化に応じて、その項目に対するロードが反映されます。例えば、DB ロードグラフを待機状態別にグループ化できます。この場合は、[待機別の DB ロード] バーは、クエリが貢献している待機状態の量を示すために、サイズ、セグメント、および色で分けられています。また、選択したクエリに影響を与えている待機状態も示されます。



詳細なクエリ情報を表示します

[上位のクエリ] テーブルで、ダイジェストステートメントを開いてその情報を表示できます。下部のペインに情報が表示されます。



クエリステートメントに関連付けられているタイプの識別子 (ID) は以下のとおりです。

1. サポートクエリ ID: クエリ ID のハッシュ値。この値は、 AWS Support で作業しているときにクエリ ID を参照するためのものです。 AWS Support は実際のクエリ IDsとクエリテキストにアクセスできません。
2. ダイジェスト ID のサポート - ダイジェスト ID のハッシュ値。この値は、 AWS Support で作業しているときにダイジェスト ID を参照するためのものです。 AWS Support は実際のダイジェスト IDsとクエリテキストにアクセスできません。

ステートメントのクエリテキストへのアクセス

デフォルトでは、[上位のクエリ] テーブルの各行には各クエリステートメントごとに 500 バイトのクエリテキストが表示されます。ダイジェストステートメントのサイズが 500 バイトを超える場合、Performance Insights ダッシュボードでこのステートメントを開いて、テキストの表示量を増やすことができます。この場合、表示されるクエリの最大長は 1 KB です。フルクエリステートメントを表示する場合は、[ダウンロード] を選択することもできます。

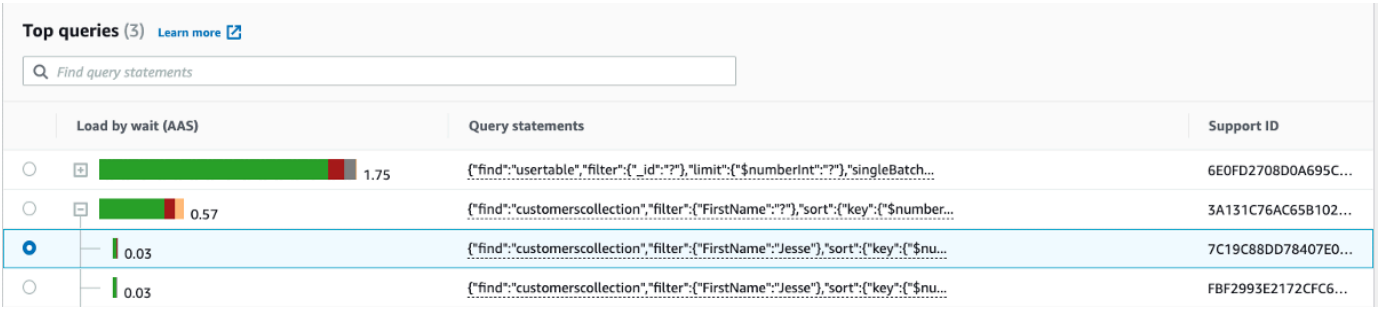
ステートメントのクエリテキストの表示とダウンロード

Performance Insights ダッシュボードで、クエリテキストを表示およびダウンロードできます。

Performance Insights ダッシュボードでクエリテキストの表示量を増やすには

1. Amazon DocumentDB コンソールを、次の場所で開きます。https://console.aws.amazon.com/docdb
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. DB インスタンスを選択します。この DB インスタンスに Performance Insights ダッシュボードが表示されます。

   500 バイトを超えるクエリステートメントは、次のように表示されます。



4. クエリ情報セクションを調べて、クエリテキストをさらに表示します。



Performance Insights ダッシュボードは、各フルクエリステートメントで最大 1 KB までを表示できます。

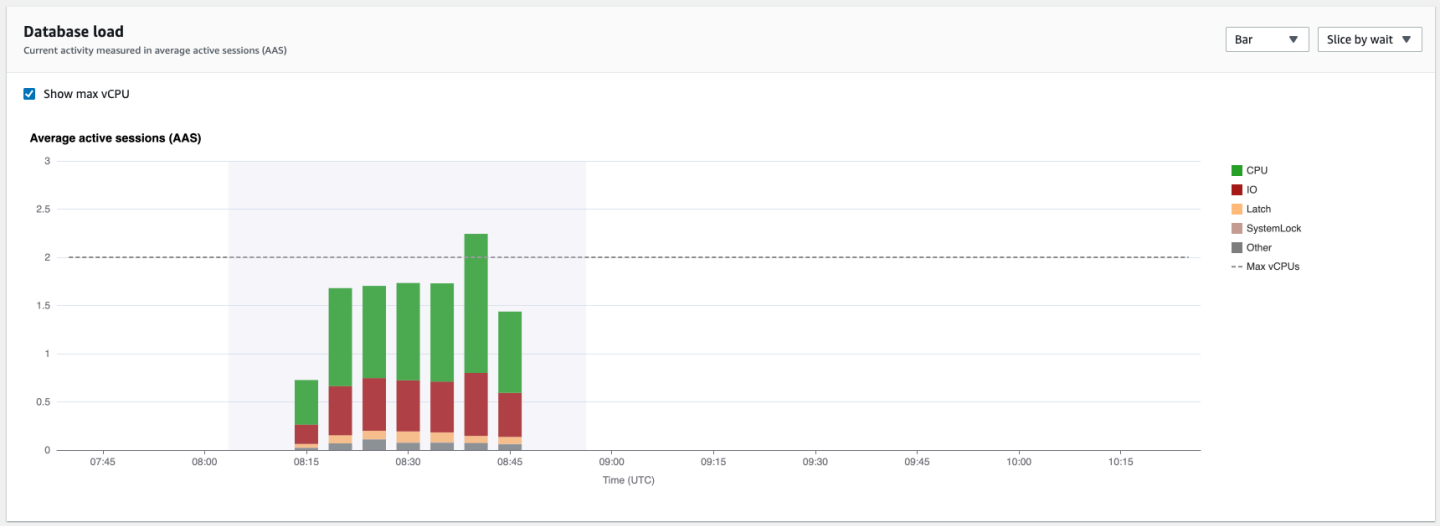
> **Note**
>
> クエリステートメントをコピーまたはダウンロードするには、ポップアップブロッカーを無効にします。

データベースロードグラフのズームイン

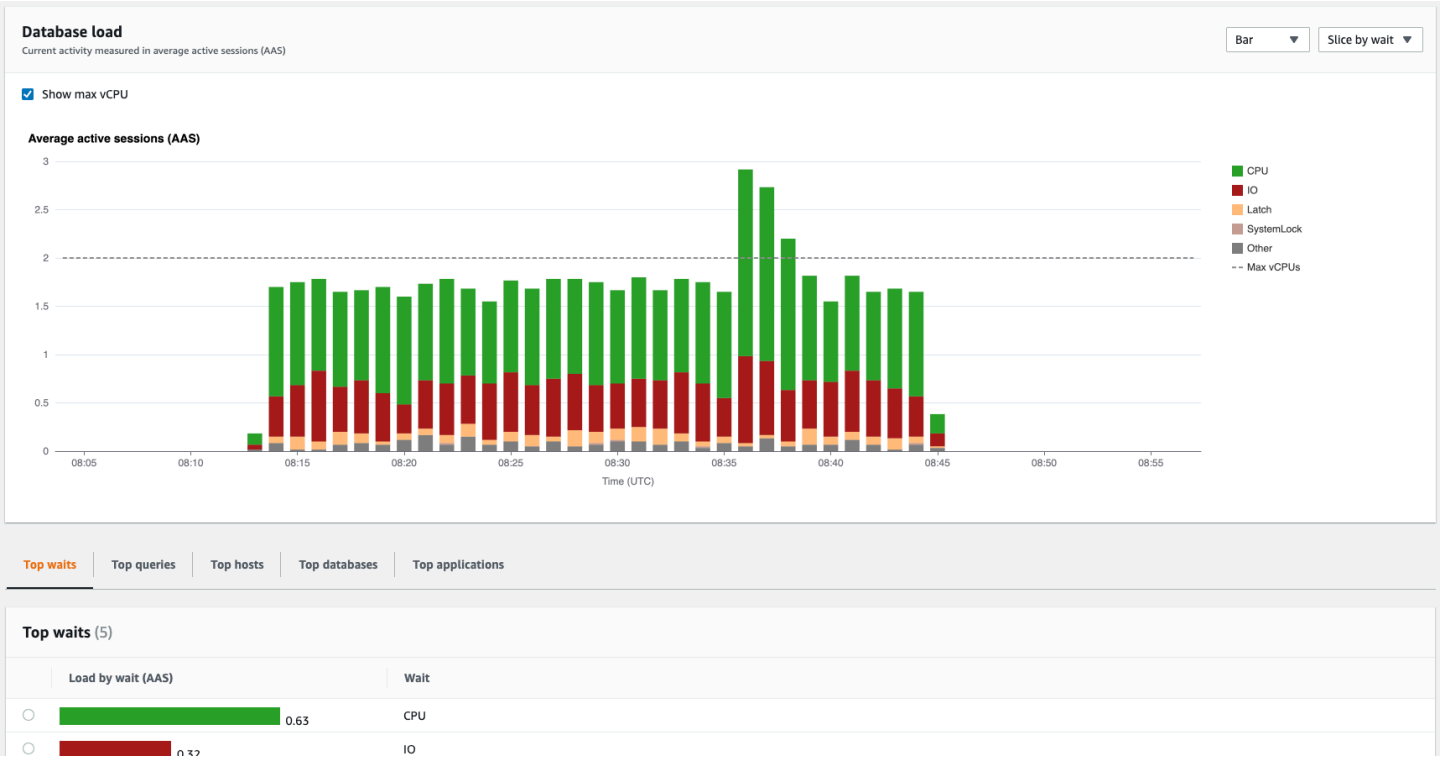
Performance Insights のユーザーインターフェイスの他の機能を使用して、パフォーマンスデータを分析できます。

クリックアンドドラッグしてズームイン

Performance Insights のインターフェイスでは、ロードグラフの小さな部分を選択して、詳細をズームインすることができます。



ロードグラフの一部を拡大するには、スタート時間を選択し、目的の期間の最後までドラッグします。これを行うと、選択した領域が強調表示されます。マウスを離すと、選択したエリアのロードグラフがズームインされ、上位項目のテーブルが再計算されます。



Performance Insights API によるメトリクスの取得

Performance Insights が有効になっている場合、API はインスタンスのパフォーマンスを可視化します。Amazon CloudWatch Logs は、 AWS サービスの提供されたモニタリングメトリクスの信頼できるソースを提供します。

Performance Insightsは、平均アクティブ・セッション(AAS)として測定されるデータベースロードのドメイン固有のビューを提供します。このメトリクスはAPI利用者には2次元時系列データセットのように見えます。データの時間ディメンションは、クエリされた時間範囲内の各時点のDBロード・データを提供します。各時点で、その時点で計測された `Query`、`Wait-state`、`Application`、`Host` などのリクエストされたディメンションに関するロード全体が分解されます。

Amazon DocumentDB Performance Insights では、Amazon DocumentDB DB インスタンスをモニタリングし、データベースパフォーマンスの分析とトラブルシューティングを行うことができます。Performance Insights は、 AWS Management Consoleで表示することができます。また、Performance Insights では独自のデータをクエリできるように、パブリック API も提供されています。API を使用して、次を実行できます。

- データベースにデータをオフロードする
- Performance Insights データを既存のモニタリングダッシュボードに追加する

- モニタリングツールを構築する

Performance Insights API を使用するには、いずれかの Amazon DocumentDB インスタンスで Performance Insights を有効にします。Performance Insights の有効化については、「Performance Insights の有効化と無効化」を参照してください。Performance Insights API の詳細については、「Performance Insights API リファレンス」を参照してください。

Performance Insights API は、以下のオペレーションを提供します。

Performance Insights でのアクション	AWS CLI コマンド	説明
DescribeDimensionKeys	aws pi describe-dimension-keys	特定の期間に、メトリクスの上位 N 個のディメンションキーを取得します。
GetDimensionKeyDetails	aws pi get-dimension-key-details	DB インスタンスまたはデータソースの指定されたディメンショングループの属性を取得します。例えば、クエリ ID を指定し、ディメンションの詳細が使用可能な場合、GetDimensionKeyDetails は、この ID に関連付けられているディメンション db.query.statement の全文を取得します。このオペレーションは、GetResourceMetrics および DescribeDimensionKeys が大きなクエリステートメントテキストの取得をサポートしないため、便利です。
GetResourceMetadata	aws pi get-resource-metadata	さまざまな機能に関するメタデータを取得します。例え

Performance Insights でのアクション	AWS CLI コマンド	説明
		ば、メタデータにより、特定の DB インスタンスで何等かの機能が有効化されているか無効化されているかを、示すことができます。
GetResourceMetrics	aws pi get-resource-metrics	期間中、データソースのセットに Performance Insights のメトリクスを取得します。特定のディメンショングループおよびディメンションを提供し、各グループの集約とフィルタリング条件を提供することができます。
ListAvailableResourceDimensions	aws pi list-available-resource-dimensions	指定したインスタンスで、指定したメトリクスタイプごとにクエリできるディメンションを取得します。
ListAvailableResourceMetrics	aws pi list-available-resource-metrics	DB インスタンスを指定しながら、指定されたメトリクスタイプでクエリが可能なメトリクスをすべて取得します。

トピック

- AWS CLI Performance Insights 用
- 時系列メトリクスの取得
- AWS CLI Performance Insights の例

AWS CLI Performance Insights 用

Performance Insights は、 AWS CLIを使用して表示することができます。Performance Insights の AWS CLI コマンドのヘルプを表示するには、コマンドラインで次のように入力します。

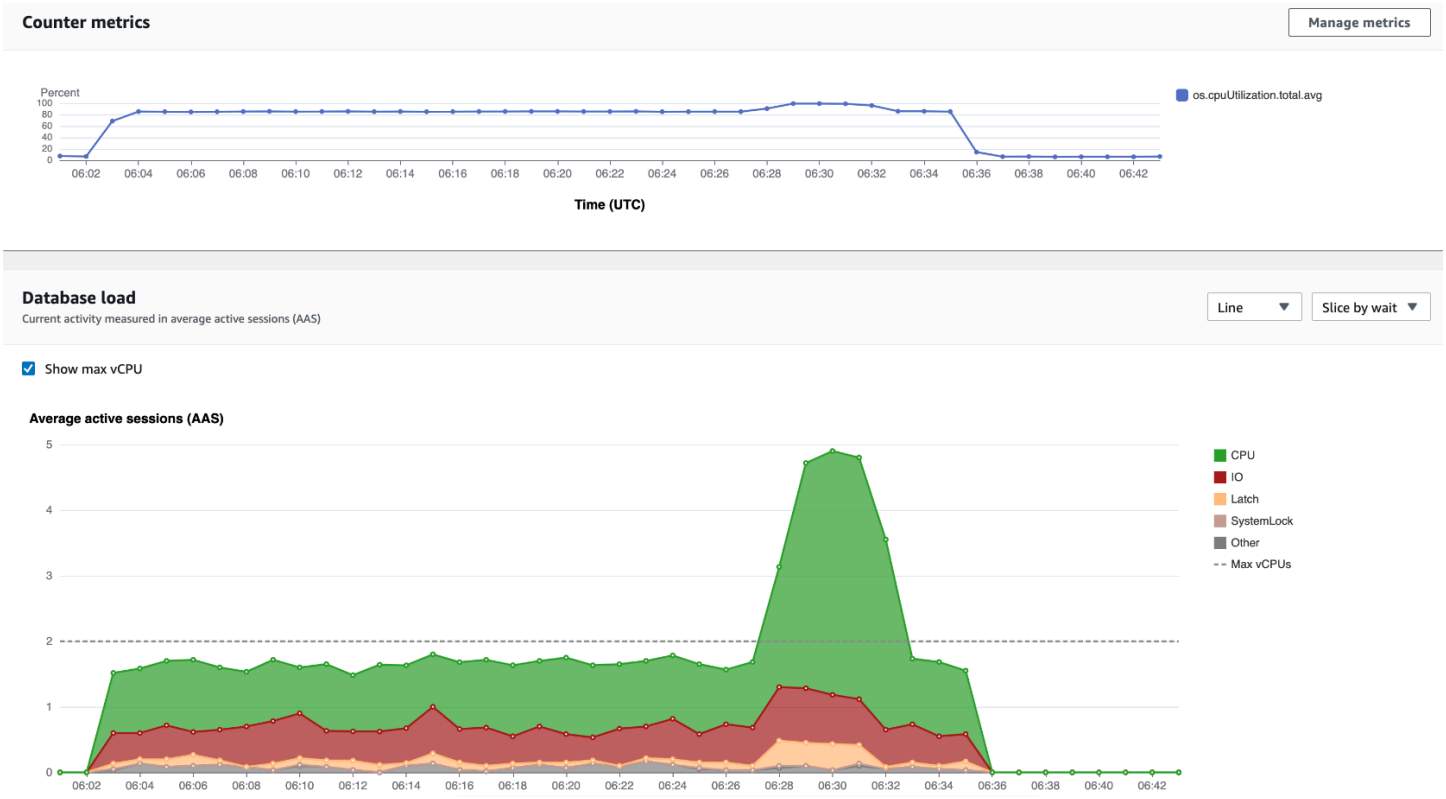
```
aws pi help
```

AWS CLI がインストールされていない場合は、「 ユーザーガイド」の AWS 「 コマンドラインインターフェイスのインストール」を参照してください。 AWS CLI

時系列メトリクスの取得

GetResourceMetrics オペレーションでは、1 つ以上の時系列メトリクスを Performance Insights データから取得します。GetResourceMetrics には、メトリクスおよび期間が必要であり、データポイントのリストを含むレスポンスが返ります。

例えば、次の図に示すように、 AWS Management Console は GetResourceMetricsを使用してカウンターメトリクスグラフとデータベースロードグラフを入力します。



GetResourceMetrics によって返るメトリクスはすべて、db.load の例外を除き、スタンダードの時系列メトリクスです。このメトリクスは、[データベースロード] グラフに表示されます。この db.load メトリクスは、ディメンションと呼ばれるサブコンポーネントに分割できるため、他の時

系列メトリクスとは異なります。前のイメージでは、db.load は分割され、db.load を構成する待機状態によってグループ化されています。

> **Note**
>
> GetResourceMetrics は、db.sampleload メトリクスを返すこともできますが、通常 db.load メトリクスが適切です。

GetResourceMetrics により返されるカウンターメトリクスに関する情報は、「カウンターメトリクス用の Performance Insights」を参照してください。

以下の計算は、メトリクスにサポートされています。

- 平均 - 期間中のメトリクスの平均値。.avg をメトリクス名に追加します。
- 最小 - 期間中のメトリクスの最小値。.min をメトリクス名に追加します。
- 最大 - 期間中のメトリクスの最大値。.max をメトリクス名に追加します。
- 合計 - 期間中のメトリクス値の合計。.sum をメトリクス名に追加します。
- サンプル数 - 期間中にメトリクスが収集された回数。.sample_count をメトリクス名に追加します。

例えば、メトリクスが 300 秒 (5 分) 収集され、メトリクスが 1 分に 1 回収集されたものと見なします。毎分の値は、1、2、3、4、5 です。この場合、以下の計算が返されます。

- 平均 - 3
- 最小 - 1
- 最大 - 5
- 合計 - 15
- サンプル数 - 5

get-resource-metrics AWS CLI コマンドの使用の詳細については、「」を参照してくださいget-resource-metrics。

--metric-queries オプションでは、結果を取得する 1 つ以上のクエリを指定します。各クエリは、必須の Metric と、オプションの GroupBy および Filter パラメータから構成されます。--metric-queries オプションの指定の例を次に示します。

```
{
   "Metric": "string",
   "GroupBy": {
     "Group": "string",
     "Dimensions": ["string", ...],
     "Limit": integer
   },
   "Filter": {"string": "string"
     ...}
```

AWS CLI Performance Insights の例

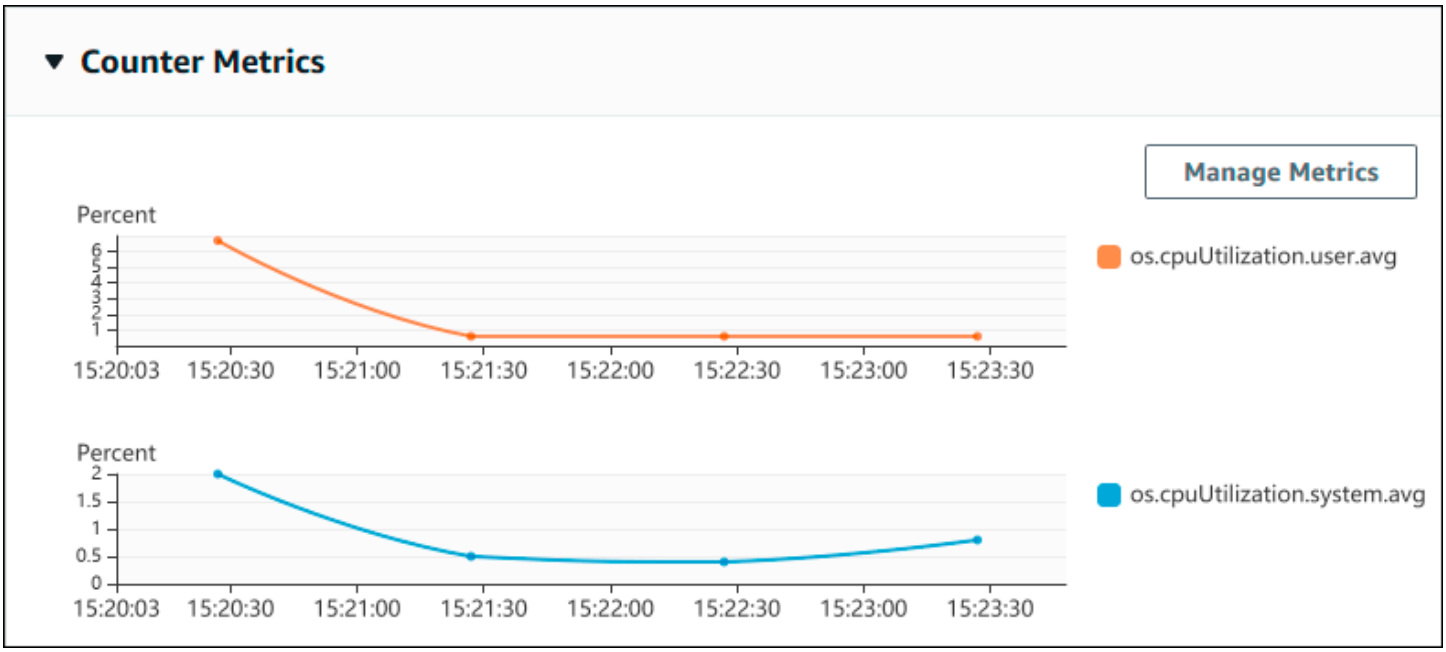
次の例は、Performance Insights AWS CLI に を使用する方法を示しています。

トピック

- カウンターメトリクスの取得
- 上位の待機状態に関する DB 平均ロードの取得
- 上位のクエリに関する DB 平均ロードの取得
- クエリによってフィルタリングされた平均 DB ロードの取得

カウンターメトリクスの取得

以下のスクリーンショットは、 AWS Management Consoleにおける 2 つのカウンターメトリクスグラフを示します。



以下の例では、2 つのカウンターメトリクスグラフを生成するために AWS Management Console で使用するデータと同じデータを生成する方法を示します。

Linux、macOS、Unix の場合:

```
aws pi get-resource-metrics \
   --service-type DOCDB \
   --identifier db-ID \
   --start-time 2022-03-13T8:00:00Z \
   --end-time   2022-03-13T9:00:00Z \
   --period-in-seconds 60 \
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

Windows の場合:

```
aws pi get-resource-metrics ^
   --service-type DOCDB ^
   --identifier db-ID ^
   --start-time 2022-03-13T8:00:00Z ^
   --end-time   2022-03-13T9:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

また、コマンドを作成しやすくするために、`--metrics-query` オプションにファイルを指定します。以下の例では、このオプション用に query.json と呼ばれるファイルを使用します。ファイルの内容は次のとおりです。

```
[
    {
        "Metric": "os.cpuUtilization.user.avg"
    },
    {
        "Metric": "os.cpuUtilization.idle.avg"
    }
]
```

ファイルを使用するには、次のコマンドを実行します。

Linux、macOS、Unix の場合:

```
aws pi get-resource-metrics \
   --service-type DOCDB \
   --identifier db-ID \
   --start-time 2022-03-13T8:00:00Z \
   --end-time   2022-03-13T9:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Windows の場合:

```
aws pi get-resource-metrics ^
   --service-type DOCDB ^
   --identifier db-ID ^
   --start-time 2022-03-13T8:00:00Z ^
   --end-time   2022-03-13T9:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

前述の例では、各オプションに次の値を指定します。

- `--service-type`: Amazon DocumentDB の DOCDB
- `--identifier` - DB インスタンスのリソース ID
- `--start-time` および `--end-time` - クエリを実行する期間の ISO 8601 DateTime 値 (サポートされている複数の形式)

クエリは 1 時間の範囲で実行されます。

- `--period-in-seconds` - 60 (1 分ごとのクエリ)
- `--metric-queries` - 2 つのクエリの配列。それぞれ 1 つのメトリクスに対して使用されます。

  メトリクス名ではドットを使用してメトリクスを有用なカテゴリに分類します。最終の要素は関数になります。この例では、関数は、クエリの avg です。Amazon CloudWatch と同様に、サポートされている関数は、min、max、total、および avg です。

レスポンスは次の例のようになります。

```
{
    "AlignedStartTime": "2022-03-13T08:00:00+00:00",
```

```
    "AlignedEndTime": "2022-03-13T09:00:00+00:00",
    "Identifier": "db-NQF3TTMFQ3GTOKIMJODMC3KQQ4",
    "MetricList": [
        {
            "Key": {
                "Metric": "os.cpuUtilization.user.avg"
            },
            "DataPoints": [
                {
                    "Timestamp": "2022-03-13T08:01:00+00:00", //Minute1
                    "Value": 3.6
                },
                {
                    "Timestamp": "2022-03-13T08:02:00+00:00", //Minute2
                    "Value": 2.6
                },
                //.... 60 datapoints for the os.cpuUtilization.user.avg metric
        {
            "Key": {
                "Metric": "os.cpuUtilization.idle.avg"
            },
            "DataPoints": [
                {
                    "Timestamp": "2022-03-13T08:01:00+00:00",
                    "Value": 92.7
                },
                {
                    "Timestamp": "2022-03-13T08:02:00+00:00",
                    "Value": 93.7
                },
                //.... 60 datapoints for the os.cpuUtilization.user.avg metric
            ]
        }
    ] //end of MetricList
} //end of response
```

レスポンスには、Identifier、AlignedStartTime、AlignedEndTime があります。--period-in-seconds 値が 60 の場合、スタート時間および終了時間は、時間 (分) に調整されます。--period-in-seconds が 3600 の場合、スタート時間および終了時間は、時間 (時) に調整されます。

レスポンスの MetricList には、多数のエントリを含み、それぞれに Key および DataPoints エントリがあります。DataPoint にはそれぞれ、Timestamp および Value を含みます。ク

エリは 1 分ごとのデータが 1 時間以上実行されるため、Datapoints の各リストには、60 個のデータポイントがあります。これには、Timestamp1/Minute1 や Timestamp2/Minute2 から、Timestamp60/Minute60 まで含まれます。

クエリは 2 つの異なるカウンターメトリクスを対象としているため、レスポンス MetricList には 2 つの要素があります。

上位の待機状態に関する DB 平均ロードの取得

次の例は、 が積み上げ面折れ線グラフの生成 AWS Management Console に使用するクエリと同じです。この例では、上位 7 つの待機状態に応じてロードを分割し、最後の 1 時間で db.load.avg を取得します。コマンドは カウンターメトリクスの取得 と同じコマンドです。ただし、query.json ファイルには、次の内容が含まれます。

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 7 }
    }
]
```

次のコマンドを実行します。

Linux、macOS、Unix の場合:

```
aws pi get-resource-metrics \
   --service-type DOCDB \
   --identifier db-ID \
   --start-time 2022-03-13T8:00:00Z \
   --end-time   2022-03-13T9:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Windows の場合:

```
aws pi get-resource-metrics ^
   --service-type DOCDB ^
   --identifier db-ID ^
   --start-time 2022-03-13T8:00:00Z ^
   --end-time   2022-03-13T9:00:00Z ^
   --period-in-seconds 60 ^
```

```
    --metric-queries file://query.json
```

この例では、上位 7 つの待機状態のうち db.load.avg と GroupBy のメトリクスを指定しています。この例の有効な値の詳細については、Performance Insights の API リファレンスの「DimensionGroup」を参照してください。

レスポンスは次の例のようになります。

```
{
    "AlignedStartTime": "2022-04-04T06:00:00+00:00",
    "AlignedEndTime": "2022-04-04T06:15:00+00:00",
    "Identifier": "db-NQF3TTMFQ3GTOKIMJODMC3KQQ4",
    "MetricList": [
        {//A list of key/datapoints
            "Key": {
                //A Metric with no dimensions. This is the total db.load.avg
                "Metric": "db.load.avg"
            },
            "DataPoints": [
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": "2022-04-04T06:01:00+00:00",//Minute1
                    "Value": 0.0
                },
                {
                    "Timestamp": "2022-04-04T06:02:00+00:00",//Minute2
                    "Value": 0.0
                },
                //... 60 datapoints for the total db.load.avg key
                ]
        },
        {
            "Key": {
                //Another key. This is db.load.avg broken down by CPU
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_state.name": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": "2022-04-04T06:01:00+00:00",//Minute1
```

```
                    "Value": 0.0
                },
                {
                    "Timestamp": "2022-04-04T06:02:00+00:00",//Minute2
                    "Value": 0.0
                },
                //... 60 datapoints for the CPU key
            ]
        },//... In total we have 3 key/datapoints entries, 1) total, 2-3) Top Wait
 States
    ] //end of MetricList
} //end of response
```

このレスポンスでは、MetricList の 3 つのエントリがあります。合計の db.load.avg のエントリが 1 つあり、上位 3 つの待機イベントのいずれかに従って分割された db.load.avg のエントリが 3 つあります。(最初の例とは異なり) グループ化ディメンションがあったため、メトリクスのグループ化ごとに 1 つのキーが必要です。基本的なカウンターメトリクスのユースケースのように、メトリクスごとに 1 つのキーのみ使用することはできません。

上位のクエリに関する DB 平均ロードの取得

以下の例では、上位 10 個のクエリステートメント別に db.wait_state をグループ化します。クエリステートメントには 2 つの異なるグループがあります。

- db.query - フルクエリステートメント (例: {"find":"customers","filter":{"FirstName":"Jesse"},"sort":{"key":{"$numberInt":"1"}}})
- db.query_tokenized - トークン化されたクエリステートメント (例: {"find":"customers","filter":{"FirstName":"?"},"sort":{"key":{"$numberInt":"?"}},"limit":{"$numberInt":"?"}})

データベースのパフォーマンスを分析するときは、パラメータが異なるだけのクエリステートメントを 1 つの論理的な項目として検討すると便利です。そのため、クエリを実行する際、db.query_tokenized を使用することができます。ただし、特に explain( ) に関心がある場合は、パラメータ付きのフルクエリステートメントを調べる方が便利な場合があります。トークン化されたクエリと完全クエリの間には親子関係があり、複数の完全クエリ (子) が同じトークン化されたクエリ (親) の下にグループ化されています。

この例のコマンドは、上位の待機状態に関する DB 平均ロードの取得 のコマンドに似ています。ただし、query.json ファイルには、次の内容が含まれます。

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.query_tokenized", "Limit": 10 }
    }
]
```

次の例では db.query_tokenized を使用しています。

Linux、macOS、Unix の場合:

```
aws pi get-resource-metrics \
   --service-type DOCDB \
   --identifier db-ID \
   --start-time 2022-03-13T8:00:00Z \
   --end-time   2022-03-13T9:00:00Z \
   --period-in-seconds 3600 \
   --metric-queries file://query.json
```

Windows の場合:

```
aws pi get-resource-metrics ^
   --service-type DOCDB ^
   --identifier db-ID ^
   --start-time 2022-03-13T8:00:00Z ^
   --end-time   2022-03-13T9:00:00Z  ^
   --period-in-seconds 3600 ^
   --metric-queries file://query.json
```

この例では、1 分の間隔 (秒単位) で 1 時間以上のクエリを実行します。

この例では、上位 7 つの待機状態のうち db.load.avg と GroupBy のメトリクスを指定しています。この例の有効な値の詳細については、Performance Insights の API リファレンスの「DimensionGroup」を参照してください。

レスポンスは次の例のようになります。

```
{
    "AlignedStartTime": "2022-04-04T06:00:00+00:00",
    "AlignedEndTime": "2022-04-04T06:15:00+00:00",
    "Identifier": "db-NQF3TTMFQ3GTOKIMJODMC3KQQ4",
    "MetricList": [
```

```
        {//A list of key/datapoints
            "Key": {
                "Metric": "db.load.avg"
            },
            "DataPoints": [
                //... 60 datapoints for the total db.load.avg key
                ]
        },
               {
            "Key": {//Next key are the top tokenized queries
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.query_tokenized.db_id": "pi-1064184600",
                    "db.query_tokenized.id": "77DE8364594EXAMPLE",
                    "db.query_tokenized.statement": "{\"find\":\"customers\",\"filter
\":{\"FirstName\":\"?\"},\"sort\":{\"key\":{\"$numberInt\":\"?\"}},\"limit\"
:{\"$numberInt\":\"?\"},\"$db\":\"myDB\",\"$readPreference\":{\"mode\":\"primary\"}}"
                }
            },
            "DataPoints": [
            //... 60 datapoints
            ]
        },
        // In total 11 entries, 10 Keys of top tokenized queries, 1 total key
    ] //End of MetricList
} //End of response
```

このレスポンスの MetricList には 11 のエントリがあり (合計が 1 つと、トークン化された上位 10 項目のクエリ)、各エントリには、1 時間あたり 24 の DataPoints があります。

トークン化されたクエリの場合は、各ディメンションリストに 3 つのエントリがあります。

- db.query_tokenized.statement: トークン化されたクエリステートメント。
- db.query_tokenized.db_id : Performance Insights が生成する合成 ID。この例では、pi-1064184600 合成 ID が返ります。
- db.query_tokenized.id - Performance Insights 内のクエリの ID。

  では AWS Management Console、この ID はサポート ID と呼ばれます。ID は、データベースの問題のトラブルシューティングに役立つように AWS サポートが調査できるデータであるため、この名前が付けられます。 はデータのセキュリティとプライバシーを非常に真剣に AWS 受け止め、ほぼすべてのデータが で暗号化されて保存されます AWS KMS key。したがって、内

部の誰もこのデータを見る AWS ことはできません。前の例では、tokenized.statement と tokenized.db_id の両方が暗号化されて保存されます。データベースに問題がある場合は、AWS サポート ID を参照してサポートにお問い合わせください。

クエリを実行する際、Group で GroupBy を指定した方が便利な場合があります。ただし、返るデータを詳細に制御できるように、ディメンションのリストを指定します。例えば、必要なデータが db.query_tokenized.statement のみの場合は、Dimensions 属性を query.json ファイルに追加することができます。

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.query_tokenized",
            "Dimensions":["db.query_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

クエリによってフィルタリングされた平均 DB ロードの取得

この例に対応する API クエリは、上位のクエリに関する DB 平均ロードの取得 のコマンドに似ています。ただし、query.json ファイルには、次の内容が含まれます。

```
[
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 5  },
        "Filter": { "db.query_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
]
```

このレスポンスでは、query.json ファイルで指定されているトークン化されたクエリ AKIAIOSFODNN7EXAMPLE の割合に従って、値はすべてフィルタリングされます。キーは、フィルタなしのクエリとは異なる順序で表示されることもあります。これは、フィルタ処理されたクエリに影響を与えるのは上位 5 つの待機クエリであるためです。

Performance Insights の Amazon CloudWatch メトリクス

Performance Insights はメトリクスを自動的に Amazon CloudWatch に発行します。Performance Insights から同じデータに対してクエリを実行できますが、CloudWatch にメトリクスを含めると、CloudWatch アラームを追加しやすくなります。また、既存の CloudWatch ダッシュボードにメトリクスを追加しやすくなります。

メトリクス	説明
DBLoad	Amazon DocumentDB のアクティブセッション数。通常、アクティブセッションの平均数に関するデータを使用します。Performance Insights で、このデータは `db.load.avg` としてクエリされます。
DBLoadCPU	待機状態タイプが CPU であるアクティブセッションの数。Performance Insights で、このデータは、待機状態タイプ `db.load.avg` でフィルタ処理された `CPU` としてクエリされます。
DBLoadNonCPU	待機状態タイプが CPU でないアクティブセッションの数。

> **Note**
>
> これらのメトリクスは、DB インスタンスにロードがある場合にのみ CloudWatch に公開されます。

これらのメトリクスは、CloudWatch コンソール、 AWS CLI、または CloudWatch API を使用して調べることができます。

例えば、`DBLoad` メトリクスの統計情報は、get-metric-statistics コマンドを実行して取得できます。

```
aws cloudwatch get-metric-statistics \
    --region ap-south-1 \
    --namespace AWS/DocDB \
```

```
    --metric-name DBLoad  \
    --period 360 \
    --statistics Average \
    --start-time 2022-03-14T8:00:00Z \
    --end-time 2022-03-14T9:00:00Z \
    --dimensions Name=DBInstanceIdentifier,Value=documentdbinstance
```

次のコマンドでは、以下のような出力が生成されます。

```
{
    "Datapoints": [
        {
            "Timestamp": "2022-03-14T08:42:00Z",
            "Average": 1.0,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:24:00Z",
            "Average": 2.0,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:54:00Z",
            "Average": 6.0,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:36:00Z",
            "Average": 5.7,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:06:00Z",
            "Average": 4.0,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:00:00Z",
            "Average": 5.2,
            "Unit": "None"
        }
    ],
    "Label": "DBLoad"
```

```
}
```

CloudWatch コンソールの DB_PERF_INSIGHTS メトリクス数学関数を使用して Amazon DocumentDB にクエリを実行し、Performance Insights カウンターメトリクスを取得できます。DB_PERF_INSIGHTS 関数には、1 分未満の間隔での DBLoad メトリクスも含まれます。これらのメトリクスに基づいた CloudWatch アラームを設定することができます。アラームの作成方法の詳細については、AWS 「データベースから Performance Insights カウンターメトリクスにアラームを作成する」を参照してください。

CloudWatch の詳細については、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch とは」を参照してください。

カウンターメトリクス用の Performance Insights

カウンターメトリクスは、Performance Insights ダッシュボードのオペレーティングシステムメトリクスのことです。カウンターメトリクスを DB ロードと関連付けることで、パフォーマンスの問題を特定して分析できます。

Performance Insights オペレーティングシステムのカウンター

Amazon DocumentDB のパフォーマンスインサイトでは、以下のオペレーティングシステムカウンターを利用できます。

Counter	タイプ	メトリクス
active	メモリ	os.memory.active
buffers	メモリ	os.memory.buffers
キャッシュ済み	メモリ	os.memory.cached
dirty	メモリ	os.memory.dirty
free	メモリ	os.memory.free
無効	メモリ	os.memory.inactive
mapped	メモリ	os.memory.mapped

Counter	タイプ	メトリクス
pageTables	メモリ	os.memory.pageTables
slab	メモリ	os.memory.slab
total	メモリ	os.memory.total
writeback	メモリ	os.memory.writeback
idle	cpuUtilization	os.cpuUtilization.idle
system	cpuUtilization	os.cpuUtilization.system
total	cpuUtilization	os.cpuUtilization.total
ユーザー	cpuUtilization	os.cpuUtilization.user
によって	cpuUtilization	os.cpuUtilization.wait
one	loadAverageMinute	os.loadAverageMinute.one
fifteen	loadAverageMinute	os.loadAverageMinute.fifteen
5	loadAverageMinute	os.loadAverageMinute.five
キャッシュ済み	スワップ	os.swap.cached
free	スワップ	os.swap.free
が	スワップ	os.swap.in
out	スワップ	os.swap.out
total	スワップ	os.swap.total
rx	ネットワーク	os.network.rx
tx	ネットワーク	os.network.tx
numVCPUs	全般	os.general.numVCPUs

Amazon OpenSearch Service とのゼロ ETL 統合

トピック

- 送信先としての Amazon OpenSearch Service。
- 制限

送信先としての Amazon OpenSearch Service。

OpenSearch Service と Amazon DocumentDB の統合により、フルロードやデータ変更イベントを OpenSearch ドメインにストリーミングすることができます。取り込みインフラストラクチャは OpenSearch 取り込みパイプラインとしてホストされ、Amazon DocumentDB コレクションから継続的にデータをストリーミングするための、高スケールで低レイテンシーなメカニズムを提供します。

フルロード中、ゼロ ETL 統合はまず、取り込みパイプラインを使用してフルロードデータ履歴を OpenSearch に抽出します。フルロードデータが取り込まれると、OpenSearch 取り込みパイプラインは Amazon DocumentDB 変更ストリームからのデータの読み取りを開始し、最終的にすべての変更を反映させて、Amazon DocumentDB と OpenSearch 間のほぼリアルタイムのデータ整合性を維持します。OpenSearch は、ドキュメントをインデックスに保存します。Amazon DocumentDB コレクションからの受信データは、1 つのインデックスに送信することも、異なるインデックスに分割することもできます。取り込みパイプラインは、Amazon DocumentDB コレクション内のすべての作成、更新、削除イベントを OpenSearch ドキュメントの対応する作成、更新、削除イベントと同期させ、2 つのデータシステムの同期を維持します。取り込みパイプラインは、1 つのコレクションからデータを読み取って 1 つのインデックスに書き込むように設定することも、あるいは 1 つのコレクションからデータを読み取って条件に応じて複数のインデックス間でルーティングするように設定することもできます。

取り込みパイプラインは、以下のモードで Amazon DocumentDB から Amazon OpenSearch Service にデータをストリーミングするように設定できます。

- フルロードのみ
- フルロードなしで Amazon DocumentDB から変更ストリームイベントのみをストリーミングする
- Amazon DocumentDB からまずフルロードを実施し、次に変更ストリームをストリーミングする

取り込みパイプラインを設定するには、次の手順を実行します。

ステップ 1: Amazon OpenSearch Service ドメインまたは OpenSearch サーバーレスコレクションを作成する

データを読み取るための適切なアクセス許可を持つ Amazon OpenSearch Service コレクションが必要です。「Amazon OpenSearch Service 開発者ガイド」の「Amazon OpenSearch Service の使用開始」または「Amazon OpenSearch Serverless の使用開始」でコレクションの作成方法を確認してください。また、「Amazon OpenSearch Service 開発者ガイド」の「Amazon OpenSearch によるデータ取り込み」を参照して、コレクションまたはドメインへの正しい書き込みアクセス許可を取得するための AIM ロールの作成方法を確認してください。

ステップ 2: Amazon DocumentDB クラスターで変更ストリームを有効にする

Amazon DocumentDB クラスター内の必要なコレクションで変更ストリームが有効になっていることを確認します。詳細については、「Amazon DocumentDB を用いた変更ストリームの使用」を参照してください。

ステップ 3: Amazon S3 バケットと送信先のドメインまたはコレクションへの書き込みアクセス許可を持つパイプラインロールを設定する

Amazon DocumentDB コレクションを作成して変更ストリームを有効にしたら、パイプライン設定に使用するパイプラインロールを設定し、次のアクセス許可をロールに付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "allowReadAndWriteToS3ForExport",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::my-bucket/export/*"
            ]
        }
    ]
}
```

OpenSearch パイプラインが OpenSearch ドメインにデータを書き込めるようにするためには、当該ドメインが、sts_role_arn パイプラインロールにドメインへのアクセスを許可するドメインレベルのアクセスポリシーを持っている必要があります。次のドメインアクセスポリシーの例では、前のステップで作成した pipeline-role と言う名前のパイプラインロールに、ingestion-domain と言う名前のドメインへの、データの書き込みが許可されています。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::{your-account-id}:role/{pipeline-role}"
      },
      "Action": ["es:DescribeDomain", "es:ESHttp*"],
      "Resource": "arn:aws:es:{region}:{your-account-id}:domain/{domain-name}/*"
    }
  ]
}
```

ステップ 4: X-ENI を作成するために必要なアクセス許可をパイプラインロールに追加する

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
```

```
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeNetworkInterfaces"
            ],
            "Resource": [
                "arn:aws:ec2:*:420497401461:network-interface/*",
                "arn:aws:ec2:*:420497401461:subnet/*",
                "arn:aws:ec2:*:420497401461:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [ "ec2:CreateTags" ],
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
               "StringEquals": { "aws:RequestTag/OSISManaged": "true" }
            }
        }
    ]
}
```

ステップ 5: パイプラインを作成する

Amazon DocumentDB をソースとして指定する OpenSearch 取り込みパイプラインを設定します。このサンプルパイプライン設定は、変更ストリームフェッチメカニズムの使用を前提としています。詳細については、「Amazon OpenSearch Service 開発者ガイド」の「Amazon DocumentDB での OpenSearch 取り込みパイプラインの使用」を参照してください。

制限

Amazon DocumentDB と OpenSearch の統合には、次の制限事項が適用されます。

- パイプラインごとにソースとしてサポートする Amazon DocumentDB コレクションは 1 つのみとなります。
- リージョン間のデータインジェストは、サポートされていません。Amazon DocumentDB クラスターと OpenSearch ドメインは同じ AWS リージョンに存在する必要があります。
- 複数アカウント間のデータインジェストはサポートされていません。Amazon DocumentDB クラスターと OpenSearch 取り込みパイプラインは、同じ AWS アカウントに存在する必要があります。
- Amazon DocumentDB Elastic クラスターはサポートされていません。Amazon DocumentDB インスタンスベースのクラスターのみがサポートされています。
- Amazon DocumentDB クラスターで AWS secrets. AWS secrets を使用した認証が有効になっていることが、サポートされている唯一の認証メカニズムであることを確認します。
- 既存のパイプライン設定を更新して、別のデータベースや別のコレクションからデータを取り込むことはできません。パイプラインのデータベース名やコレクション名を更新するには、新しいパイプラインを作成する必要があります。

Amazon DocumentDB サーバーレスの使用

Amazon DocumentDB サーバーレスは、アプリケーションの需要に基づいて Amazon DocumentDB データベースの容量を動的に調整するプロセスを自動化するオンデマンドの自動スケーリング設定です。クラスターが消費するリソースに対してのみ課金されます。したがって、DocumentDB サーバーレスは予算内に収まり、使用しないコンピュータリソースへの支払いを避けるのに役立ちます。

このタイプの自動化は、マルチテナントデータベース、分散データベース、開発およびテストシステムなど、変動性が高く、予測不可能なワークロードがある環境では特に重要です。

トピック

- DocumentDB サーバーレスのユースケース
- Amazon DocumentDB サーバーレスの利点
- Amazon DocumentDB サーバーレスの仕組み
- DocumentDB サーバーレスの要件と制限
- Amazon DocumentDB サーバーレスを使用するクラスターの作成
- Amazon DocumentDB サーバーレスへの移行
- Amazon DocumentDB サーバーレスの管理
- Amazon DocumentDB サーバーレスインスタンスの制限
- Amazon DocumentDB サーバーレススケーリング設定
- Amazon DocumentDB サーバーレスのモニタリング

DocumentDB サーバーレスのユースケース

Amazon DocumentDB プロビジョニングクラスターと DocumentDB サーバーレスはどちらも、開発環境やテスト環境から、高いスケールと可用性を必要とする最も要求の厳しいビジネスクリティカルなアプリケーションまで、さまざまなタイプのデータベースワークロードをサポートします。ただし、DocumentDB サーバーレスは、顧客のワークロードに別のディメンションを追加します。つまり、予測不可能なワークロードを持つウェブサイトやアプリケーションをサポートする機能です。

DocumentDB サーバーレスは、次のユースケースで特に役立ちます。

- 可変ワークロード — アクティビティが突然増加し、予測不可能なワークロードを実行しています。雨が降り出したときにアクティビティが急増 (サージ) するトラフィックサイトなどが該当します。別のケースとして、販売や特別なプロモーションを行うことで、トラフィックが増加する

e コマースサイトがあります。DocumentDB サーバーレスを使用すると、データベースはアプリケーションのピーク負荷のニーズに合わせて容量を自動的にスケールし、アクティビティの急増が終わったらスケールダウンします。DocumentDB サーバーレスでは、ピーク容量または平均容量をプロビジョニングする必要がなくなりました。最悪の状況に対応するために容量の上限を指定でき、その容量は必要な場合以外使用されません。

  - DocumentDB サーバーレスでのスケーリングの粒度は、容量をデータベースのニーズに厳密に合わせるのに役立ちます。プロビジョニングされたクラスターの場合、スケールアップするには、まったく新しいインスタンスを追加する必要があります。DocumentDB サーバーレスは、必要な容量がわずかに増えるだけで、DCU の半分を追加できます。ワークロードの増加を処理するために必要な追加容量に基づいて、0.5、1、1.5、2、または追加の半 DCUs を追加できます。また、ワークロードが減少し、その容量が不要になったときに、0.5、1、1.5、2、または追加の半 DCUs を削除できます。

- マルチテナントアプリケーション — DocumentDB サーバーレスでは、フリート内の各アプリケーションのデータベース容量を個別に管理する必要はありません。DocumentDB サーバーレスは、個々のデータベース容量を管理します。

  - テナントごとにクラスターを作成できます。これにより、クローン作成やスナップショット復元などの機能を使用して、テナントごとに必要に応じて高可用性とディザスタリカバリを強化できます。

  - 各テナントには、時間帯、時期、プロモーションイベントなどに応じて、繁忙期と休止期間が設定される場合があります。各クラスターには、広い範囲で容量を指定できます。これにより、アクティビティの少ないクラスターでは、インスタンス料金が最小限に抑えられます。どのクラスターでも、アクティビティの高い期間に対応できるように迅速にスケールアップできます。

- 新しいアプリケーション — 新しいアプリケーションをデプロイしていて、必要なインスタンスサイズが不明な場合。DocumentDB サーバーレスを使用すると、1 つ以上のインスタンスでクラスターを設定し、データベースをアプリケーションの容量要件に合わせて自動スケーリングできます。

- 混合用途アプリケーション — オンライントランザクション処理 (OLTP) アプリケーションがあるが、クエリトラフィックが定期的に急増するとします。クラスター内の DocumentDB サーバーレスインスタンスの昇格階層を指定することで、リーダーインスタンスがライターインスタンスとは独立してスケーリングして追加の負荷を処理できるようにクラスターを設定できます。使用量のスパイクが減ると、リーダーインスタンスはライターインスタンスの容量に合わせてスケールダウンします。

- キャパシティプランニング — 通常、クラスター内のすべてのインスタンスのインスタンスクラスを変更して、データベース容量を調整するか、ワークロードに最適なデータベース容量を検証するとします。DocumentDB サーバーレスを使用すると、この管理オーバーヘッドを回避できます。

ワークロードを実行し、インスタンスの実際のスケーリング量を確認することで、適切な最小容量と最大容量を決定できます。

  - 既存のインスタンスは、プロビジョニングされたインスタンスから DocumentDB サーバーレスに、または DocumentDB サーバーレスからプロビジョニングされたインスタンスに変更できます。このような場合は、新しいクラスターや新しいインスタンスを作成する必要はありません。

- 開発とテスト — 最も要求の厳しいアプリケーションを実行するだけでなく、開発環境とテスト環境に DocumentDB サーバーレスを使用することもできます。DocumentDB サーバーレスを使用すると、バースト可能な db.t* インスタンスクラスを使用する代わりに、最小容量の低いインスタンスを作成できます。最大容量を十分高く設定することで、それらのインスタンスはメモリ不足することなく大量のワークロードを引き続き実行できます。データベースが使用されていない場合、不要な料金が発生しないように、すべてのインスタンスがスケールダウンされます。

既存のプロビジョニングされたワークロードに Amazon DocumentDB サーバーレスを使用する

プロビジョニングされたクラスターで実行されている DocumentDB アプリケーションが既にあるとします。1 つ以上の DocumentDB サーバーレスインスタンスをリーダーインスタンスとして既存のクラスターに追加することで、アプリケーションが DocumentDB サーバーレスとどのように連携するかを確認できます。リーダーインスタンスのスケールアップとスケールダウンの頻度を確認できます。DocumentDB フェイルオーバーメカニズムを使用して、DocumentDB サーバーレスインスタンスをライターに昇格させ、読み取り/書き込みワークロードの処理方法を確認できます。これにより、クライアントアプリケーションが使用するエンドポイントを変更することなく、最小限のダウンタイムで切り替えが可能です。既存のクラスターを DocumentDB サーバーレスに変換する手順の詳細については、「」を参照してくださいAmazon DocumentDB サーバーレスへの移行。

Amazon DocumentDB サーバーレスの利点

DocumentDB サーバーレスは、可変ワークロードまたは「スパイク」ワークロードを対象としています。このような予測不可能なワークロードでは、データベース容量を変更するタイミングを計画するのが難しい場合があります。また、インスタンスの追加やインスタンスクラスの変更など、使い慣れたメカニズムを使用して、容量をすばやく変更できない場合があります。DocumentDB サーバーレスには、このようなユースケースに役立つ以下の利点があります。

- プロビジョニングよりもシンプルなキャパシティ管理 — DocumentDB サーバーレスは、ワークロードの変化に応じてインスタンスサイズを計画し、インスタンスのサイズを変更する労力を削減

します。また、クラスター内のすべてのインスタンスで一貫した容量を維持するための労力も削減されます。

- アクティビティの多い期間におけるスケーリングの高速化と簡易化 — DocumentDB サーバーレスは、クライアントトランザクションや全体的なワークロードを中断することなく、必要に応じてコンピューティング容量とメモリ容量をスケーリングします。DocumentDB サーバーレスでリーダーインスタンスを使用する機能は、垂直スケーリングに加えて水平スケーリングを活用するのに役立ちます。

- アクティビティが少ない期間におけるコスト効率 — DocumentDB サーバーレスは、インスタンスのオーバープロビジョニングを回避するのに役立ちます。DocumentDB サーバーレスは、インスタンスのスケールアップ時にリソースをきめ細かく追加します。料金は、使用したデータベースリソースに対してのみ発生します。DocumentDB サーバーレスリソースの使用量は、秒単位で測定されます。これにより、インスタンスがスケールダウンすると、リソース使用量の減少がすぐに登録されます。

- プロビジョニングされた との機能パリティ — DocumentDB サーバーレスですべての DocumentDB 機能を使用できます。例えば、DocumentDB サーバーレスでは、リーダーインスタンス、 AWS Identity and Access Management (IAM) データベース認証、Performance Insights を使用できます。

  特に、DocumentDB サーバーレスでは、プロビジョニングされたクラスターから次の機能を利用できます。

  - リーダーインスタンス — DocumentDB サーバーレスはリーダーインスタンスを活用して水平方向にスケーリングできます。クラスターに 1 つ以上のリーダーインスタンスが含まれている場合、ライターインスタンスに問題がある場合、クラスターはすぐにフェイルオーバーする可能性があります。
  - マルチ AZ クラスター — クラスターの DocumentDB サーバーレスインスタンスを複数のアベイラビリティーゾーン (AZs) に分散できます。マルチ AZ クラスターを設定することで、AZ 全体に影響する問題が発生するようなまれなケースでも、ビジネスの継続性を確保できます。

Amazon DocumentDB サーバーレスの仕組み

トピック

- 概要

- Amazon DocumentDB クラスターの設定
- Amazon DocumentDB サーバーレススケーリング容量
- Amazon DocumentDB サーバーレススケーリング
- アイドル状態 (0.5 DCUs)

概要

Amazon DocumentDB サーバーレスは、最も要求の厳しい、変動性の高いワークロードに適しています。用途の例としては、データベースの使用負荷が短時間の間だけ増大し、その後に軽いアクティビティが長時間続くか、またはアクティビティがまったく発生しなくなるケースが挙げられます。例えば、定期的に販売促進イベントを行う小売り、ゲーム、スポーツなどのウェブサイト、必要なときにレポートを作成するレポートデータベースなどがあります。また、開発やテスト環境、また、新しいアプリケーションでは急激に利用が増加することがあります。他にも多くが考えられますが、このようなケースに対してプロビジョニングされたモデルを使用しても、事前に容量を正しく指定できるとは限りません。また、過剰なプロビジョニングを行い、使用しない容量が生じた場合には、コストが高くなる可能性もあります。

対照的に、DocumentDB でプロビジョニングされたクラスターは安定したワークロードに適しています。プロビジョニングされたクラスターでは、事前定義された量のメモリ、CPU 電源、I/O 帯域幅などを持つインスタンスクラスを選択します。ワークロードが変更された場合、ライターとリーダーのインスタンスクラスを手動で変更します。プロビジョン済みモデルは、消費パターンが予想され、事前に容量を調整できる場合に有効です。クラスター内のライターとリーダーのインスタンスクラスを変更しながら、短時間の停止の発生が許容される場合は有効に機能します。

DocumentDB サーバーレスは、すぐにスケーラブルなサーバーレスクラスターをサポートするためにゼロから設計されています。DocumentDB サーバーレスは、プロビジョニングされたライターやリーダーと同じレベルのセキュリティと分離を提供するように設計されています。このような側面は、マルチテナントのサーバーレスクラウド環境では非常に重要です。データベースのワークロードの変更に迅速に対応できるように、動的スケーリングメカニズムにはオーバーヘッドがほとんどありません。また、処理需要の劇的な増加に対応するための、十分な能力も備わっています。

DocumentDB サーバーレスを使用すると、ライターとリーダーごとに特定のデータベース容量にロックされることなく、DocumentDB クラスターを作成できます。お客様は、最小容量と最大容量の範囲を指定します。DocumentDB は、その容量範囲内のクラスター内の各 DocumentDB サーバーレスライターまたはリーダーをスケーリングします。各ライターまたはリーダーによって動的にスケーリングできるマルチ AZ クラスターを使用することで、動的スケーリングと高可用性を活用できます。

DocumentDB サーバーレスは、最小容量と最大容量の仕様に基づいてデータベースリソースを自動的にスケーリングします。ほとんどのスケーリングイベントのオペレーションは、ライターまたはリーダーが同じホスト上で保持されるため、高速にスケーリングします。まれにDocumentDB サーバーレスライターまたはリーダーがホスト間で移動された場合、DocumentDB サーバーレスは接続を自動的に管理します。データベースクライアントアプリケーションのコードやデータベースの接続文字列を変更する必要はありません。

DocumentDB サーバーレスでは、プロビジョニングされたクラスターと同様に、ストレージ容量とコンピューティング容量は別々になります。DocumentDB サーバーレスキャパシティとスケーリングを参照すると、常にコンピューティングキャパシティが増減します。したがって、CPU やメモリの容量がスケールダウンしても、クラスターには数テラバイトのデータを格納できます。

プロビジョニングやデータベースサーバーを管理する代わりに、データベース容量を指定します。各DocumentDB サーバーレスライターまたはリーダーの実際の容量は、ワークロードに応じて時間の経過とともに変化します。このメカニズムの詳細については、「Amazon DocumentDB サーバーレススケーリング」を参照してください。

Amazon DocumentDB クラスターの設定

Amazon DocumentDB クラスターごとに、DocumentDB サーバーレスキャパシティ、プロビジョニングされたキャパシティ、またはその両方の任意の組み合わせを選択できます。

混合設定クラスターと呼ばれる、DocumentDB サーバーレスキャパシティとプロビジョニングされたキャパシティの両方を含むクラスターを設定できます。たとえば、DocumentDB サーバーレスライターで使用できる以上の読み取り/書き込み容量が必要であるとします。この場合、非常に大きいプロビジョン済みライターを使用してクラスターをセットアップできます。その後も、リーダーにDocumentDB サーバーレスを使用できます。また、クラスターの書き込みワークロードは変化しているが、読み取りワークロードは安定しているとします。この場合、DocumentDB サーバーレスライターと 1 つ以上のプロビジョニングされたリーダーを使用してクラスターを設定できます。

すべての容量が DocumentDB サーバーレスによって管理されるクラスターを設定することもできます。これを行うには、新しいクラスターを作成し、最初から DocumentDB サーバーレスを使用できます。または、既存のクラスターのすべてのプロビジョニングされた容量を DocumentDB サーバーレスに置き換えることができます。DocumentDB サーバーレスで新しいクラスターを作成する手順、または既存のクラスターを DocumentDB サーバーレスに切り替える手順については、Amazon DocumentDB サーバーレスを使用するクラスターの作成「」および「」を参照してくださいAmazon DocumentDB サーバーレスへの移行。

クラスター内で DocumentDB サーバーレスをまったく使用しない場合、クラスター内のすべてのライターとリーダーがプロビジョニングされます。これは、ほとんどのユーザーが慣れている最も一般的なタイプのクラスターです。プロビジョン済み容量は一定です。料金は比較的簡単に予測できます。ただし、必要な容量を事前に予測する必要があります。場合によっては、予測が不正確だったり、容量のニーズが変わったりすることもあります。このような場合、クラスターのプロビジョニング不足 (希望よりも遅い) またはプロビジョニング過剰 (希望よりも高価) になる可能性があります。

Amazon DocumentDB サーバーレススケーリング容量

Amazon DocumentDB サーバーレスの測定単位は、DocumentDB キャパシティーユニット (DCU) です。DocumentDB サーバーレススケーリング容量は、プロビジョニングされたクラスターに使用するインスタンスクラスに関連付けられていません。

各 DCU は、約 2 ギビバイト (GiB) のメモリ、対応する CPU、ネットワークの組み合わせです。この単位を使用して、データベース容量の範囲を指定します。`ServerlessDatabaseCapacity` および `DCUUtilization` CloudWatch メトリクスは、データベースが実際に使用している容量と、その容量が指定された範囲内のどこにあるかを判断するのに役立ちます。

いつでも、各 DocumentDB サーバーレスライターまたはリーダーには容量があります。容量は DCUs。容量は、ライターまたはリーダーがスケーリングするごとに増減します。この値は毎秒測定されます。DocumentDB サーバーレスを使用するクラスターごとに、容量範囲を定義します。各 DocumentDB サーバーレスライターまたはリーダーがスケーリングできる最小容量値と最大容量値です。容量範囲は、クラスター内の DocumentDB サーバーレスライターまたはリーダーごとに同じです。各 DocumentDB サーバーレスライターまたはリーダーには独自の容量があり、その範囲内のどこかに落ちます。

DocumentDB サーバーレスは、容量範囲が 0.5～256 DCU の DocumentDB 5.0.0 以降でサポートされています。 DCUs

定義できる DocumentDB サーバーレスの最小容量は 0.5 DCUsです。サポートされる最大容量値以下の場合は、大きい数値を指定できます。最小容量を少数に設定すると、負荷の低いクラスターは最小限のコンピューティングリソースを消費できます。同時に、直ちに接続を受け入れ、ビジーになったらスケールアップする準備ができています。

最小値を、各ライターまたはリーダーがバッファプールにアプリケーションのワーキングセットを保持できるようにする値に設定することをお勧めします。これにより、アイドル中にバッファプールの内容が破棄されることはありません。スケーリング容量範囲を選択する際のすべての考慮事項については、「」を参照してくださいDocumentDB サーバーレスクラスターのスケーリング容量範囲の選択。

マルチ AZ 配置でのリーダーの設定方法に応じて、その容量はライターの容量に関連付けることも、独立させることもできます。これを行う方法については、「サーバーレスリーダーの昇格階層の表示と変更」を参照してください。

DocumentDB サーバーレスのモニタリングには、クラスター内のライターとリーダーの容量値を経時的に測定する必要があります。データベースが最小容量にスケールダウンしない場合は、最小値の調整やデータベースアプリケーションの最適化などのアクションを実行できます。データベースが常に最大容量に達している場合は、最大容量を増やすなどのアクションを実行できます。また、データベースアプリケーションを最適化し、クエリのロードをより多くのリーダーに分散させることもできます。

DocumentDB サーバーレス容量の料金は、DCU 時間単位で測定されます。DocumentDB サーバーレス料金の計算方法については、Amazon DocumentDB の料金」を参照してください。クラスター内のライターとリーダーの合計数が n であるとします。この場合、クラスターはデータベースオペレーションを実行していないときに約 n x 最小 DCUs を消費します。Amazon DocumentDB 自体は、少量の負荷を引き起こすモニタリングまたはメンテナンスオペレーションを実行する場合があります。データベースがフルキャパシティで実行されている場合、そのクラスターは n x 最大 DCUsを消費しません。

適切な最小 DCU 値と最大 DCU 値の選択の詳細については、「」を参照してくださいDocumentDB サーバーレスクラスターのスケーリング容量範囲の選択。指定する最小 DCU 値と最大 DCU 値は、一部の Amazon DocumentDB インスタンスの制限にも影響します。容量範囲とインスタンスの制限間の相互作用の詳細については、「」を参照してくださいAmazon DocumentDB サーバーレスインスタンスの制限。

Amazon DocumentDB サーバーレススケーリング

DocumentDB サーバーレスライターまたはリーダーごとに、Amazon DocumentDB は CPU、メモリ、ネットワークなどのリソースの使用率を継続的に追跡します。これらの測定値を総称してロードと呼びます。ロードには、アプリケーションによって実行されるデータベースオペレーションが含まれます。また、データベースサーバーと Amazon DocumentDB 管理タスクのバックグラウンド処理も含まれます。容量がこれらのいずれかによって制約されている場合、DocumentDB サーバーレスはスケールアップします。DocumentDB サーバーレスは、そうすることで解決できるパフォーマンスの問題を検出するとスケールアップします。「」の手順を使用して、リソース使用率と DocumentDB サーバーレススケーリングへの影響をモニタリングできますAmazon DocumentDB サーバーレスのモニタリング。

負荷は、クラスター内のライターとリーダーによって異なる場合があります。ライターは書き込みオペレーションを処理し、クラスターボリュームへのすべてのデータ変更を実行します。閲覧者は読み取り専用リクエストを処理できます。

スケーリングは、データベースの DocumentDB サーバーレス容量を増減するオペレーションです。DocumentDB サーバーレスでは、各ライターとリーダーの現在の容量値は DCUs。DocumentDB サーバーレスは、現在の容量が低すぎて負荷を処理できない場合、ライターまたはリーダーをより高い容量にスケールアップします。ライターまたはリーダーの現在の容量が必要以上に大きい場合、小さい容量にスケールダウンします。

DocumentDB サーバーレスは、容量を段階的に増やすことができます。ワークロードの需要がライターまたはリーダーの現在のデータベース容量に達し始めると、DocumentDB サーバーレスはそのライターまたはリーダーの DCUs の数を増やします。DocumentDB サーバーレスは、消費されるリソースに最適なパフォーマンスを提供するために必要な増分で容量をスケーリングします。スケーリングは、0.5 DCUs という小さな増分で行われます。現在の容量が大きいほど、スケーリングの増分が大きくなり、そのため、スケーリングがより高速になります。

DocumentDB サーバーレススケーリングは非常に頻繁で、きめ細かで、中断されないため、 で個別のイベントは発生しません AWS Management Console。代わりに、`serverlessDatabaseCapacity`や などの Amazon CloudWatch メトリクスを測定し`DCUUtilization`、最小値、最大値、平均値を経時的に追跡できます。DocumentDB サーバーレスのモニタリングの詳細については、「」を参照してくださいAmazon DocumentDB サーバーレスのモニタリング。

スケールアップまたはスケールダウンは、以下が原因で発生する可能性があります。

- メモリ使用率
- CPU 使用率
- ネットワーク使用率
- ストレージの使用率

DocumentDB サーバーレスインスタンスのスケールアップ/スケールダウンの原因をモニタリングできます。詳細については、「Amazon DocumentDB サーバーレスのモニタリング」を参照してください。

リーダースケールは、関連するライターと同時に作成するか、ライターから独立して作成するかを選択できます。これを行うには、そのリーダーの昇格階層を指定します。

- DocumentDB サーバーレスリーダーは、昇格階層 0 および 1 で、ライターと同時にスケーリングします。このスケーリング動作によって、優先階層 0 および 1 のリーダーは可用性に最適です。これは、フェイルオーバー時にライターからワークロードを引き継ぐために、常に適切な容量に合わせてサイズ調整されているためです。
- 昇格階層 2 ~ 15 のリーダーは、ライターとは独立してスケーリングできます。各リーダーは、クラスターに指定した最小および最大 DCU 値内に留まります。リーダーが関連するライター DB とは独立してスケーリングすると、ライターが大量のトランザクションを処理し続けている間に、リーダーがアイドル状態になってスケールダウンする場合があります。下位の昇格階層で他のリーダーが利用できない場合でも、フェイルオーバーターゲットとして利用できます。ただし、ライターに昇格した場合は、ライターのワークロード全体を処理するためにスケールアップが必要になる場合があります。

サーバーレスインスタンスの昇格階層の表示と変更の詳細については、「」を参照してくださいサーバーレスリーダーの昇格階層の表示と変更。

DocumentDB サーバーレススケーリングは、データベース接続が開いている間、トランザクションが進行中などに発生する可能性があります。DocumentDB サーバーレスは、クワイエットポイントがスケーリングを開始するまで待機しません。スケーリングによって、進行中のデータベースオペレーションが中断されることはありません。

ワークロードで 1 つのライターと 1 つのリーダーで使用できる以上の読み取り容量が必要な場合は、複数の DocumentDB サーバーレスリーダーをクラスターに追加できます。各 DocumentDB サーバーレスリーダーは、クラスターに指定した最小容量値と最大容量値の範囲内でスケールできます。クラスターのリーダーエンドポイントを使用して、読み取り専用のセッションをリーダーに送信し、ライターのロードを軽減できます。

DocumentDB サーバーレスがスケーリングを実行するかどうか、および起動後にスケーリングが発生する速度も、クラスターの最小 DCU 設定と最大 DCU 設定によって異なります。さらに、リーダーとライターが一緒にスケーリングするように設定されているか、ライターとは独立してスケーリングするように設定されているかによって異なります。スケーリング設定の詳細については、「」を参照してくださいAmazon DocumentDB サーバーレススケーリング設定。

アイドル状態 (0.5 DCUs)

Amazon DocumentDB サーバーレスライターまたはリーダーがアイドル状態のとき、クラスターの MinCapacity が 0.5 に設定されている場合、DocumentDB サーバーレスインスタンスはアイドル状態の 0.5 DCUs へのスケールダウンをサポートします。

アイドル状態では、DocumentDB サーバーレスインスタンスには、ほとんどの本稼働ワークロードをサポートするのに十分な CPU コンピューティング容量がありませんが、新しいワークロードをサポートするために迅速にスケールアップする準備ができています。アイドル状態では、DocumentDB サーバーレスインスタンスには通常、少なくとも 1.0 ~ 2.5 DCUsが必要です。したがって、DocumentDB サーバーレスインスタンスがアイドル状態から非アイドル状態にスケールアップすると、1.0 ~ 2.5 DCUs (または低い場合は MaxCapacity の値) に直接スケールアップされます。

アイドル状態のときに 0.5 DCUs へのスケールダウンをサポートするために、MinCapacity が 1.0 DCUs。MinCapacity 設定によって制限がどのように影響を受けるかの詳細については、「」を参照してくださいAmazon DocumentDB サーバーレスインスタンスの制限。

DocumentDB サーバーレスの要件と制限

機能の要件

利用可能なリージョン

Amazon DocumentDB サーバーレスインスタンスタイプは、次のリージョンで使用できます。

次の AWS CLI コマンドを使用して、特定のリージョンで提供されている正確な DocumentDB サーバーレスインスタンスオプションを確認できます。

```
aws docdb describe-orderable-db-instance-options \
        --region my_region \
        --db-instance-class db.serverless \
        --engine docdb
```

エンジンバージョンの可用性

DocumentDB サーバーレスは、Amazon DocumentDB 5.0.0 エンジンバージョンでサポートされています。

> **Note**
>
> DocumentDB サーバーレスは、5.0.0 の新しいパッチバージョンでのみサポートされています。クラスターが最新のエンジンパッチバージョンに更新されていることを確認してください。Amazon DocumentDB クラスターのメンテナンスの詳細については、「」を参照してください。 Amazon DocumentDB のメンテナンス

クラスターの設定

Amazon DocumentDB サーバーレスインスタンスを Amazon DocumentDB クラスターに追加する前に、クラスターに `ServerlessV2ScalingConfiguration`パラメータも設定する必要があります。これは、クラスター内の DocumentDB サーバーレスインスタンスの容量範囲を定義します。スケーリング設定の詳細については、「」を参照してくださいAmazon DocumentDB サーバーレススケーリング設定。

特定の Amazon DocumentDB 機能の最小スケーリング容量範囲設定

Amazon DocumentDB の一部の機能は DocumentDB サーバーレスで動作しますが、特定のワークロードでこれらの機能のメモリ要件に必要な容量範囲を下回ると、問題が発生する可能性があります。その場合、通常よりもデータベースのパフォーマンスが低下するか、メモリ不足のエラーが発生する可能性があります。

以下の機能では、最適なオペレーションのためにより高い `MinCapacity`および/または `MaxCapacity` 値を設定する必要があります。

- Performance Insights
- データボリュームが大きいクラスターでのサーバーレスインスタンスの作成

  これには、クラスター復元の一部としてのサーバーレスインスタンスの作成が含まれます。

適切な容量範囲の設定に関する推奨事項 (この機能を使用している場合) については、「」を参照してくださいDocumentDB サーバーレスクラスターのスケーリング容量範囲の選択。容量範囲の設定が間違っているためにデータベースでメモリ不足のエラーが発生した場合のトラブルシューティング情報については、「メモリ不足エラーを回避する」を参照してください。

機能の制限

Amazon DocumentDB プロビジョニングインスタンスの以下の機能は、DocumentDB サーバーレスインスタンスでは使用できません。

- グローバルクラスター

Amazon DocumentDB サーバーレスを使用するクラスターの作成

Amazon DocumentDB サーバーレスクラスターの作成

Amazon DocumentDB サーバーレスでは、クラスターはプロビジョニングされたクラスターと交換可能です。一部のインスタンスがサーバーレスを使用し、一部のインスタンスがプロビジョニングされているクラスターを持つことができます。

目的のリージョンとエンジンバージョンが DocumentDB サーバーレスをサポートしていることを確認します。「DocumentDB サーバーレスの要件と制限」を参照してください。

サーバーレスインスタンスを追加できる Amazon DocumentDB クラスターを作成するには、 と同じ手順に従いますAmazon DocumentDB クラスターの作成。唯一の違いは、引`ServerlessV2ScalingConfiguration`数も指定する必要があることです。

`ServerlessV2ScalingConfiguration` 引数は、DocumentDB サーバーレスインスタンスのスケーリング容量範囲を指定します。これは、クラスター内のすべての DocumentDB サーバーレスインスタンスに適用される DocumentDB キャパシティユニット (DCU) DocumentDB の最小値と最大値で構成されます。

- `MinCapacity` 値は最小スケーリング容量を指定します。
- `MaxCapacity` 値は最大スケーリング容量を指定します。

スケーリングの詳細については、「Amazon DocumentDB サーバーレススケーリング設定」を参照してください。

Using the AWS Management Console

次の AWS Management Console 設定例は、DocumentDB サーバーレスクラスターを作成する方法を示しています。

1. AWS Management Console にサインインし、Amazon DocumentDB 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。

> **Tip**
>
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン

(≡)
を選択します。

クラスターテーブルが表示されます。

3. [作成] を選択します。

   Amazon DocumentDB クラスターの作成ページが表示されます。

4. Amazon DocumentDB クラスターの作成ページのクラスタータイプセクションで、インスタンスベースのクラスターを選択します (これがデフォルトのオプションです) 。

5. クラスター設定セクションで、次の操作を行います。

   a. [クラスター識別子] には、**myserverlesscluster** などの一意の名称を入力します。コンソール操作では、入力方法に関係なくクラスター名のすべての文字が小文字に変換されることにご留意ください。

   b. エンジンバージョンでは、5.0.0 を選択します (これはデフォルトのオプションです) 。

6. [クラスターストレージ設定]セクションでは、[Amazon DocumentDB 標準] (デフォルトのオプションです) を選択します。

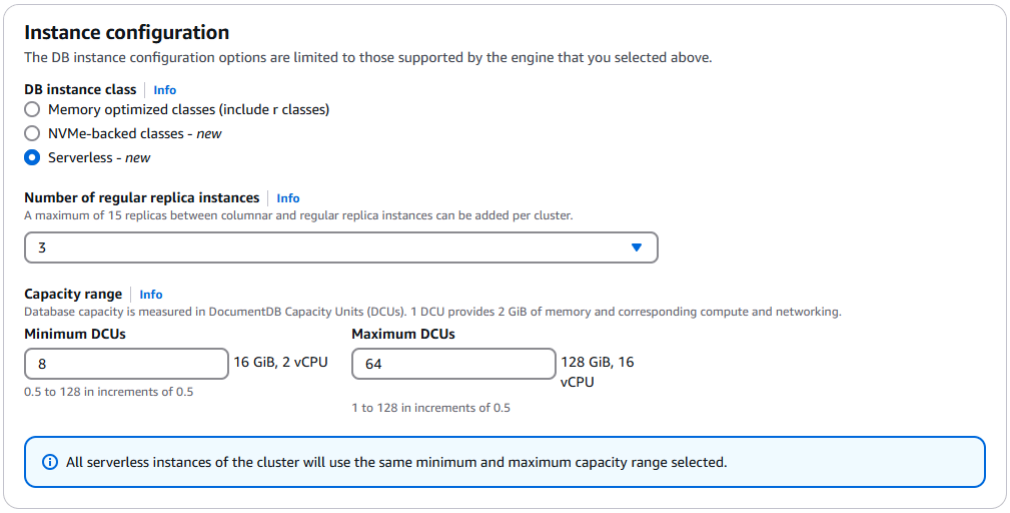
   > **Note**
   >
   > このカテゴリのもう 1 つのオプションは、Amazon DocumentDB I/O 最適化です。いずれかのオプションの詳細については、「」を参照してください。 Amazon DocumentDB クラスターストレージ設定

7. インスタンス設定 セクション:

   a. [DB インスタンスクラス] で、[サーバーレス] を選択します。

   b. 通常のレプリカインスタンスの数で、3 を選択します (これはデフォルトのオプションです) 。

   c. 容量範囲セクションで、最小 DCUsと最大 DCUs。これらのパラメータの設定については、「」を参照してくださいAmazon DocumentDB サーバーレスインスタンスの制限。



8. Connectivity セクションで、デフォルト設定のEC2 コンピューティングリソースに接続しない」のままにします。
9. 認証セクションで、プライマリユーザーのユーザー名を入力し、セルフマネージドを選択します。パスワードを入力して確認します。

   代わりに Managed in AWS Secrets Manager を選択した場合は、詳細についてはAmazon DocumentDB と によるパスワード管理 AWS Secrets Manager「」を参照してください。
10. 他のすべてのオプションはデフォルトのままにして、[クラスターの作成] を選択します。

Using the AWS CLI

次の例では、各###############を独自の情報または設定パラメータに置き換えます。

を使用して DocumentDB サーバーレスインスタンスと互換性のあるクラスターを作成するには AWS CLI、「」を参照してくださいを使用したクラスターの作成 AWS CLI。

`create-db-cluster` コマンドに次の追加パラメータを含めます。

```
--serverless-v2-scaling-configuration
     MinCapacity=minimum_capacity,MaxCapacity=maximum_capacity
```

例:

```
aws docdb create-db-cluster \
      --db-cluster-identifier sample-cluster \
      --engine docdb \
```

```
    --engine-version 5.0.0 \
    --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16 \
    --master-username user-name \
    --master-user-password password
```

Amazon DocumentDB サーバーレスインスタンスの追加

DocumentDB サーバーレスインスタンスを追加するには、 の同じ手順に従いクラスターへの Amazon DocumentDB インスタンスの追加、インスタンスクラスとして db.serverless を必ず指定します。

を使用してサーバーレスインスタンスを追加します AWS Management Console。

コンソールを使用して DocumentDB サーバーレスインスタンスを追加するには、「」を参照クラスターへの Amazon DocumentDB インスタンスの追加し、「 の使用 AWS Management Console」タブを選択します。

を使用したサーバーレスインスタンスの追加 AWS CLI

を使用して DocumentDB サーバーレスインスタンスを追加するには AWS CLI、「」を参照クラスターへの Amazon DocumentDB インスタンスの追加し、「 の使用 AWS CLI」タブを選択します。

次のインスタンスクラスの CLI 引数を使用します。

```
--db-instance-class db.serverless
```

例:

```
aws docdb create-db-instance \
      --db-cluster-identifier sample-cluster \
      --db-instance-identifier sample-instance \
      --db-instance-class db.serverless \
      --engine docdb
```

Amazon DocumentDB サーバーレスへの移行

トピック

- 既存の DocumentDB クラスターをサーバーレスに移行する
- MongoDB から DocumentDB サーバーレスへの移行

既存の DocumentDB クラスターをサーバーレスに移行する

クラスターのエンジンバージョンのアップグレード

プロビジョニングされたクラスターが DocumentDB サーバーレスをサポートしていない下位のエンジンバージョンを実行している場合は、まずクラスターをサポートされているエンジンバージョンにアップグレードする必要があります。詳細については、「Amazon DocumentDB のインプレースメジャーバージョンアップグレード」を参照してください。

プロビジョニングされたクラスターを DocumentDB サーバーレスに移行する

プロビジョニングされたクラスターを DocumentDB サーバーレスを使用するように切り替えるには、次の手順に従います。

1. DocumentDB サーバーレスで使用するには、プロビジョニングされたクラスターのエンジンバージョンをアップグレードする必要があるかどうかを確認します。「DocumentDB サーバーレスの要件と制限」を参照してください。

   > **Note**
   >
   > プロビジョニングされたクラスターが DocumentDB サーバーレスで使用できないエンジンバージョンを実行している場合は、クラスターのエンジンバージョンをアップグレードします。「Amazon DocumentDB のインプレースメジャーバージョンアップグレード」を参照してください。

2. クラスターのスケーリング設定を設定します。スケーリング設定の選択の詳細については、「」を参照してくださいDocumentDB サーバーレスクラスターのスケーリング容量範囲の選択。クラスターのスケーリング設定を変更するには、「」を参照してくださいクラスターのスケーリング容量範囲設定の表示と変更。
3. からの DocumentDB サーバーレスの要件と制限を考慮して、他のクラスタープロパティを設定しますDocumentDB サーバーレスの要件と制限。
4. クラスターに 1 つ以上の DocumentDB サーバーレスインスタンスを追加します。「Amazon DocumentDB サーバーレスインスタンスの追加」の手順に従います。

   > **Note**
   >
   > 場合によっては、クラスターにプロビジョニングされたリーダーインスタンスが既に 1 つ以上あることがあります。その場合は、新しいインスタンスを作成する代わりに、

リーダーの 1 つを DocumentDB サーバーレスインスタンスに変換することを選択できます。これを行うには、「インスタンスクラスの変更」の手順に従います。

5. （オプション) フェイルオーバーオペレーションを実行して、DocumentDB サーバーレスインスタンスをクラスターライターにします。「Amazon DocumentDB フェイルオーバー」を参照してください。
6. （オプション) 残りのプロビジョニングされた Amazon DocumentDB インスタンスを DocumentDB サーバーレスインスタンスに変換するか (「」を参照インスタンスクラスの変更）、クラスターから削除します (「」を参照) Amazon DocumentDB インスタンスの削除。

Using the AWS Management Console

次の AWS Management Console 設定例は、Amazon DocumentDB 5.0.0 を実行している Amazon DocumentDB プロビジョニング済みクラスターを使用した移行プロセスを示しています。このクラスターでは、DocumentDB サーバーレスの使用を開始するためにエンジンバージョンのアップグレードは必要ありません。クラスターは という名前でsample、sample、、sample2およびという名前の 3 つのプロビジョニングされたインスタンスで始まりますsample3。この例では、これら 3 つのインスタンスは 3 つのサーバーレスインスタンスに置き換えられます。

1. AWS Management Console にサインインし、Amazon DocumentDB 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。

> **Tip**
> 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

クラスターテーブルが表示されます。

3. クラスターテーブルで、サーバーレスインスタンスを追加するクラスターのチェックボックスをオンにします。
4. [アクション] を選択し、[インスタンスの追加] を選択します。

5. インスタンスの追加ダイアログボックスで、作成する新しいサーバーレスインスタンスごとに DB インスタンスクラスセクションでサーバーレスを選択します。
6. Serverless 容量設定では、ダイアログボックスの容量の説明に基づいてスケーリング設定を設定します。
7. (オプション) 別のインスタンスを追加するには、インスタンスの追加を選択します。新しいインスタンスの必要数に達するまで、インスタンスの追加を続行します。

   この例では、3 つの新しいサーバーレスインスタンスが作成されます。
8. [作成] を選択します。

   インスタンスが作成されるまでに数分かかります。コンソールまたは を使用して AWS CLI 、インスタンスのステータスを表示できます。詳細については、Amazon DocumentDB クラスターのステータスのモニタリング」を参照してください。
9. クラスターテーブルに戻り、元のプロビジョニング済みインスタンスの 3 つすべてのチェックボックスをオンにします。
10. [アクション] を選択し、[削除] を選択します。

削除中、フェイルオーバーが自動的に実行され、残りのインスタンスの 1 つがライターに昇格されます。数分後、削除プロセスが完了します。既存のクラスターに 3 つの DocumentDB サーバーレスインスタンスが含まれるようになりました (Size 列で定義) 。

Using the AWS CLI

次の AWS CLI 設定例は、Amazon DocumentDB 5.0.0 を実行している Amazon DocumentDB プロビジョニング済みクラスターを使用した移行プロセスを示しています。このクラスターでは、DocumentDB サーバーレスの使用を開始するためにエンジンバージョンのアップグレードは必要ありません。クラスターは という名前でsample、sample、、sample2および という名前の 3 つのプロビジョニングされたインスタンスで始まりますsample3。この例では、これら 3 つのインスタンスは 3 つのサーバーレスインスタンスに置き換えられます。クラスターは という名前でsample-cluster、ライターインスタンスとリーダーインスタンスsample-provisioned-instance-2の sample-provisioned-instance-1と という名前の 2 つのプロビジョニングされたインスタンスで始まります。

次の例では、各###############を独自の情報または設定パラメータに置き換えます。

aws docdb describe-db-clusters オペレーションを使用して、クラスターのステータスを決定します。次のコードは、クラスターのステータスを検索sample-clusterし、結果をテーブルに出力します。

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterMembers' \
    --output table
```

```
-------------------------------------------------------------------------------------------------
|                                          DescribeDBClusters
                  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  DBClusterParameterGroupStatus |      DBInstanceIdentifier       | IsClusterWriter
  | PromotionTier  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  in-sync                       |  sample-provisioned-instance-2  |  False
  |  1             |
|  in-sync                       |  sample-provisioned-instance-1  |  True
  |  1             |
+---------------------------------+---------------------------------
+------------------+----------------+
```

クラスターのスケーリング設定を追加します。

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16
```

サーバーレスインスタンスを追加します。この例では、 sample-serverless-instance-1および という名前の新しいサーバーレスインスタンスが追加されsample-serverless-instance-2ています。

```
aws docdb create-db-instance \
    --db-cluster-identifier sample-cluster \
    --db-instance-identifier sample-serverless-instance-1 \
    --db-instance-class db.serverless \
    --engine docdb

aws docdb create-db-instance \
    --db-cluster-identifier sample-cluster \
    --db-instance-identifier sample-serverless-instance-2 \
    --db-instance-class db.serverless \
```

```
    --engine docdb
```

次を入力して、サーバーレスインスタンスが使用可能になるまで待ってから次に進みます。

```
aws docdb wait db-instance-available \
    --db-instance-identifier sample-serverless-instance-1

aws docdb wait db-instance-available \
    --db-instance-identifier sample-serverless-instance-2
```

フェイルオーバーを実行して、新しいsample-serverless-instance-1インスタンスをクラスターライターにします。

```
aws docdb failover-db-cluster \
    --db-cluster-identifier sample-cluster \
    --target-db-instance-identifier sample-serverless-instance-1
```

フェイルオーバーが完了するまでに数秒かかり、その後 sample-serverless-instance-1 がクラスターライターになります。これを次の入力で検証します。

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterMembers' \
    --output table
```

```
-------------------------------------------------------------------------------------------
|                                        DescribeDBClusters
                  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  DBClusterParameterGroupStatus |      DBInstanceIdentifier       | IsClusterWriter
  | PromotionTier  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  in-sync                       |  sample-provisioned-instance-2  |  False
  |  1             |
|  in-sync                       |  sample-provisioned-instance-1  |  False
  |  1             |
|  in-sync                       |  sample-serverless-instance-2   |  False
  |  1             |
```

```
|  in-sync                      |  sample-serverless-instance-1   |  True
  |  1             |
+-------------------------------+---------------------------------
+-----------------+---------------+
```

最後に、元のプロビジョニングされたインスタンスを削除します。

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-provisioned-instance-1

aws docdb delete-db-instance \
    --db-instance-identifier sample-provisioned-instance-2
```

MongoDB から DocumentDB サーバーレスへの移行

プロビジョニングされた Amazon DocumentDB と同様に、MongoDB データベースを DocumentDB サーバーレスに移行できます。 Amazon DocumentDB 詳細については、「Amazon DocumentDB への移行」を参照してください。

Amazon DocumentDB サーバーレスの管理

クラスターのスケーリング容量範囲設定の表示と変更

`ServerlessV2ScalingConfiguration` 引数は、DocumentDB サーバーレスインスタンスのスケーリング容量範囲を指定します。これは、クラスター内のすべての DocumentDB サーバーレスインスタンスに適用される DocumentDB キャパシティーユニット (DCU) の最小値と最大値で構成されます。

- **`MinCapacity`** — クラスター内の DocumentDB サーバーレスインスタンスの最小スケーリング容量。
- **`MaxCapacity`** — クラスター内の DocumentDB サーバーレスインスタンスの最大スケーリング容量。

> **Note**
>
> 次のスケーリング設定の変更では、新しい `MinCapacity`と `MaxCapacity`の値を反映するためにインスタンスを再起動する必要があります。

- MaxCapacity 値の変更
- MinCapacity 値をより高い値から 1.0 以下に変更する
- MinCapacity 値を低い値から 1.0 より大きい値に変更する

スケーリング設定と適切なスケーリング容量制限の選択方法の詳細については、「」を参照してくださいAmazon DocumentDB サーバーレススケーリング設定。

Using the AWS Management Console

次の AWS Management Console 設定例は、DocumentDB サーバーレスクラスターのスケーリング設定を表示および編集する方法を示しています。

1. AWS Management Console にサインインし、Amazon DocumentDB 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。

   > **Tip**
   >
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

   クラスターテーブルが表示されます。
3. クラスターテーブルで、スケーリング容量を変更するクラスターのチェックボックスをオンにします。
4. [アクション] を選択してから、[変更] を選択します。

   クラスターの変更ダイアログボックスが表示されます。
5. Serverless 容量設定セクションを見つけ、ダイアログボックスの容量の説明に基づいてスケーリング設定 (容量範囲) を設定します。

   スケーリングと容量範囲の詳細については、「」を参照してくださいAmazon DocumentDB サーバーレススケーリング設定。
6. [続行] をクリックしてください。

7. [変更のスケジューリング] で、[すぐに適用] を選択します。
8. [クラスタークラスターの変更] を選択します。
9. 変更が完了したら、各サーバーレスインスタンスを再起動する必要があります。ライターの可用性を最小限に抑えるには、次の一連のオペレーションを実行します。

    a. 各サーバーレスリーダーインスタンスを再起動します。

        i. リーダーインスタンスを選択し、アクションを選択し、再起動します。
        ii. インスタンスのステータスが使用可能に戻るまで待ちます。

    b. 再起動されたサーバーレスインスタンスへのフェイルオーバーを実行します。

        i. クラスターを選択し、アクション、フェイルオーバーを選択します。
        ii. フェイルオーバーオペレーションが完了するまで待ちます。

    c. 残りのサーバーレスインスタンスを再起動します。

        i. 残りのインスタンスを選択し、アクションを選択し、再起動します。
        ii. インスタンスのステータスが Available に戻るまで待ちます。

Using the AWS CLI

次の AWS CLI 設定例は、現在のスケーリング設定を示しています。

次の例では、各*###############*を独自の情報または設定パラメータに置き換えます。

クラスターの現在のスケーリング設定は、 `describe-db-clusters` AWS CLI コマンドを使用して表示できます。

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[0].ServerlessV2ScalingConfiguration'
```

このコマンドの出力は次のとおりです。

```
{
    "MinCapacity": 0.5,
    "MaxCapacity": 16.0
}
```

クラスターのスケーリング設定は、 `modify-db-cluster` コマンドを使用して変更できます。

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=32
```

完了したら、各サーバーレスインスタンスを再起動する必要があります。ライターの可用性を最小限に抑えるために、次の一連のオペレーションを実行できます。

```
aws docdb reboot-db-instance \
    --db-instance-identifier sample-serverless-instance-reader

aws docdb wait db-instance-available \
    --db-instance-identifier sample-serverless-instance-reader

aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster \
   --target-db-instance-identifier sample-serverless-instance-reader

aws docdb reboot-db-instance \
    --db-instance-identifier sample-serverless-instance-writer

aws docdb wait db-instance-available \
    --db-instance-identifier sample-serverless-instance-writer

aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster \
   --target-db-instance-identifier sample-serverless-instance-writer
```

サーバーレスリーダーの昇格階層の表示と変更

複数の DocumentDB サーバーレスインスタンスまたはプロビジョニングされたインスタンスとサーバーレスインスタンスが混在するクラスターの場合は、サーバーレスインスタンスごとに昇格階層設定に注意してください。この設定は、プロビジョニングされたインスタンスよりもサーバーレスインスタンスの動作を制御します。

プロビジョニングされたインスタンスの場合、階層 0 ~ 15 を選択すると、フェイルオーバーオペレーション中に Amazon DocumentDB がライターに昇格するリーダーインスタンスを選択する順序のみが決まります。ただし、サーバーレスインスタンスの場合、階層番号は、インスタンスがライターインスタンスの容量に合わせてスケールアップするか、独自のワークロードに基づいて個別にスケールするかも決定します。階層 0 または 1 のサーバーレスリーダーインスタンスは、少なくともライターインスタンスと同じ最小容量に保持されます。これにより、フェイルオーバーが発生した場

合にライターインスタンスから引き継ぐ準備が整います。ライターインスタンスがプロビジョニングされたインスタンスの場合、Amazon DocumentDB は同等の DocumentDB サーバーレス容量を推定します。この見積もりをサーバーレスリーダーインスタンスの最小容量として使用します。

階層 2 ~ 15 の DocumentDB サーバーレスリーダーインスタンスには、最小容量に同じ制約はありません。アイドル状態のときは、クラスターの容量範囲で指定された最小 DocumentDB 容量単位 (DCU) 値にスケールダウンできます。

Using the AWS Management Console

次の AWS Management Console 設定例は、DocumentDB サーバーレスインスタンスリーダーのプロモーション階層設定を表示および変更する方法を示しています。

1. AWS Management Console にサインインし、Amazon DocumentDB コンソールを開きます。

   各インスタンスの昇格階層は、 の昇格階層列に表示されます AWS Management Console。
2. ナビゲーションペインで クラスター を選択します。

   > **Tip**
   > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

   クラスターテーブルが表示されます。
3. クラスターテーブルで、昇格階層を変更するインスタンスのチェックボックスをオンにします。
4. [アクション] を選択してから、[変更] を選択します。

   インスタンスの変更ダイアログボックスが表示されます。
5. フェイルオーバーセクションを見つけ、昇格階層を希望のレベルに設定します。
6. [続行] をクリックしてください。
7. [変更のスケジューリング] で、[すぐに適用] を選択します。
8. [インスタンスの変更] を選択します。

Using the AWS CLI

次の AWS CLI 設定例は、現在のスケーリング設定を示しています。

次の例では、各##############を独自の情報または設定パラメータに置き換えます。

クラスター内のすべてのインスタンスの昇格階層は、 `describe-db-clusters` AWS CLI コマンドを使用して表示できます。

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[0].DBClusterMembers' \
    --output table
```

このコマンドの出力は次のとおりです。

```
-----------------------------------------------------------------------------------------------
|                                          DescribeDBClusters
                   |
+--------------------------------+--------------------------------
+------------------+----------------+
|  DBClusterParameterGroupStatus |      DBInstanceIdentifier       | IsClusterWriter
  | PromotionTier  |
+--------------------------------+--------------------------------
+------------------+----------------+
|  in-sync                       |  sample-serverless-instance-2   |  False
  |  1             |
|  in-sync                       |  sample-serverless-instance-1   |  True
  |  1             |
+--------------------------------+--------------------------------
+------------------+----------------+
```

特定のインスタンスの昇格階層は、 `modify-db-instance` コマンドを使用して変更できます。

```
aws docdb modify-db-instance \
    --db-instance-identifier sample-serverless-instance-2 \
    --promotion-tier 3
```

Amazon DocumentDB サーバーレスインスタンスの制限

DocumentDB サーバーレスインスタンスの場合、インスタンスあたりの以下の制限は、インスタンスの現在のスケーリング容量によって異なります。

- インスタンスメモリ (GiB)
- 接続 (すべて)
- カーソル制限
- 同時に開くことのできるトランザクション数
- 接続 (アクティブ)

次の表は、インスタンスの現在のスケーリング容量に関連して、サーバーレスインスタンスのインスタンスごとの制限がどのようにスケーリングされるかを示しています。

制限値は、クラスターの`MinCapacity`スケーリング設定が 2 (2) 以上に設定されているかどうかによって異なることに注意してください。`MinCapacity` が 2 以下に設定されている場合、カーソル制限、オープントランザクション、および接続 (アクティブ) の制限値は、より低い最大値に制限されます。(上限の技術的な理由は、「アイドル」状態へのスケールダウンをサポートすることです。詳細については、「アイドル状態 (0.5 DCUs)」を参照してください。

MinCapacity ≤1 の場合の db.serverless インスタンスの制限

現在の容量 (DCU)	インスタンスメモリ (GiB)	接続 (すべて)	カーソル制限	同時に開くことのできるトランザクション数	接続 (アクティブ)
0.5	1	250	6	12	39
1	2	500	12	24	79
2	4	1,000	24	48	173
4	8	2000	48	96	416
8	16	4000	96	192	1071
16	32	8000	132	264	1550

現在の容量 (DCU)	インスタンスメモリ (GiB)	接続 (すべて)	カーソル制限	同時に開くことのできるトランザクション数	接続 (アクティブ)
32	64	16000	132	264	1550
64	128	32000	132	264	1550
128	256	60000	132	264	1550
256	512	60000	132	264	1550

MinCapacity >1 の場合の db.serverless インスタンスの制限

現在の容量 (DCU)	インスタンスメモリ (GiB)	接続 (すべて)	カーソル制限	同時に開くことのできるトランザクション数	接続 (アクティブ)
1.5	3	7250	18	36	124
2	4	1,000	24	48	173
4	8	2000	48	96	416
8	16	4000	96	192	1071
16	32	8000	192	384	2709
32	64	16000	384	768	4500
64	128	32000	768	1536	4500
128	256	60000	1536	3072	4500
256	512	60000	3072	6144	4500

次の CloudWatch メトリクスを使用して、インスタンスごとの制限を監視およびアラームできます。Amazon DocumentDB CloudWatch のメトリクスのさらなる詳細については、Amazon DocumentDB と CloudWatch のモニタリング をご参照ください。

リソース	CloudWatch 制限メトリクス	CloudWatch 使用状況メトリクス (最大 1 分)	CloudWatch 使用状況メトリクス
インスタンスメモリ	-	-	FreeableMemory
接続 (すべて)	DatabaseConnectionsLimit	DatabaseConnectionsMax	DatabaseConnections
カーソル	DatabaseCursorsLimit	DatabaseCursorsMax	DatabaseCursors
トランザクション	TransactionsOpenLimit	TransactionsOpenMax	TransactionsOpen

Amazon DocumentDB サーバーレススケーリング設定

トピック

- DocumentDB サーバーレスクラスターのスケーリング容量範囲の選択
- DocumentDB サーバーレスクラスターMinCapacityの設定の選択
- DocumentDB サーバーレスクラスターMaxCapacityの設定の選択
- メモリ不足エラーを回避する
- サーバーレスインスタンスがスケールダウンしないのはなぜですか？

DocumentDB サーバーレスクラスターのスケーリング容量範囲の選択

Amazon DocumentDB クラスターに Amazon DocumentDBサーバーレスインスタンスを追加する前に、クラスターに `ServerlessV2ScalingConfiguration`パラメータも設定する必要があります。

`ServerlessV2ScalingConfiguration` パラメータは、クラスター内の任意のサーバーレスインスタンスのサーバーレススケーリング容量範囲を定義する 2 つの値で構成されます。

- **MinCapacity** — クラスター内の DocumentDB サーバーレスインスタンスの最小スケーリング容量。
- **MaxCapacity** — クラスター内の DocumentDB サーバーレスインスタンスの最大スケーリング容量。

DocumentDB サーバーレスクラスター**MinCapacity**の設定の選択

には常に 0.5 を選択したいと考えていますMinCapacity。この値により、インスタンスは完全にアイドル状態でアクティブのまま最小容量にスケールダウンできます。ただし、そのクラスターの使用方法やその他の設定によっては、最も効果的な最小容量が異なる場合もあります。最小容量設定を選択する場合は、以下の要素を考慮してください。

- DocumentDB サーバーレスインスタンスのスケーリングレートは、現在の容量によって異なります。現在の容量が大きいほど、スケールアップが速くなります。インスタンスが非常に大容量にすばやくスケールアップする必要がある場合は、最小容量をスケーリングレートが要件を満たす値に設定することを検討してください。
- 通常、特に高ワークロードまたは低ワークロードを想定してインスタンスのインスタンスクラスを変更する場合は、そのエクスペリエンスを使用して、同等の DocumentDB サーバーレス容量範囲を概算で見積もることができます。プロビジョニングされた Amazon DocumentDB インスタンスタイプのメモリサイズを確認するには、「」を参照してくださいインスタンス制限。

  たとえば、クラスターのワークロードが低い場合にdb.r6g.xlargeインスタンスクラスを使用するとします。そのインスタンスクラスには 32 GiB のメモリがあります。したがって、16 MinCapacityの を指定して、ほぼ同じ容量にスケールダウンできるサーバーレスインスタンスを設定できます。これは、各 DCU が約 2 GiB のメモリに対応するためです。インスタンスの使用率が低い場合に備えて、db.r6g.xlargeインスタンスをさらにスケールダウンできるように、ある程度低い値を指定できます。
- インスタンスのバッファキャッシュに一定量のデータがある場合にアプリケーションが最も効率的に機能する場合は、頻繁にアクセスされるデータを保持するのに十分なメモリ容量の最小 DCU 設定を指定することを検討してください。それ以外の場合、サーバーレスインスタンスがより低いメモリサイズにスケールダウンすると、一部のデータがバッファキャッシュから削除されます。次に、インスタンスがスケールアップすると、情報は時間の経過とともにバッファキャッシュに読み戻されます。データをバッファキャッシュに戻すための I/O の量が多い場合は、より高い最小 DCU 値を選択する方が効果的である可能性があります。詳細については、「インスタンスのサイズ指定」を参照してください。

- DocumentDB サーバーレスインスタンスがほとんどの場合特定の容量で実行される場合は、そのベースラインよりも小さい最小容量設定を指定することを検討してください。サーバーレスインスタンスは、現在の容量が必要な容量よりも大幅に低い場合のスケールアップの量と速度を最も効果的に見積もることができます。
- プロビジョニングされたワークロードのメモリ要件が T3 や T4g などの小さなインスタンスクラスで高すぎる場合は、R5 または R6g インスタンスと同等のメモリを提供する最小 DCU 設定を選択します。
- 特に、指定された機能で使用するMinCapacityには、以下の最小値をお勧めします (これらの推奨事項は変更される可能性があります) 。
  - Performance Insights — 2 DCUs
- Amazon DocumentDB では、レプリケーションはストレージレイヤーで発生するため、リーダーの容量はレプリケーションに直接影響しません。ただし、個別にスケールする DocumentDB サーバーレスリーダーインスタンスの場合は、クエリのレイテンシーを回避するために、書き込み負荷の高い期間にワークロードを処理するのに十分な最小容量があることを確認してください。昇格階層 2 ~ 15 のリーダーインスタンスでパフォーマンスの問題が発生した場合は、クラスターの最小容量を増やすことを検討してください。リーダーインスタンスをライターと一緒にスケールするか個別にスケールするかの変更の詳細については、「」を参照してくださいサーバーレスリーダーの昇格階層の表示と変更。

  DocumentDB サーバーレスリーダーインスタンスを持つクラスターがある場合、リーダーの昇格階層が 0 または 1 でない場合、リーダーはライターインスタンスに合わせてスケーリングされません。この場合、最小容量を小さく設定すると、レプリケーションの遅延が大きくなる場合があります。これは、データベースがビジー状態のときに、ライターからの変更を適用するのに十分な容量がリーダーにない可能性があるためです。最小容量は、ライターインスタンスと同等の量のメモリと CPU を表す値に設定することをお勧めします。
- DocumentDB サーバーレスインスタンスが最小容量から最大容量にスケールするのにかかる時間は、最小 DCU 値と最大 DCU 値の差によって異なります。インスタンスの現在の容量が大きい場合、DocumentDB サーバーレスは、インスタンスが小さな容量から起動する場合よりも大きくスケールアップします。したがって、比較的大きな最大容量を指定し、インスタンスがその容量の近くでほとんどの時間を費やす場合は、最小 DCU 設定を増やすことを検討してください。これにより、アイドル状態のインスタンスは最大容量までより迅速にスケールアップできます。
- 特定のインスタンス制限は、接続制限、カーソル制限、オープントランザクション制限など、サーバーレスインスタンスの現在の容量によって決まります。インスタンスの現在の容量が小さい場合、それに応じて制限も小さくなります。サーバーレスインスタンスをそのMinCapacity値にスケールダウンするときにこれらの制限に問題がある場合は、 をより高い値MinCapacityに増やす

ことを検討してください。詳細については、「Amazon DocumentDB サーバーレスインスタンスの制限」を参照してください。

- さらに、アクティブな接続制限、カーソル制限、オープントランザクション制限など、MinCapacity が 1.0 DCUs以下に設定されている場合、特定のインスタンス制限はより低い最大値に制限されます。これらの制限がワークロードに対して不十分な場合は、少なくとも 1.5 DCUsMinCapacityの値を使用してください。詳細については、「Amazon DocumentDB サーバーレスインスタンスの制限」を参照してください。

クラスターのスケーリング設定を変更する方法については、「」を参照してくださいAmazon DocumentDB サーバーレスの管理。

DocumentDB サーバーレスクラスターMaxCapacityの設定の選択

DocumentDB サーバーレスの最大容量設定には、常に高い値を選択したいと考えています。最大容量が大きいと、インスタンスは負荷の高いワークロードを実行しているときに最もスケールアップできます。値を小さくすると、予期せぬ料金が発生する可能性を回避できます。そのクラスターの使用方法およびその他の設定によっては、最も効果的な値が当初検討していたより大きくなったり、小さくなったりすることがあります。最大容量設定を選択する場合は、以下の要素を考慮してください。

- 最大容量は、最小容量より大きくなければなりません。最小容量と最大容量を同一に設定することができます。ただし、その場合は容量がスケールアップまたはスケールダウンすることはありません。したがって、テスト以外では、最小容量と最大容量に同じ値を使用することは適切ではありません。
- 最大容量は 1.0 DCUs 以上、最大 256 DCUsである必要があります。
- サーバーレスインスタンスのスケーリングとリソース使用量をモニタリングすることをお勧めします。サーバーレスインスタンスが最大容量に頻繁にスケーリングされ、リソースの制約 (DCUUtilizationメトリクスが 100.0 の場合など) にヒットする場合は、より高いMaxCapacity値を選択することをお勧めします。詳細については、「Amazon DocumentDB サーバーレスのモニタリング」を参照してください。
- 通常、特に高ワークロードまたは低ワークロードを想定してプロビジョニングされたインスタンスのインスタンスクラスを変更する場合は、そのエクスペリエンスを使用して同等の DocumentDB サーバーレス容量範囲を見積もることができます。プロビジョニングされた Amazon DocumentDB インスタンスのメモリサイズを確認するには、「」を参照してくださいインスタンス制限。

  たとえば、クラスターのワークロードが高い場合にdb.r6g.4xlargeインスタンスクラスを使用するとします。そのインスタンスクラスには 128 GiB のメモリがあります。したがって、最大

DCU 設定を 64 に指定して、ほぼ同じ容量にスケールアップできるサーバーレスインスタンスを設定できます。これは、各 DCU が約 2 GiB のメモリに対応するためです。インスタンスにワークロードを効果的に処理するのに十分な容量がない場合に備えて、db.r6g.4xlargeインスタンスをさらにスケールアップできるように、ある程度高い値を指定できます。

- データベース使用量に予算の上限がある場合は、すべてのサーバーレスインスタンスが常に最大容量で実行されていても、その上限内に留まる値を選択します。クラスターにサーバーレスインスタンスが n つある場合、クラスターがいつでも消費できる理論上の最大サーバーレス容量は、クラスターの最大 DCU 設定の n 倍であることに注意してください。(例えば、一部のリーダーがライターから独立してスケーリングする場合など、実際の消費量は少なくなる場合があります)。
- サーバーレスリーダーインスタンスを使用してライターインスタンスから読み取り専用ワークロードの一部をオフロードする場合、より低い最大容量設定を選択できる場合があります。これは、各リーダーインスタンスがクラスターに 1 つのインスタンスしか含まれていないかのようにスケールする必要がないことを反映させるために行います。
- データベースパラメータの設定間違いやアプリケーション内の非効率的なクエリによる過度の使用から保護したいとします。その場合、設定可能な理論的な最大値よりも最大容量設定を小さく選択することで、誤って過剰に使用することを回避できます。
- 実際のユーザーアクティビティによるスパイクがまれしか発生しない場合は、最大容量設定を選択する際にその機会を考慮できます。アプリケーションが完全なパフォーマンスとスケーラビリティで動作し続けることを優先する場合は、通常の使用状況よりも大きい最大容量設定を指定できます。アクティビティの非常に極端なスパイク中、アプリケーションのスループットが低下しても問題ない場合は、最大容量を少し小さめに設定できます。アプリケーションの実行を維持するのに十分なメモリと CPU リソースがある設定を選択してください。
- 各インスタンスのメモリ使用量を増やすクラスターの設定をオンにする場合は、最大 DCU 値を決定するときにそのメモリを考慮してください。このような設定には、Performance Insights の設定が含まれます。最大 DCU 値により、サーバーレスインスタンスが、それらの機能の使用時にワークロードを処理するのに十分なスケールアップが許可されていることを確認します。最大 DCU 設定の低さと、メモリオーバーヘッドを課す Amazon DocumentDB 機能の組み合わせによって発生する問題のトラブルシューティングについては、メモリ不足エラーを回避する「(以下)」を参照してください。
- 特に、指定された機能で使用するMaxCapacityには、以下の最小値をお勧めします (これらの推奨事項は変更される可能性があります)。
  - データボリュームが大きいクラスターでのサーバーレスインスタンスの作成 – 2 DCUs (クラスター復元の一部としてのサーバーレスインスタンスの作成が含まれます)。
- 特定のインスタンス制限は、接続制限、カーソル制限、オープントランザクション制限など、インスタンスの現在の容量によって決まります。ワークロードMaxCapacityの値を選択するときは、

これらの制限のいずれかによってボトルネックにならないように、これらのインスタンスの制限を念頭に置いてください。詳細については、「Amazon DocumentDB サーバーレスインスタンスの制限」を参照してください。

クラスターのスケーリング設定を変更する方法については、「」を参照してくださいAmazon DocumentDB サーバーレスの管理。

メモリ不足エラーを回避する

DocumentDB サーバーレスインスタンスの 1 つが一貫して最大容量の制限に達すると、Amazon DocumentDB はインスタンスを互換性のないパラメータのステータスに設定することで、この条件を示します。インスタンスのステータスが incompatible-parameters の場合、一部のオペレーションはブロックされます。例えば、エンジンバージョンをアップグレードすることはできません。Amazon DocumentDB インスタンスのステータスの詳細については、Amazon DocumentDB インスタンスのステータスのモニタリング」を参照してください。

通常、インスタンスはout-of-memoryエラーにより頻繁に再起動すると、このステータスになります。Amazon DocumentDB は、このタイプの再起動が発生したときにイベントを記録します。リソースイベントを表示するには、「」を参照してくださいAmazon DocumentDB イベントの表示。Performance Insights などの設定をオンにするとオーバーヘッドが発生するため、メモリ使用率が異常に高くなる可能性があります。また、インスタンスの負荷の高いワークロードや、多数のスキーマオブジェクトに関連付けられたメタデータの管理によっても発生する可能性があります。

インスタンスが最大容量に頻繁に達しないようにメモリ負荷が低くなると、Amazon DocumentDB は自動的にインスタンスのステータスを使用可能な状態に戻します。

この状態から回復させるには、以下アクションの一部またはすべてを実行できます。

- クラスターの DocumentDB 容量単位 (DCU) の最小値を変更して、サーバーレスインスタンスの容量の下限を増やします。これを行うことで、アイドル状態のデータベースが、クラスターでオンになっている機能に必要なメモリよりも少ない容量にスケールダウンする問題を回避できます。クラスターの DCU 設定を変更したら、サーバーレスインスタンスを再起動します。これにより、Amazon DocumentDB がステータスを使用可能な状態にリセットできるかどうかが評価されます。
- クラスターの最大 DCU 値を変更して、サーバーレスインスタンスの容量の上限を増やします。そうすることで、クラスターでオンになっている機能やデータベースワークロードに、ビジー状態のデータベースが十分なメモリがある容量にスケールアップできない問題を回避できます。クラスターの DCU 設定を変更したら、サーバーレスインスタンスを再起動します。これによ

り、Amazon DocumentDB がステータスを使用可能な状態にリセットできるかどうかが評価されます。

- メモリオーバーヘッドが必要な設定をオフにします。たとえば、Performance Insights などの機能がオンになっているが、使用していないとします。その場合は、オフにすることができます。または、クラスターの最小容量値と最大容量値を調整して、これらのタイプの機能で使用されるメモリを考慮することもできます。最小および最大容量設定の選択に関するガイドラインについては、「DocumentDB サーバーレスクラスターのスケーリング容量範囲の選択」を参照してください。
- インスタンスのワークロードを減らします。たとえば、リーダーインスタンスをクラスターに追加して、読み取り専用クエリからの負荷をより多くのインスタンスに分散できます。

サーバーレスインスタンスがスケールダウンしないのはなぜですか？

場合によっては、データベースに負荷がかからなくても、DocumentDB サーバーレスは最小容量までスケールダウンしません。これは、次のような理由で発生します。

- Performance Insights はリソース使用量を増やし、データベースが最小容量にスケールダウンするのを防ぐことができます。主な機能は以下のとおりです。
- リーダーインスタンスが最小にスケールダウンせず、ライターインスタンスと同じかそれ以上の容量に留まる場合は、リーダーインスタンスの優先度層を確認します。階層 0 または 1 の DocumentDB サーバーレスリーダーインスタンスは、少なくともライターインスタンスと同じ最小容量に保持されます。リーダーの優先度の階層を 2 以上に変更して、ライターとは無関係にスケールアップおよびスケールダウンされるようにします。詳細については、「Amazon DocumentDB サーバーレススケーリング」を参照してください。
- データベースワークロードが重いと、リソースの使用量が増える可能性があります。
- データベースのボリュームが大きいと、リソースの使用量が増える可能性があります。Amazon DocumentDB は、クラスター管理にメモリと CPU リソースを使用します。Amazon DocumentDB では、データベースボリュームが大きいクラスターを管理するために、より多くの CPU とメモリが必要です。クラスターの最小容量がクラスター管理に必要な最小容量よりも少ない場合、クラスターは最小容量までスケールダウンされません。
- バックグラウンドメンテナンスアクティビティでは、リソースの使用量を定期的に増やすことができます。

それでもデータベースが設定された最小容量までスケールダウンしない場合は、データベースを停止して再起動し、時間の経過とともに蓄積した可能性があるメモリフラグメントを再利用します。デー

タベースを停止して起動するとダウンタイムが発生するため、これを実行するかどうかは慎重に判断することをお勧めします。

Amazon DocumentDB サーバーレスのモニタリング

Amazon DocumentDB でのモニタリングの詳細については、「」を参照してくださいAmazon DocumentDB のモニタリング。

トピック

- メモリ不足: incompatible-parameters ステータス
- DocumentDB サーバーレスの Amazon CloudWatch メトリクス
- Performance Insights による DocumentDB サーバーレスパフォーマンスのモニタリング

メモリ不足: incompatible-parameters ステータス

サーバーレスインスタンスの 1 つが最大容量の制限に一貫して達する場合、Amazon DocumentDB はインスタンスを互換性のないパラメータのステータスに設定することで、この条件を示します。詳細については、「メモリ不足エラーを回避する」を参照してください。

DocumentDB サーバーレスの Amazon CloudWatch メトリクス

Amazon DocumentDB での CloudWatch の使用の詳細については、「」を参照してくださいAmazon DocumentDB と CloudWatch のモニタリング。

CloudWatch でサーバーレスインスタンスを表示して、 `ServerlessDatabaseCapacity`メトリクスを使用して各インスタンスが消費する容量をモニタリングできます。また、`DatabaseConnections`や クエリなど、標準の DocumentDB CloudWatch メトリクスをすべてモニタリングすることもできます。Amazon DocumentDB でモニタリングできる CloudWatch メトリクスの完全なリストについては、「」を参照してくださいAmazon DocumentDB と CloudWatch のモニタリング。以下の CloudWatch インスタンスレベルのメトリクスは、DocumentDB サーバーレスインスタンスのスケールアップとスケールダウンの方法を理解するためにモニタリングすることが重要です。これらすべてのメトリクスは 1 秒ごとに計算されます。これにより、サーバーレスインスタンスの現在のステータスをモニタリングできます。サーバーレスインスタンスが容量に関連するメトリクスのしきい値に近づいた場合に通知するようにアラームを設定できます。最小容量と最大容量設定は適切か、調整が必要かを判断できます。データベースの効率を最適化するため、どこに注力すべきかを判断できます。

- **ServerlessDatabaseCapacity** — インスタンスレベルのメトリクスとして、現在のインスタンス容量で表DCUs の数を報告します。クラスターレベルのメトリクスとして、クラスター内のすべての DocumentDB サーバーレスインスタンスServerlessDatabaseCapacityの値の平均を表します。
- **DCUUtilization.** — このメトリクスは DocumentDB サーバーレスでは新しいものです。この値は割合 (%) で表されます。これは、ServerlessDatabaseCapacityメトリクスの値をクラスターの最大 DCU 値で割ったものとして計算されます。このメトリクスを解釈してアクションを実行するには、以下のガイドラインを考慮してください。
  - このメトリクスが の値に近づくと100.0、インスタンスは可能な限り大きくスケールアップされています。クラスターの最大 DCU 設定を増やすことを検討してください。これにより、ライターインスタンスとリーダーインスタンスの両方をより高い容量にスケールできます。
  - 読み取り専用ワークロードによってリーダーインスタンスが DCUUtilizationの に近づき100.0、ライターインスタンスが最大容量に近づかないとします。この場合、クラスターにリーダーインスタンスを追加することを検討してください。これにより、ワークロードの読み取り専用部分をより多くのインスタンスに分散し、各リーダーインスタンスの負荷を軽減できます。
  - パフォーマンスとスケーラビリティが主な考慮事項である本番アプリケーションを実行しているとします。この場合、クラスターの最大 DCU 値を高い数値に設定できます。目標は、DCUUtilizationメトリクスを常に 100.0 未満にすることです。最大 DCU 値が高いため、データベースアクティビティに予期しないスパイクが発生した場合に備えて、十分な余裕があることを確信できます。実際に消費されたデータベース容量に対してのみ課金されます。
- **CPUUtilization** — このメトリクスは、プロビジョニングされたインスタンスとは異なり、DocumentDB サーバーレスで解釈されます。DocumentDB サーバーレスの場合、この値は、現在使用されている CPU の量として計算され、クラスターの最大 DCU 値で使用可能な CPU 容量で割られたパーセンテージです。Amazon DocumentDB は、この値を自動的にモニタリングし、インスタンスが CPU 容量の大部分を一貫して使用している場合にサーバーレスインスタンスをスケールアップします。

  このメトリクスが の値に近づくと100.0、インスタンスは最大 CPU 容量に達します。クラスターの最大 DCU 設定を増やすことを検討してください。このメトリクスがリーダーインスタンス100.0の の値に近づいた場合は、クラスターにリーダーインスタンスを追加することを検討してください。これにより、ワークロードの読み取り専用部分をより多くのインスタンスに分散させ、各リーダーインスタンスの負荷を軽減できます。
- **FreeableMemory** — この値は、DocumentDB サーバーレスインスタンスが最大容量にスケールされたときに使用可能な未使用のメモリの量を表します。現在の容量が最大容量を下回っている

DCU ごとに、この値は約 2 GiB 増加します。したがって、インスタンスができるだけ大きくスケールアップされるまで、このメトリクスはゼロに近づきません。

このメトリクスがゼロに近づくと、インスタンスは可能な限りスケールアップし、使用可能なメモリの制限に近づいています。クラスターの最大 DCU 設定を増やすことを検討してください。このメトリクスがリーダーインスタンスで 0 の値に近づいた場合は、クラスターにリーダーインスタンスを追加することを検討してください。これにより、ワークロードの読み取り専用部分をより多くのインスタンスに分散できるため、各リーダーインスタンスのメモリ使用量を減らすことができます。

- **TempStorageIops** — インスタンスにアタッチされたローカルストレージで実行された IOPS の数。これには、読み取りと書き込みの両方の IOPS が含まれます。このメトリクスはカウントを表し、1 秒に 1 回測定されます。これは DocumentDB サーバーレスの新しいメトリクスです。詳細については、「Amazon DocumentDB と CloudWatch のモニタリング」を参照してください。
- **TempStorageThroughput** — インスタンスに関連付けられたローカルストレージとの間で転送されるデータの量。このメトリクスはバイトを表し、1 秒に 1 回測定されます。これは DocumentDB サーバーレスの新しいメトリクスです。詳細については、「Amazon DocumentDB と CloudWatch のモニタリング」を参照してください。

通常、DocumentDB サーバーレスインスタンスのほとんどのスケールアップは、メモリ使用量と CPU アクティビティが原因です。TempStorageIops および TempStorageThroughputメトリクスは、インスタンスとローカルストレージデバイス間の転送のためのネットワークアクティビティが予期しない容量の増加の原因となるまれなケースを診断するのに役立ちます。他のネットワークアクティビティを監視するには、以下の既存のメトリクスを使用できます。

- NetworkReceiveThroughput
- NetworkThroughput
- NetworkTransmitThroughput
- StorageNetworkReceiveThroughput
- StorageNetworkThroughput
- StorageNetworkTransmitThroughput

DocumentDB サーバーレス CloudWatch メトリクスが AWS 請求書にどのように適用されるか

AWS 請求書の DocumentDB サーバーレス料金は、モニタリングできるのと同じ`ServerlessDatabaseCapacity`メトリクスに基づいて計算されます。請求メカニズムは、DocumentDB サーバーレス容量を 1 時間の一部のみ使用する場合、このメトリクスの計算された CloudWatch 平均とは異なることがあります。また、システムの問題で、短時間 CloudWatch メトリクスが利用できない場合にも異なる場合があります。したがって、`ServerlessDatabaseCapacity`平均値から数値を計算した場合とは少し異なる DCU 時間の値が請求書に表示されることがあります。

DocumentDB サーバーレスメトリクスの Amazon CloudWatch CLI コマンドの例

次の AWS CLI 例は、DocumentDB サーバーレスに関連する最も重要な CloudWatch メトリクスをモニタリングする方法を示しています。いずれの場合も、 `--dimensions`パラメータの`Value=`文字列を独自の DocumentDB サーバーレスインスタンスの識別子に置き換えます。

次の Linux の例では、1 時間にわたって 10 分ごとに測定される、インスタンスの最小容量値、最大容量値、平均容量値を表示します。Linux の日付コマンドは、現在の日付と時刻に関連する開始時刻と終了時刻を指定します。`--query` パラメータの `sort_by` 関数は、`Timestamp` のフィールドに基づいて結果を時系列でソートします。

```
aws cloudwatch get-metric-statistics \
    --metric-name "ServerlessDatabaseCapacity" \
    --start-time "$(date -d '1 hour ago')" \
    --end-time "$(date -d 'now')" \
    --period 600 \
    --namespace "AWS/DocDB" \
    --statistics Minimum Maximum Average \
    --dimensions Name=DBInstanceIdentifier,Value=my_instance \
    --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
    --output table
```

次の Linux の例は、クラスター内のインスタンスの容量をモニタリングする方法を示しています。インスタンスの最小容量使用率、最大容量使用率、平均容量使用率を測定します。測定は、1 時間に 1 回、3 時間にわたって行います。これらの例では、固定数の DCUs を表すのではなく、DCUs の上限の割合`ServerlessDatabaseCapacity`を表す `DCUUtilization`メトリクスを使用します。これにより、容量範囲内の最小および最大 DCU 値の実際の数値を知る必要はありません。割合は 0 から 100 までの範囲で表示できます。

```
aws cloudwatch get-metric-statistics \
    --metric-name "DCUUtilization" \
    --start-time "$(date -d '3 hours ago')" \
    --end-time "$(date -d 'now')" \
    --period 3600 \
    --namespace "AWS/DocDB" \
    --statistics Minimum Maximum Average \
    --dimensions Name=DBInstanceIdentifier,Value=my_instance \
    --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
    --output table
```

以下の Linux の例では、前のものと同様の測定を実行します。この場合は、CPUUtilization のメトリクスのための測定になります。測定は、1 時間で 10 分ごとに行われます。数値は、インスタンスの最大容量設定に使用できる CPU リソースに基づいて、使用可能な CPU の割合を表します。

```
aws cloudwatch get-metric-statistics \
    --metric-name "CPUUtilization" \
    --start-time "$(date -d '1 hour ago')" \
    --end-time "$(date -d 'now')" \
    --period 600 \
    --namespace "AWS/DocDB" \
    --statistics Minimum Maximum Average \
    --dimensions Name=DBInstanceIdentifier,Value=my_instance \
    --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
    --output table
```

以下の Linux の例では、前のものと同様の測定を実行します。この場合は、FreeableMemory のメトリクスのための測定になります。測定は、1 時間で 10 分ごとに行われます。

```
aws cloudwatch get-metric-statistics \
    --metric-name "FreeableMemory" \
    --start-time "$(date -d '1 hour ago')" \
    --end-time "$(date -d 'now')" \
    --period 600 \
    --namespace "AWS/DocDB" \
    --statistics Minimum Maximum Average \
    --dimensions Name=DBInstanceIdentifier,Value=my_instance \
    --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
```

```
    --output table
```

Performance Insights による DocumentDB サーバーレスパフォーマンスのモニタリング

Performance Insights を使用して、DocumentDB サーバーレスインスタンスのパフォーマンスをモニタリングできます。パフォーマンスインサイトの手順については、「Performance Insights を使用したモニタリング」を参照してください。

次の新しい Performance Insights カウンターが DocumentDB サーバーレスインスタンスに適用されます。

- **os.general.serverlessDBCapacity** — DCUs 内のインスタンスの現在の容量。値は、インスタンスの ServerlessDatabaseCapacity CloudWatch メトリクスに対応します。
- **os.general.dcuUtilization** — 最大設定容量のうち現在の容量の割合。値は、インスタンスの DCUUtilization CloudWatch メトリクスに対応します。
- **os.general.maxConfiguredDcu** — この DocumentDB サーバーレスインスタンスに設定した最大容量。DCUs。
- **os.general.minConfiguredDcu** — この DocumentDB サーバーレスインスタンス用に設定した最小容量。DCUs。

パフォーマンスインサイトカウンターのすべてのリストは、「カウンターメトリクス用の Performance Insights」を参照してください。

Performance Insights で DocumentDB サーバーレスインスタンスの vCPU 値を表示すると、それらの値はインスタンスの DCU 値に基づく推定値を表します。デフォルトの 1 分間隔では、vCPU 値の小数分は整数に切り上げられます。それ以上の間隔の場合、表示される vCPU 値は、1 分ごとの vCPU 値の整数の平均になります。

Amazon DocumentDB の操作

これらのセクションでは、Amazon DocumentDB を使用した開発について説明します。

トピック

- Amazon DocumentDB に接続する
- Amazon DocumentDB を使用したプログラム

Amazon DocumentDB に接続する

Amazon DocumentDB には、MongoDB ドライバーを使用したプログラムによる接続、SSH トンネリングまたは AWS VPN を介した VPC 外部からの接続、Studio 3T や DataGrip などの一般的なツールなど、複数の接続オプションが用意されています。このサービスは、高可用性のためにレプリカセット接続をサポートしています。分析とレポートでは、JDBC または ODBC ドライバーを使用して接続できます。同じ VPC 内の EC2 インスタンスは、DocumentDB クラスターに直接接続できます。すべての接続には TLS と適切な AWS 認証情報/認証が必要です。

トピック

- プログラムによる Amazon DocumentDB への接続
- Amazon VPC の外部から Amazon DocumentDB クラスターへの接続
- レプリカセットとしての Amazon DocumentDB に接続する
- Studio 3T から Amazon DocumentDB クラスターに接続する
- DataGrip を使用して Amazon DocumentDB に接続する
- Amazon EC2 を使用した接続
- Amazon DocumentDB JDBC ドライバーを使用して接続する
- Amazon DocumentDB JDBC ドライバーを使用して接続する

プログラムによる Amazon DocumentDB への接続

このセクションでは、複数の言語を使用して Amazon DocumentDB (MongoDB 互換) に接続する方法を示すコード例を紹介します。コード例は、Transport Layer Security (TLS) が有効または無効になっているクラスターに接続しているかどうかに基づいて、2 つのセクションに分かれています。TLS は、新しい Amazon DocumentDB クラスターに対してデフォルトで有効になっています。

ただし、必要に応じて TLS を無効にすることができます。詳細については、「Encrypting data in transit」を参照してください。

クラスターが存在する VPC の外部から Amazon DocumentDB に接続しようとしている場合は、「Amazon VPC の外部から Amazon DocumentDB クラスターへの接続」を参照してください。

クラスターに接続する前に、TLS がクラスターで有効になっているかどうかを把握しておく必要があります。次のセクションでは、`tls` または AWS Management Console を使用してクラスターの AWS CLIパラメータの値を確認する方法について説明します。続いて、適切なコード例を見つけて適用する操作について説明します。

トピック

- tls パラメータの値の確認
- TLS が有効な場合の接続
- TLS が無効な場合の接続

`tls` パラメータの値の確認

クラスターで TLS が有効になっているかどうかを判断するには、 AWS Management Console または のいずれかを使用して 2 ステップのプロセスを実行します AWS CLI。

1. クラスターを管理しているパラメータグループを確認します。

   Using the AWS Management Console

   1. にサインインし AWS Management Console、「https://https://console.aws.amazon.com/docdb://www.com で Amazon DocumentDB コンソールを開きます。
   2. 左のナビゲーションペインで [Clusters] (クラスター) を選択します。
   3. クラスターのリストで、クラスターの名前を選択します。
   4. 結果のページには、選択したクラスターの詳細が表示されます。[Configuration (設定)] タブを選択します。Configurations and status セクションで、クラスターパラメータグループの下にパラメータグループの名前を見つけます。

   Using the AWS CLI

   次の AWS CLI コードは、クラスターを管理しているパラメータを決定します。*`sample-cluster`* は、必ずクラスターの名前に置き換えてください。

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

このオペレーションによる出力は、次のようになります。

```
[
       [
           "sample-cluster",
           "sample-parameter-group"
       ]
]
```

2. クラスターの **tls** パラメータグループでパラメータの値を確認します。

Using the AWS Management Console

1. ナビゲーションペインで、[パラメータグループ] を選択します。
2. クラスターパラメータグループウィンドウで、ステップ 1d からクラスターパラメータグループ名を選択します。
3. 結果ページに、クラスターパラメータグループのパラメータが表示されます。ここで `tls` パラメータの値を確認できます。このパラメータの変更については、「Amazon DocumentDB クラスターパラメータグループを変更する」を参照してください。

Using the AWS CLI

`describe-db-cluster-parameters` AWS CLI コマンドを使用して、クラスターパラメータグループ内のパラメータの詳細を表示できます。

- **`--describe-db-cluster-parameters`** — パラメータグループ内のすべてのパラメータとその詳細の一覧を表示するには。
  - **`--db-cluster-parameter-group name`** — 必須。クラスターパラメータグループの名前。

次の例では、各`###############`をクラスターの情報に置き換えます。

```
aws docdb describe-db-cluster-parameters \
```

```
    --db-cluster-parameter-group-name sample-parameter-group
```

このオペレーションによる出力は、次のようになります。

```
{
        "Parameters": [
            {
                "ParameterName": "profiler_threshold_ms",
                "ParameterValue": "100",
                "Description": "Operations longer than profiler_threshold_ms
 will be logged",
                "Source": "system",
                "ApplyType": "dynamic",
                "DataType": "integer",
                "AllowedValues": "50-2147483646",
                "IsModifiable": true,
                "ApplyMethod": "pending-reboot"
            },
            {
                "ParameterName": "tls",
                "ParameterValue": "disabled",
                "Description": "Config to enable/disable TLS",
                "Source": "user",
                "ApplyType": "static",
                "DataType": "string",
                "AllowedValues": "disabled,enabled,fips-140-3",
                "IsModifiable": true,
                "ApplyMethod": "pending-reboot"
            }
        ]
}
```

> **Note**
>
> Amazon DocumentDBは、Amazon DocumentDB 5.0 (エンジンバージョン 3.0.3727) クラスター以降、ca-central-1、us-west-2、us-east-1、us-east-2、us-gov-east-1、us-gov-west-1 リージョンで FIPS 140-3 エンドポイントをサポートします。

`tls` パラメータの値を確認したら、続いて以下のセクションのコード例のうち 1 つを使用してクラスターに接続します。

- TLS が有効な場合の接続
- TLS が無効な場合の接続

TLS が有効な場合の接続

TLS が有効化されている Amazon DocumentDB クラスターにプログラムで接続するためのコード例を表示するには、使用する言語に該当するタブを選択してください。

転送中のデータを暗号化するには、以下のオペレーションを使用して、 `global-bundle.pem` という Amazon DocumentDB のパブリックキーをダウンロードします。

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

アプリケーションが Microsoft Windows 上にあり、PKCS7 ファイルが必要な場合は、PKCS7 証明書バンドルをダウンロードできます。このバンドルには、https://truststore.pki.rds.amazonaws.com/global/global-bundle.p7b にある中間証明書とルート証明書の両方が含まれています。

Python

次のコードは、TLS が有効になっているときに Python を使用して Amazon DocumentDB に接続する方法を示しています。

次の例では、各*###############*をクラスターの情報に置き換えます。

```
import pymongo
import sys

##Create a MongoDB client, open a connection to Amazon DocumentDB as a replica set
 and specify the read preference as secondary preferred
client = pymongo.MongoClient('mongodb://sample-user:password@sample-
cluster.node.us-east-1.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

##Specify the database to be used
db = client.sample_database

##Specify the collection to be used
col = db.sample_collection

##Insert a single document
col.insert_one({'hello':'Amazon DocumentDB'})
```

```
##Find the document that was previously written
x = col.find_one({'hello':'Amazon DocumentDB'})

##Print the result to the screen
print(x)

##Close the connection
client.close()
```

Node.js

次のコードは、TLS が有効になっているときに Python を使用して Amazon DocumentDB に接続する方法を示しています。

> **Important**
>
> バージョン 6.13.1 より前の Node.js ドライバーには既知の制限がありますが、現在 Amazon DocumentDB の IAM ID 認証ではサポートされていません。Node.js ドライバーを使用する Node.js ドライバーとツール (mongosh など) は、Node.js ドライバーバージョン 6.13.1 以降を使用するようにアップグレードする必要があります。

次の例では、各###############をクラスターの情報に置き換えます。

```
var MongoClient = require('mongodb').MongoClient

//Create a MongoDB client, open a connection to DocDB; as a replica set,
//  and specify the read preference as secondary preferred

var client = MongoClient.connect(
'mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/sample-database?
tls=true&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false',
{
  tlsCAFile: `global-bundle.pem` //Specify the DocDB; cert
},
function(err, client) {
    if(err)
        throw err;

    //Specify the database to be used
```

```
    db = client.db('sample-database');

    //Specify the collection to be used
    col = db.collection('sample-collection');

    //Insert a single document
    col.insertOne({'hello':'Amazon DocumentDB'}, function(err, result){
      //Find the document that was previously written
      col.findOne({'hello':'Amazon DocumentDB'}, function(err, result){
        //Print the result to the screen
        console.log(result);

        //Close the connection
        client.close()
      });
   });
});
```

PHP

次のコードは、TLS が有効になっているときに Python を使用して Amazon DocumentDB に接続する方法を示しています。

次の例では、各###############をクラスターの情報に置き換えます。

```
<?php
//Include Composer's autoloader
require 'vendor/autoload.php';

$TLS_DIR = "/home/ubuntu/global-bundle.pem";

//Create a MongoDB client and open connection to Amazon DocumentDB
$client = new MongoDB\Client("mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/?retryWrites=false", ["tls" => "true", "tlsCAFile"
 => $TLS_DIR ]);

//Specify the database and collection to be used
$col = $client->sampledatabase->samplecollection;

//Insert a single document
$result = $col->insertOne( [ 'hello' => 'Amazon DocumentDB'] );

//Find the document that was previously written
```

```
$result = $col->findOne(array('hello' => 'Amazon DocumentDB'));

//Print the result to the screen
print_r($result);
?>
```

Go

次のコードは、TLS が有効になっているときに Go を使用して Amazon DocumentDB に接続する方法を示しています。

> **Note**
>
> バージョン 1.2.1 現在、MongoDB Go ドライバーは sslcertificateauthorityfile にある最初の CA サーバー証明書のみを使用します。次のコード例では、この制限に対処するために、sslcertificateauthorityfile にあるすべてのサーバー証明書を、クライアントの作成時に使用されるカスタム TLS 設定に手動で追加しています。

次の例では、各###############をクラスターの情報に置き換えます。

```
package main

import (
 "context"
 "fmt"
 "log"
 "time"

 "go.mongodb.org/mongo-driver/bson"
 "go.mongodb.org/mongo-driver/mongo"
 "go.mongodb.org/mongo-driver/mongo/options"

 "io/ioutil"
 "crypto/tls"
 "crypto/x509"
 "errors"
)

const (
 // Path to the AWS CA file
 caFilePath = "global-bundle.pem"
```

```
 // Timeout operations after N seconds
 connectTimeout  = 5
 queryTimeout    = 30
 username        = "sample-user"
 password        = "password"
 clusterEndpoint = "sample-cluster.node.us-east-1.docdb.amazonaws.com:27017"

 // Which instances to read from
 readPreference = "secondaryPreferred"

 connectionStringTemplate = "mongodb://%s:%s@%s/sample-database?
tls=true&replicaSet=rs0&readpreference=%s"
)

func main() {

 connectionURI := fmt.Sprintf(connectionStringTemplate, username, password,
 clusterEndpoint, readPreference)

 tlsConfig, err := getCustomTLSConfig(caFilePath)
 if err != nil {
  log.Fatalf("Failed getting TLS configuration: %v", err)
 }

 client, err :=
 mongo.NewClient(options.Client().ApplyURI(connectionURI).SetTLSConfig(tlsConfig))
 if err != nil {
  log.Fatalf("Failed to create client: %v", err)
 }

 ctx, cancel := context.WithTimeout(context.Background(),
 connectTimeout*time.Second)
 defer cancel()

 err = client.Connect(ctx)
 if err != nil {
  log.Fatalf("Failed to connect to cluster: %v", err)
 }

 // Force a connection to verify our connection string
 err = client.Ping(ctx, nil)
 if err != nil {
  log.Fatalf("Failed to ping cluster: %v", err)
```

```
 }

 fmt.Println("Connected to DocumentDB!")

 collection := client.Database("sample-database").Collection("sample-collection")

 ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
 defer cancel()

 res, err := collection.InsertOne(ctx, bson.M{"name": "pi", "value": 3.14159})
 if err != nil {
  log.Fatalf("Failed to insert document: %v", err)
 }

 id := res.InsertedID
 log.Printf("Inserted document ID: %s", id)

 ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
 defer cancel()

 cur, err := collection.Find(ctx, bson.D{})

 if err != nil {
  log.Fatalf("Failed to run find query: %v", err)
 }
 defer cur.Close(ctx)

 for cur.Next(ctx) {
  var result bson.M
  err := cur.Decode(&result)
  log.Printf("Returned: %v", result)

  if err != nil {
   log.Fatal(err)
  }
 }

 if err := cur.Err(); err != nil {
  log.Fatal(err)
 }

}

func getCustomTLSConfig(caFile string) (*tls.Config, error) {
```

```
tlsConfig := new(tls.Config)
certs, err := ioutil.ReadFile(caFile)

if err != nil {
 return tlsConfig, err
}

tlsConfig.RootCAs = x509.NewCertPool()
ok := tlsConfig.RootCAs.AppendCertsFromPEM(certs)

if !ok {
 return tlsConfig, errors.New("Failed parsing pem file")
}

return tlsConfig, nil
```

Java

Java アプリケーションから TLS 対応の Amazon DocumentDB クラスターに接続する場合、プログラムは AWSが提供する認証機関 (CA) ファイルを使用して接続を検証する必要があります。Amazon RDS CA 証明書を使用するには、以下の手順に従ってください。

1. Amazon RDS CA ファイルを https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem ダウンロード。
2. 次のコマンドを実行し、ファイルに含まれている CA 認定を使用して信頼ストアを作成します。*truststore-password* は必ず別のものに変更してください。古い CA 認定 (rds-ca-2015-root.pem) と新しい CA 認定 (rds-ca-2019-root.pem) の両方を含む信頼ストアにアクセスする場合は、認定バンドルを信頼ストアにインポートできます。

   Linux オペレーティングシステムで、証明書バンドルを信頼ストアにインポートするサンプルシェルスクリプトを次に示します。次の例では*###############*をユーザー自身の情報で置き換えます。特に、サンプルディレクトリ「*mydir*」がスクリプトに含まれている場合は、このタスク用に作成したディレクトリに置き換えます。

   ```
   mydir=/tmp/certs
   truststore=${mydir}/rds-truststore.jks
   storepassword=truststore-password

   curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem" >
    ${mydir}/global-bundle.pem
   ```

```
awk 'split_after == 1 {n++;split_after=0} /-----END CERTIFICATE-----/
 {split_after=1}{print > "rds-ca-" n ".pem"}' < ${mydir}/global-bundle.pem

for CERT in rds-ca-*; do
  alias=$(openssl x509 -noout -text -in $CERT | perl -ne 'next unless /
Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep
 Alias | cut -d " " -f3- | while read alias
do
   expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword}
 -alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print
 "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

macOS で証明書バンドルを信頼ストアにインポートするサンプルシェルスクリプトを次に示します。

```
mydir=/tmp/certs
truststore=${mydir}/rds-truststore.jks
storepassword=truststore-password

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem" >
 ${mydir}/global-bundle.pem
split -p "-----BEGIN CERTIFICATE-----" ${mydir}/global-bundle.pem rds-ca-

for CERT in rds-ca-*; do
  alias=$(openssl x509 -noout -text -in $CERT | perl -ne 'next unless /
Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
```

```
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep
 Alias | cut -d " " -f3- | while read alias
do
   expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword}
 -alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print
 "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

3. Amazon DocumentDB クラスターに接続する前に、プログラムで keystore を使用するために、アプリケーションで次のシステムプロパティを設定します。

```
javax.net.ssl.trustStore: truststore
javax.net.ssl.trustStorePassword: truststore-password;
```

4. 次のコードは、TLS が有効になっているときに Java を使用して Amazon DocumentDB に接続する方法を示しています。

   次の例では、各###############をクラスターの情報に置き換えます。

```
package com.example.documentdb;

import com.mongodb.client.*;
import org.bson.Document;

public final class Test {
    private Test() {
    }
    public static void main(String[] args) {

        String template = "mongodb://%s:%s@%s/sample-database?
ssl=true&replicaSet=rs0&readpreference=%s";
        String username = "sample-user";
        String password = "password";
        String clusterEndpoint = "sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
```

```
        String readPreference = "secondaryPreferred";
        String connectionString = String.format(template, username, password,
clusterEndpoint, readPreference);

        String truststore = "truststore";
        String truststorePassword = "truststore-password";

        System.setProperty("javax.net.ssl.trustStore", truststore);
        System.setProperty("javax.net.ssl.trustStorePassword",
truststorePassword);

        MongoClient mongoClient = MongoClients.create(connectionString);

        MongoDatabase testDB = mongoClient.getDatabase("sample-database");
        MongoCollection<Document> numbersCollection =
testDB.getCollection("sample-collection");

        Document doc = new Document("name", "pi").append("value", 3.14159);
        numbersCollection.insertOne(doc);

        MongoCursor<Document> cursor = numbersCollection.find().iterator();
        try {
            while (cursor.hasNext()) {
                System.out.println(cursor.next().toJson());
            }
        } finally {
            cursor.close();
        }

    }
}
```

C# / .NET

次のコードは、TLS が有効になっているときに C# / .NET を使用して Amazon DocumentDB に接続する方法を示しています。

次の例では、各##############をクラスターの情報に置き換えます。

```
using System;
using System.Text;
using System.Linq;
using System.Collections.Generic;
```

```
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using MongoDB.Driver;
using MongoDB.Bson;

namespace DocDB
{
    class Program
    {
        static void Main(string[] args)
        {
            string template = "mongodb://{0}:{1}@{2}/sampledatabase?
tls=true&replicaSet=rs0&readpreference={3}";
            string username = "sample-user";
            string password = "password";
            string readPreference = "secondaryPreferred";
            string clusterEndpoint="sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
            string connectionString = String.Format(template, username, password,
 clusterEndpoint, readPreference);

            string pathToCAFile = "<PATH/global-bundle.p7b_file>";

            // ADD CA certificate to local trust store
            // DO this once - Maybe when your service starts
            X509Store localTrustStore = new X509Store(StoreName.Root);
            X509Certificate2Collection certificateCollection = new
 X509Certificate2Collection();
            certificateCollection.Import(pathToCAFile);
            try
            {
                localTrustStore.Open(OpenFlags.ReadWrite);
                localTrustStore.AddRange(certificateCollection);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Root certificate import failed: " + ex.Message);
                throw;
            }
            finally
            {
                localTrustStore.Close();
            }
```

```
            var settings = MongoClientSettings.FromUrl(new
 MongoUrl(connectionString));
            var client = new MongoClient(settings);

            var database = client.GetDatabase("sampledatabase");
            var collection =
 database.GetCollection<BsonDocument>("samplecollection");
            var docToInsert = new BsonDocument { { "pi", 3.14159 } };
            collection.InsertOne(docToInsert);
        }
    }
}
```

MongoDB Shell

次のコードは、TLS が有効になっている場合に、最新バージョン、mongosh、または以前の mongo シェルバージョンを使用して Amazon DocumentDB に接続してクエリを実行する方法を示しています。

mongosh を使用して Amazon DocumentDB に接続する

> **Important**
>
> バージョン 6.13.1 より前の Node.js ドライバーには既知の制限がありますが、現在 Amazon DocumentDB の IAM ID 認証ではサポートされていません。Node.js ドライバーを使用する Node.js ドライバーとツール (mongosh など) は、Node.js ドライバーバージョン 6.13.1 以降を使用するようにアップグレードする必要があります。

次の例では、各*###############*をクラスターの情報に置き換えます。

```
mongosh --tls --host cluster-end-point:27017 --tlsCAFile  global-bundle.pem --
username sample-user --password password --retryWrites false
```

以前の mongo シェルバージョンを使用して Amazon DocumentDB に接続する

IAM を使用する場合は、以前のバージョンの mongo シェルを使用する必要があります。次のいずれかのコマンドオプションを入力します。

```
mongo --ssl --host cluster-end-point:27017 --sslCAFile global-bundle.pem --
username sample-user --password password
```

4.2 かそれ以降のバージョンの Mongo シェルを使用している場合は、次のコードを使用して接続します。再試行可能な書き込みは Amazon DocumentDB ではサポートされていません。例外: mongo シェル を使用している場合は、どのコード文字列にも `retryWrites=false` コマンドを含めないでください。デフォルトでは、再試行可能な書き込みは無効になっています。`retryWrites=false` を含めると、通常の読み取りコマンドでは失敗する可能性があります。

```
mongo --tls --host cluster-end-point:27017 --tlsCAFile global-bundle.pem --
username sample-user --password password
```

接続をテストする

1. 単一のドキュメントを挿入します。

   ```
   db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
   ```

2. 以前に挿入されたドキュメントを検索します。

   ```
   db.myTestCollection.find({'hello':'Amazon DocumentDB'})
   ```

R

次のコードは、TLS が有効になっているときに mongolite (https://jeroen.github.io/mongolite/) を使用して Amazon DocumentDB に接続する方法を示しています。

次の例では、各##############をクラスターの情報に置き換えます。

```
#Include the mongolite library.
library(mongolite)

mongourl <- paste("mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/test2?ssl=true&",
          "readPreference=secondaryPreferred&replicaSet=rs0", sep="")

#Create a MongoDB client, open a connection to Amazon DocumentDB as a replica
#   set and specify the read preference as secondary preferred
```

```
client <-  mongo(url = mongourl, options = ssl_options(weak_cert_validation = F, ca
 ="<PATH/global-bundle.pem>"))

#Insert a single document
str <- c('{"hello" : "Amazon DocumentDB"}')
client$insert(str)

#Find the document that was previously written
client$find()
```

Ruby

次のコードは、TLS が有効になっているときに Ruby を使用して Amazon DocumentDB に接続する方法を示しています。

次の例では、各*###############*をクラスターの情報に置き換えます。

```
require 'mongo'
require 'neatjson'
require 'json'
client_host = 'mongodb://sample-cluster.node.us-east-1.docdb.amazonaws.com:27017'
client_options = {
   database: 'test',
   replica_set: 'rs0',
   read: {:secondary_preferred => 1},
   user: 'sample-user',
   password: 'password',
   ssl: true,
   ssl_verify: true,
   ssl_ca_cert: 'PATH/global-bundle.pem',
   retry_writes: false
}


begin
   ##Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ##   replica set and specify the read preference as secondary preferred
   client = Mongo::Client.new(client_host, client_options)

   ##Insert a single document
   x = client[:test].insert_one({"hello":"Amazon DocumentDB"})

   ##Find the document that was previously written
```

```
   result = client[:test].find()

   #Print the document
   result.each do |document|
      puts JSON.neat_generate(document)
   end
end

#Close the connection
client.close
```

TLS が無効な場合の接続

TLS が無効化されている Amazon DocumentDB クラスターにプログラムで接続するためのコード例を表示するには、使用する言語のタブを選択してください。

Python

次のコードは、TLS が無効になっているときに Python を使用して Amazon DocumentDB に接続する方法を示しています。

次の例では、各*###############*をクラスターの情報に置き換えます。

```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a replica set
 and specify the read preference as secondary preferred

import pymongo
import sys

client = pymongo.MongoClient('mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/?
replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

##Specify the database to be used
db = client.sample_database

##Specify the collection to be used
col = db.sample_collection

##Insert a single document
col.insert_one({'hello':'Amazon DocumentDB'})
```

```
##Find the document that was previously written
x = col.find_one({'hello':'Amazon DocumentDB'})

##Print the result to the screen
print(x)

##Close the connection
client.close()
```

Node.js

次のコードは、TLS が無効になっているときに Node.js を使用して Amazon DocumentDB に接続する方法を示しています。

> **Important**
>
> バージョン 6.13.1 より前の Node.js ドライバーには既知の制限がありますが、現在 Amazon DocumentDB の IAM ID 認証ではサポートされていません。Node.js ドライバーを使用する Node.js ドライバーとツール (mongosh など) は、Node.js ドライバーバージョン 6.13.1 以降を使用するようにアップグレードする必要があります。

次の例では、各*###############*をクラスターの情報に置き換えます。

```
var MongoClient = require('mongodb').MongoClient;

//Create a MongoDB client, open a connection to Amazon DocumentDB as a replica set,
//  and specify the read preference as secondary preferred
var client = MongoClient.connect(
'mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/sample-database?
replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false',
{
  useNewUrlParser: true
},

function(err, client) {
    if(err)
        throw err;
    //Specify the database to be used
    db = client.db('sample-database');
```

```
    //Specify the collection to be used
    col = db.collection('sample-collection');

    //Insert a single document
    col.insertOne({'hello':'Amazon DocumentDB'}, function(err, result){
      //Find the document that was previously written
      col.findOne({'hello':'Amazon DocumentDB'}, function(err, result){
        //Print the result to the screen
        console.log(result);

        //Close the connection
        client.close()
      });
   });
});
```

PHP

次のコードは、TLS が無効になっているときに PHP を使用して Amazon DocumentDB に接続する方法を示しています。

次の例では、各*###############*をクラスターの情報に置き換えます。

```
<?php
//Include Composer's autoloader
require 'vendor/autoload.php';

//Create a MongoDB client and open connection to Amazon DocumentDB
$client = new MongoDB\Client("mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/?retryWrites=false");

//Specify the database and collection to be used
$col = $client->sampledatabase->samplecollection;

//Insert a single document
$result = $col->insertOne( [ 'hello' => 'Amazon DocumentDB'] );

//Find the document that was previously written
$result = $col->findOne(array('hello' => 'Amazon DocumentDB'));

//Print the result to the screen
print_r($result);
?>
```

Go

次のコードは、TLS が無効になっているときに Go を使用して Amazon DocumentDB に接続する方法を示しています。

次の例では、各###############をクラスターの情報に置き換えます。

```
package main

import (
 "context"
 "fmt"
 "log"
 "time"

 "go.mongodb.org/mongo-driver/bson"
 "go.mongodb.org/mongo-driver/mongo"
 "go.mongodb.org/mongo-driver/mongo/options"
)

const (
 // Timeout operations after N seconds
 connectTimeout  = 5
 queryTimeout    = 30
 username        = "sample-user"
 password        = "password"
 clusterEndpoint = "sample-cluster.node.us-east-1.docdb.amazonaws.com:27017"

 // Which instances to read from
 readPreference           = "secondaryPreferred"
 connectionStringTemplate = "mongodb://%s:%s@%s/sample-database?
replicaSet=rs0&readpreference=%s"
)

func main() {

 connectionURI := fmt.Sprintf(connectionStringTemplate, username, password,
 clusterEndpoint, readPreference)

 client, err := mongo.NewClient(options.Client().ApplyURI(connectionURI))
 if err != nil {
  log.Fatalf("Failed to create client: %v", err)
 }
```

```
ctx, cancel := context.WithTimeout(context.Background(),
connectTimeout*time.Second)
defer cancel()

err = client.Connect(ctx)
if err != nil {
 log.Fatalf("Failed to connect to cluster: %v", err)
}

// Force a connection to verify our connection string
err = client.Ping(ctx, nil)
if err != nil {
 log.Fatalf("Failed to ping cluster: %v", err)
}

fmt.Println("Connected to DocumentDB!")

collection := client.Database("sample-database").Collection("sample-collection")

ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
defer cancel()

res, err := collection.InsertOne(ctx, bson.M{"name": "pi", "value": 3.14159})
if err != nil {
 log.Fatalf("Failed to insert document: %v", err)
}

id := res.InsertedID
log.Printf("Inserted document ID: %s", id)

ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
defer cancel()

cur, err := collection.Find(ctx, bson.D{})

if err != nil {
 log.Fatalf("Failed to run find query: %v", err)
}
defer cur.Close(ctx)

for cur.Next(ctx) {
 var result bson.M
 err := cur.Decode(&result)
 log.Printf("Returned: %v", result)
```

```
  if err != nil {
   log.Fatal(err)
  }
 }

 if err := cur.Err(); err != nil {
  log.Fatal(err)
 }

}
```

Java

次のコードは、TLS が無効になっているときに Java を使用して Amazon DocumentDB に接続する方法を示しています。

次の例では、各##############をクラスターの情報に置き換えます。

```
package com.example.documentdb;

import com.mongodb.MongoClient;
import com.mongodb.MongoClientURI;
import com.mongodb.ServerAddress;
import com.mongodb.MongoException;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoCollection;
import org.bson.Document;


public final class Main {
    private Main() {
    }
    public static void main(String[] args) {

        String template = "mongodb://%s:%s@%s/sample-database?
replicaSet=rs0&readpreference=%s";
        String username = "sample-user";
        String password = "password";
        String clusterEndpoint = "sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
        String readPreference = "secondaryPreferred";
```

```
        String connectionString = String.format(template, username, password,
 clusterEndpoint, readPreference);

        MongoClientURI clientURI = new MongoClientURI(connectionString);
        MongoClient mongoClient = new MongoClient(clientURI);

        MongoDatabase testDB = mongoClient.getDatabase("sample-database");
        MongoCollection<Document> numbersCollection = testDB.getCollection("sample-
collection");

        Document doc = new Document("name", "pi").append("value", 3.14159);
        numbersCollection.insertOne(doc);

        MongoCursor<Document> cursor = numbersCollection.find().iterator();
        try {
            while (cursor.hasNext()) {
                System.out.println(cursor.next().toJson());
            }
        } finally {
            cursor.close();
        }

    }
}
```

C# / .NET

次のコードは、TLS が無効になっているときに C# / .NET を使用して Amazon DocumentDB に接続する方法を示しています。

次の例では、各###############をクラスターの情報に置き換えます。

```
using System;
using System.Text;
using System.Linq;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using MongoDB.Driver;
using MongoDB.Bson;

namespace CSharpSample
```

```
{
   class Program
    {
       static void Main(string[] args)
        {
           string template = "mongodb://{0}:{1}@{2}/sampledatabase?
replicaSet=rs0&readpreference={3}";
           string username = "sample-user";
           string password = "password";
           string clusterEndpoint = "sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
           string readPreference = "secondaryPreferred";
           string connectionString = String.Format(template, username, password,
 clusterEndpoint, readPreference);

           var settings = MongoClientSettings.FromUrl(new
 MongoUrl(connectionString));
           var client = new MongoClient(settings);

           var database = client.GetDatabase("sampledatabase");
           var collection =
 database.GetCollection<BsonDocument>("samplecollection");
           var docToInsert = new BsonDocument { { "pi", 3.14159 } };
            collection.InsertOne(docToInsert);
        }
    }
}
```

MongoDB Shell

次のコードは、TLS が無効になっている場合に、最新バージョン、mongosh、または以前の mongo シェルバージョンを使用して Amazon DocumentDB に接続してクエリを実行する方法を示しています。

mongosh を使用して Amazon DocumentDB に接続する

> **Important**
>
> バージョン 6.13.1 より前の Node.js ドライバーには既知の制限がありますが、現在 Amazon DocumentDB の IAM ID 認証ではサポートされていません。Node.js ドライバーを使用する Node.js ドライバーとツール (mongosh など) は、Node.js ドライバーバージョン 6.13.1 以降を使用するようにアップグレードする必要があります。

次の例では、各##############をクラスターの情報に置き換えます。

```
mongosh --host cluster-end-point:27017 --username sample-user --password password --
retryWrites false
```

以前の mongo シェルバージョンを使用して Amazon DocumentDB に接続する

IAM を使用する場合は、以前のバージョンの mongo シェルを使用する必要があります。次のいずれかのコマンドオプションを入力します。

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

4.2 かそれ以降のバージョンの Mongo シェルを使用している場合は、次のコードを使用して接続します。再試行可能な書き込みは Amazon DocumentDB ではサポートされていません。例外: mongo シェル を使用している場合は、どのコード文字列にも `retryWrites=false` コマンドを含めないでください。デフォルトでは、再試行可能な書き込みは無効になっています。`retryWrites=false` を含めると、通常の読み取りコマンドでは失敗する可能性があります。

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

接続をテストする

1. 単一のドキュメントを挿入します。

   ```
   db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
   ```

2. 以前に挿入されたドキュメントを検索します。

   ```
   db.myTestCollection.find({'hello':'Amazon DocumentDB'})
   ```

R

次のコードは、TLS が無効になっているときに mongolite (https://jeroen.github.io/mongolite/) を使用して Amazon DocumentDB に接続する方法を示しています。

次の例では、各##############をクラスターの情報に置き換えます。

```
#Include the mongolite library.
```

```
library(mongolite)

#Create a MongoDB client, open a connection to Amazon DocumentDB as a replica
#   set and specify the read preference as secondary preferred
client <- mongo(url = "mongodb://sample-user:password@sample-
cluster.node.us-east-1.docdb.amazonaws.com:27017/sample-database?
readPreference=secondaryPreferred&replicaSet=rs0")

##Insert a single document
str <- c('{"hello" : "Amazon DocumentDB"}')
client$insert(str)

##Find the document that was previously written
client$find()
```

Ruby

次のコードは、TLS が無効になっているときに Ruby を使用して Amazon DocumentDB に接続する方法を示しています。

次の例では、各###############をクラスターの情報に置き換えます。

```
require 'mongo'
require 'neatjson'
require 'json'
client_host = 'mongodb://sample-cluster.node.us-east-1.docdb.amazonaws.com:27017'
client_options = {
   database: 'test',
   replica_set: 'rs0',
   read: {:secondary_preferred => 1},
   user: 'sample-user',
   password: 'password',
   retry_writes: false
}

begin
   ##Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ##   replica set and specify the read preference as secondary preferred
   client = Mongo::Client.new(client_host, client_options)

   ##Insert a single document
   x = client[:test].insert_one({"hello":"Amazon DocumentDB"})
```

```
   ##Find the document that was previously written
   result = client[:test].find()

   #Print the document
   result.each do |document|
      puts JSON.neat_generate(document)
   end
end

#Close the connection
client.close
```

Amazon VPC の外部から Amazon DocumentDB クラスターへの接続

Amazon DocumentDB (MongoDB 互換) クラスターは、Amazon VPC (Amazon Virtual Private Cloud) 内にデプロイされます。これらは、Amazon EC2 インスタンス、または同じ Amazon VPC にデプロイされている他の AWS サービスから直接アクセスできます。さらに、Amazon DocumentDB には、VPC ピア接続を介して、同じリージョンまたは他のリージョンの異なる VPCs の EC2 インスタンス AWS リージョン または他の AWS サービスからアクセスできます。

ただし、お客様のユースケースで、お客様 (またはお客様のアプリケーション) が、クラスターの VPC の外部から Amazon DocumentDB リソースにアクセスする必要があるとします。その場合は、SSH トンネリング (ポート転送 とも呼ばれる) を使用して、Amazon DocumentDB リソースにアクセスできます。

SSH トンネリングに関する詳細な説明は、このトピックでは扱いません。SSH トンネリングの詳細については、次のドキュメントを参照してください。

- SSH トンネル
- SSH ポート転送の例、特に「ローカル転送」セクション

SSH トンネルを作成するには、Amazon DocumentDB クラスターと同じ Amazon VPC で実行中の Amazon EC2 インスタンスが必要です。クラスターと同じ VPC で既存の EC2 インスタンスを使用するか、新しい EC2 インスタンスを作成できます。詳細については、以下でオペレーティングシステムに該当するトピックを参照してください。

- Amazon EC2 Linux インスタンスの使用開始
- Amazon EC2 Windows インスタンスの開始方法

通常、以下のコマンドを使用して EC2 インスタンスに接続します。

```
ssh -i "ec2Access.pem" ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com
```

その場合、ローカルコンピュータで以下のコマンドを実行することにより Amazon DocumentDB クラスターsample-cluster.node.us-east-1.docdb.amazonaws.com に SSH トンネルを設定できます。ローカルポートの転送を行う場合は、-L フラグを使用します。SSH トンネルを使用する場合は、クラスターエンドポイントを使用してクラスターに接続することをお勧めします。レプリカセットモード (接続文字列に replicaSet=rs0 を指定) で接続するとエラーが発生します。

```
ssh -i "ec2Access.pem" -L 27017:sample-cluster.node.us-east-1.docdb.amazonaws.com:27017
 ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com -N
```

SSH トンネルが作成されると、localhost:27017 に対して発行するすべてのコマンドは、Amazon VPC で実行中の Amazon DocumentDB クラスター sample-cluster に転送されます。Amazon DocumentDB クラスターで TLS (Transport Layer Security) が有効になっている場合は、Amazon DocumentDB のhttps://truststore.pki.rds.amazonaws.com/global/global-bundle.pem パブリックキーをダウンロードする必要があります。次のオペレーションでは、このファイルをダウンロードします。

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

> **Note**
>
> TLS は、新しい Amazon DocumentDB クラスターに対してデフォルトで有効になっています。ただし、無効にすることは可能です。詳細については、「Amazon DocumentDB クラスター TLS 設定の管理」を参照してください。

Amazon VPC の外部から Amazon DocumentDB クラスターに接続するには、以下のコマンドを使用します。

```
mongo --sslAllowInvalidHostnames --ssl --sslCAFile global-bundle.pem --username
 <yourUsername> --password <yourPassword>
```

レプリカセットとしての Amazon DocumentDB に接続する

Amazon DocumentDB (MongoDB 互換）で開発する場合は、レプリカセットとしてクラスターに接続し、ドライバーに組み込まれている読み取り設定機能を使用してレプリカインスタンスに読み取りを分散することをお勧めします。このセクションでは、その意味についてさらに詳しく説明し、例として SDK for Python を使用して、レプリカセットとして Amazon DocumentDB クラスターに接続する方法について説明します。

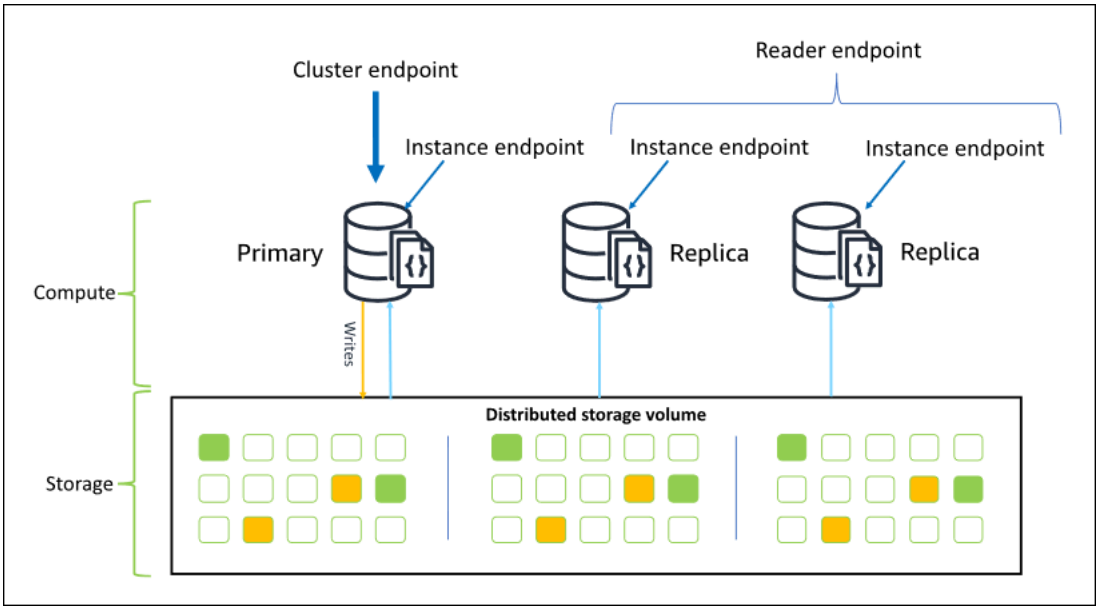
Amazon DocumentDB には、クラスターへの接続に使用できる 3 つのエンドポイントがあります。

- クラスターエンドポイント
- リーダーエンドポイント
- インスタンスエンドポイント

ほとんどの場合、Amazon DocumentDB に接続するときは、クラスターエンドポイントを使用することをお勧めします。これは、次の図に示すように、クラスター内のプライマリインスタンスをポイントする CNAME です。

SSH トンネルを使用する場合は、クラスターエンドポイントを使用してクラスターに接続することをお勧めします。レプリカセットモード (接続文字列に `replicaSet=rs0` を指定) で接続するとエラーが発生します。

> **Note**
>
> Amazon DocumentDB ウェブサイトのエンドポイントの詳細については、「Amazon DocumentDB エンドポイント」を参照してください。



クラスターエンドポイントを使用して、レプリカセットモードでクラスターに接続できます。その後、組み込まれた読み込み設定ドライバー機能を使用できます。次の例では、/?replicaSet=rs0 を指定することで、レプリカセットとして接続することを SDK を示します。/?replicaSet=rs0' を省略すると、クライアントはすべてのリクエストをプライマリインスタンスであるクラスターエンドポイントにルーティングします。

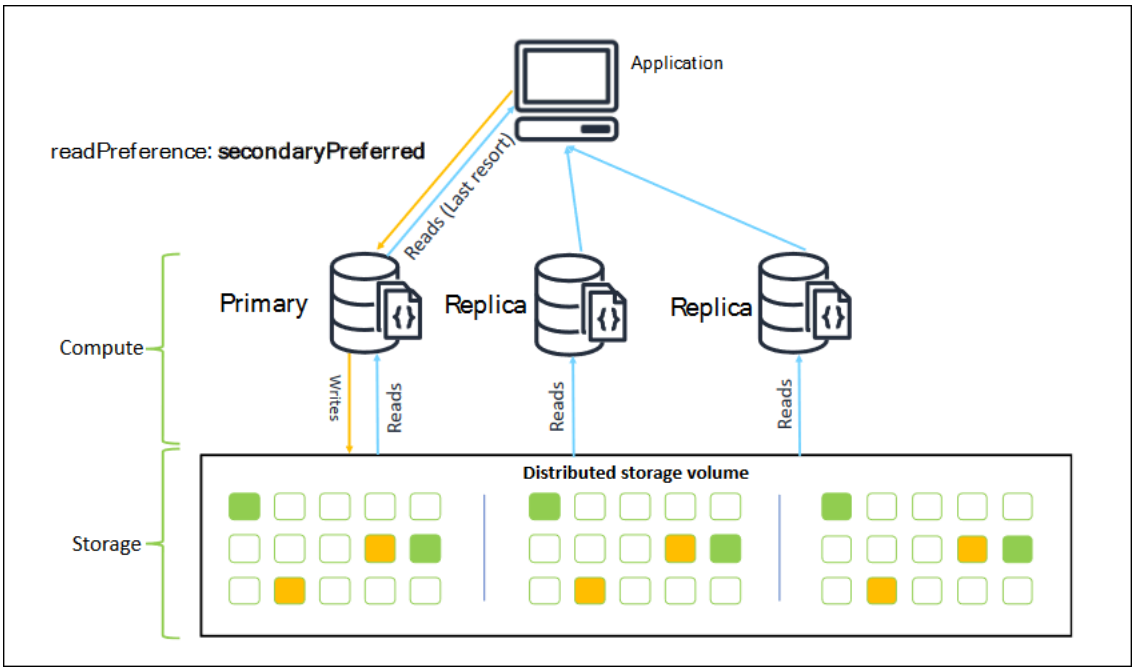
```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ## replica set and specify the read preference as secondary preferred
   client = pymongo.MongoClient('mongodb://<user-name>:<password>@mycluster.node.us-
east-1.docdb.amazonaws.com:27017/?replicaSet=rs0')
```

レプリカセットとして接続する利点は、インスタンスがクラスターに対して追加または削除されるタイミングを含め、SDK がクラスタートポグラフィを自動的に検出できることです。その後、レプリカインスタンスに読み込みリクエストをルーティングすることで、クラスターをより効率的に使用できます。

レプリカセットとして接続するときは、接続の readPreference を指定できます。secondaryPreferred の読み込み設定を指定した場合、クライアントは (以下の図のように) 読み取りクエリをレプリカにルーティングし、書き込みクエリをプライマリインスタンスにルーティングします。これにより、クラスターリソースをより適切に利用できます。詳細については、「読み込み設定のオプション」を参照してください。

```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ##   replica set and specify the read preference as secondary preferred
```

```
client = pymongo.MongoClient('mongodb://<user-name>:<password>@mycluster.node.us-
east-1.docdb.amazonaws.com:27017/?replicaSet=rs0&readPreference=secondaryPreferred')
```



Amazon DocumentDBレプリカからの読み込みには結果整合性があります。プライマリに書き込まれたのと同じ順序でデータを返し、50 ミリ秒未満のレプリケーション遅延もよくあります。Amazon CloudWatch メトリクス `DBInstanceReplicaLag` と `DBClusterReplicaLagMaximum` を使用して、クラスターのレプリカラグをモニタリングできます。詳細については、「Amazon DocumentDB と CloudWatch のモニタリング」を参照してください。

従来のモノリシックデータベースアーキテクチャとは異なり、Amazon DocumentDB はストレージとコンピューティングを分離します。この最新のアーキテクチャでは、レプリカインスタンスの読み取りのスケールをお勧めします。レプリカインスタンスの読み込みは、プライマリインスタンスからレプリケートされる書き込みをブロックしません。クラスターに最大 15 個のリードレプリカインスタンスを追加し、1 秒あたり数百万回の読み込みにスケールアウトできます。

レプリカセットとして接続し、レプリカに読み取りを分散する主な利点は、アプリケーションの作業に利用できるクラスター内のリソース全体が増加することです。ベストプラクティスとして、レプリカセットとして接続することをお勧めします。さらに、以下のシナリオでは最も一般的にお勧めします。

- プライマリでほぼ 100% の CPU を使用している。
- バッファキャッシュヒット率がほぼゼロである。
- 個別のインスタンスで接続またはカーソルの制限に達する。

クラスターインスタンスサイズのスケールアップは 1 つの選択肢であり、場合によっては、これがクラスターをスケーリングするための最良の方法です。ただし、クラスターにすでに存在するレプリカをより適切に使用する方法も考慮する必要があります。これにより、より大きいインスタンスタイプを使用することでコストが増大することなく、スケールを増加できます。また、CloudWatch アラームを使用してこれらの制限 (`CPUUtilization`、`DatabaseConnections`、および `BufferCacheHitRatio`) をモニタリングおよびアラートし、リソースが頻繁に使用されているタイミングを把握することをお勧めします。

詳細については、以下の各トピックを参照してください。

- Amazon DocumentDB のベストプラクティス
- Amazon DocumentDB のクォータと制限事項

クラスター接続の使用

クラスター内のすべての接続を使用するシナリオを検討します。たとえば、`r5.2xlarge` インスタンスの接続は 4,500 個 (および開いているカーソルは 450 個) に制限されています。3 つのインスタンスの Amazon DocumentDB クラスターを作成し、クラスターエンドポイントを使用してプライマリインスタンスにのみ接続する場合、開いている接続とカーソルのクラスター制限はそれぞれ 4,500 と 450 になります。コンテナでスピンアップする多くのワーカーを使用するアプリケーションを構築する場合、これらの制限に達する可能性があります。コンテナは、一度に多数の接続を開き、クラスターを飽和させます。

代わりに、レプリカセットとして Amazon DocumentDB クラスターに接続し、レプリカインスタンスに読み取りを分散できます。その後、クラスターで使用できる接続とカーソルの数をそれぞれ 13,500 と 1,350 に実質的に 3 倍にできます。クラスターにインスタンスを追加すると、読み取りワークロードの接続とカーソルの数が増えるだけです。クラスターへの書き込みの接続数を増やす必要がある場合は、インスタンスサイズを大きくすることをお勧めします。

> **Note**
>
> `large` インスタンス、`xlarge` インスタンス、および `2xlarge` インスタンスの接続数は、インスタンスサイズが最大 4,500 に達するまで増加します。`4xlarge` インスタンス以上の場合、インスタンスあたりの最大接続数は 4,500 です。インスタンスタイプごとの制限の詳細については、「インスタンス制限」を参照してください。

通常、secondary の読み込み設定を使用してクラスターに接続することはお勧めしません。これは、クラスターにレプリカインスタンスがない場合、読み込みが失敗するためです。例えば、1 つのプライマリと 1 つのレプリカを持つ 2 つのインスタンスの Amazon DocumentDB クラスターがあるとします。レプリカに問題がある場合、secondary に設定されている接続プールからの読み取りリクエストは失敗します。secondaryPreferred の利点は、クライアントが接続する適切なレプリカインスタンスを見つけることができない場合、読み取りのためにプライマリにフォールバックすることです。

複数の接続プール

シナリオによっては、アプリケーションの読み取りに、書き込み後読み取り整合性が必要です。これは、Amazon DocumentDB のプライマリインスタンスからのみ提供できます。このようなシナリオでは、2 つのクライアント接続プールを作成できます。1 つは書き込み用、もう 1 つは書き込み後読み込み整合性を必要とする読み込み用です。そのためには、コードは次のようになります。

```
## Create a MongoDB client,
##    open a connection to Amazon DocumentDB as a replica set and specify the
 readPreference as primary
clientPrimary = pymongo.MongoClient('mongodb://<user-
name>:<password>@mycluster.node.us-east-1.docdb.amazonaws.com:27017/?
replicaSet=rs0&readPreference=primary')

## Create a MongoDB client,
##    open a connection to Amazon DocumentDB as a replica set and specify the
 readPreference as secondaryPreferred
secondaryPreferred = pymongo.MongoClient('mongodb://<user-
name>:<password>@mycluster.node.us-east-1.docdb.amazonaws.com:27017/?
replicaSet=rs0&readPreference=secondaryPreferred')
```

もう 1 つのオプションは、1 つの接続プールを作成し、特定のコレクションの読み込み設定を上書きすることです。

```
##Specify the collection and set the read preference level for that collection
col = db.review.with_options(read_preference=ReadPreference.SECONDARY_PREFERRED)
```

概要

クラスター内のリソースを有効に使用するために、レプリカセットモードを使用してクラスターに接続することをお勧めします。アプリケーションに適している場合は、レプリカインスタンスに読み取りを分散することで、アプリケーションの読み取りをスケーリングできます。

Studio 3T から Amazon DocumentDB クラスターに接続する

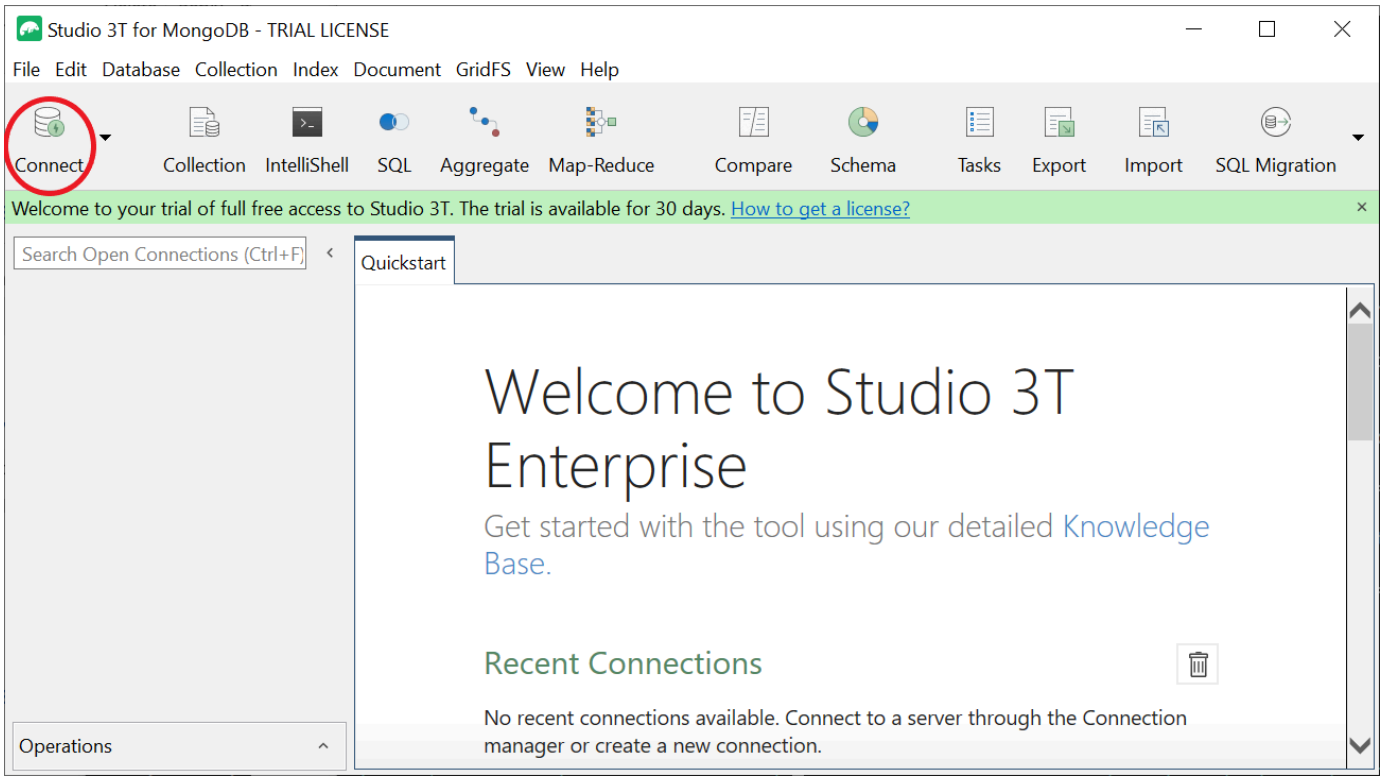
Studio 3T は、MongoDB で作業するデベロッパーやデータエンジニアにとって人気のある GUI と IDE です。データのツリー、表、および JSON ビュー、CSV、JSON、SQL および BSON/MongoDump での簡単なインポート/エクスポート、柔軟なクエリオプション、視覚的なドラッグアンドドロップ UI、自動補完機能を備えた組み込みの Mongo シェル、集計パイプラインエディタ、および SQL クエリのサポートを提供しています。

前提条件

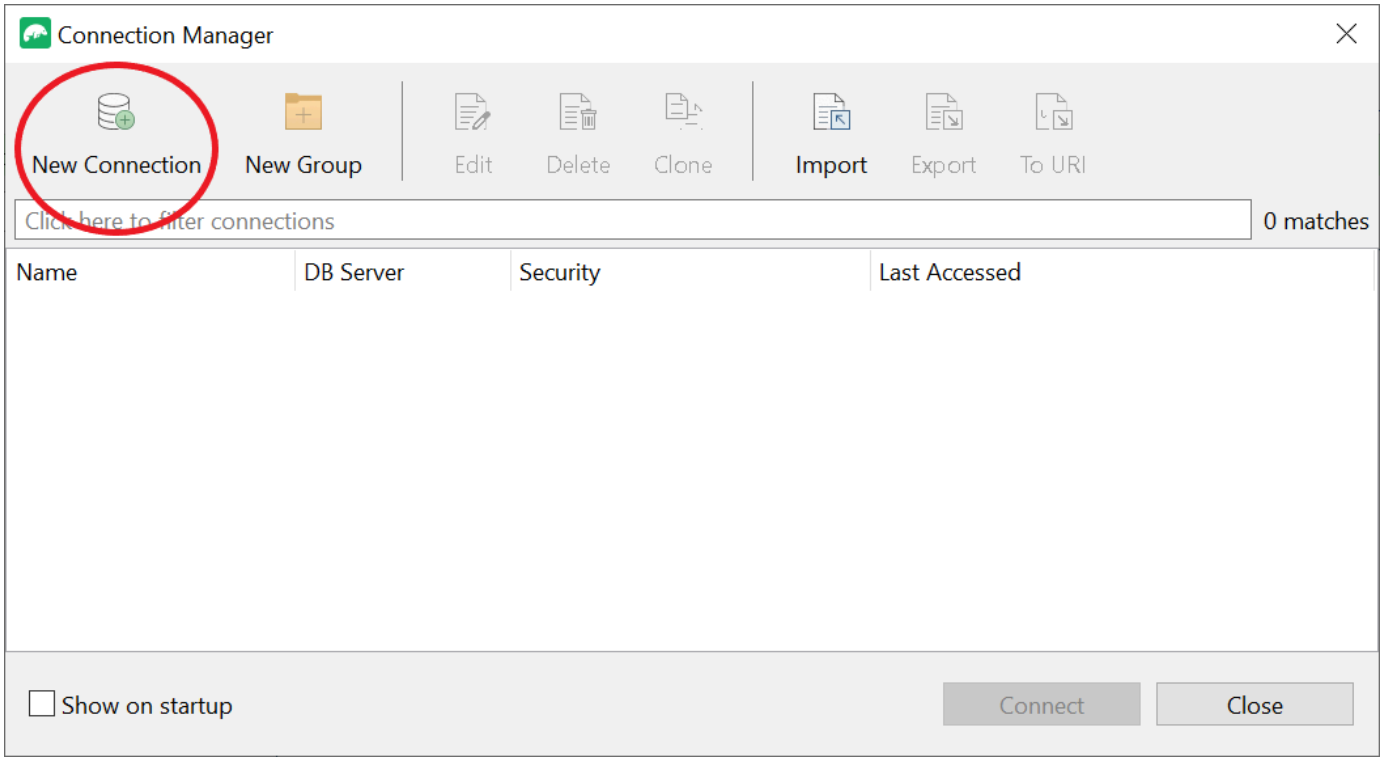
- Amazon DocumentDB クラスターで Amazon EC2 を「踏み台ホスト」として利用するシステムをまだ構築していないお客様は、「Amazon EC2 に接続する」の手順に従ってください。
- Studio 3T をお持ちでない場合は、ダウンロードしてインストールします。

Studio 3T に接続する

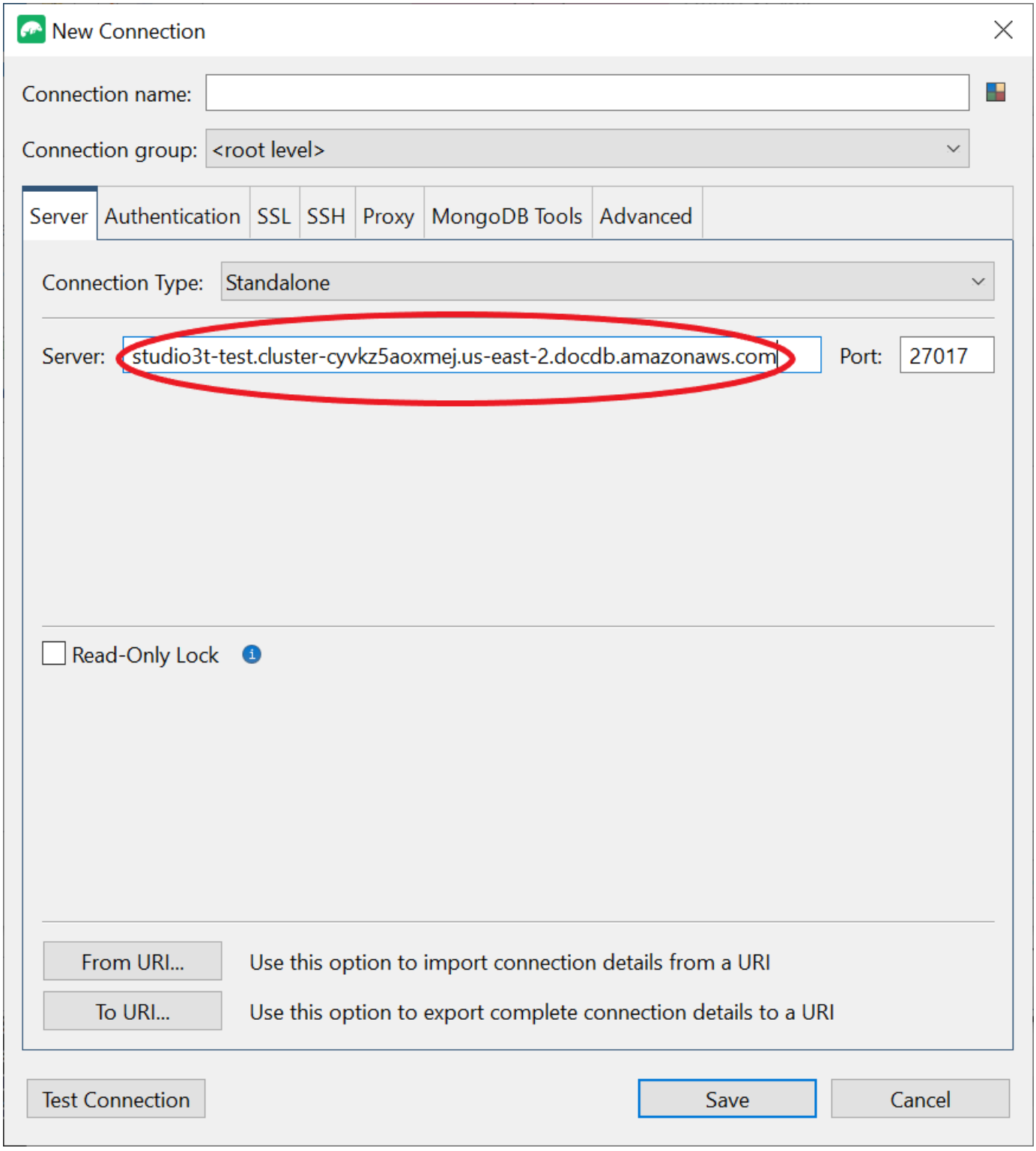
1. ツールバーの左上隅にある 接続 を選択します。



2. ツールバーの左上隅にある 新規接続 を選択します。

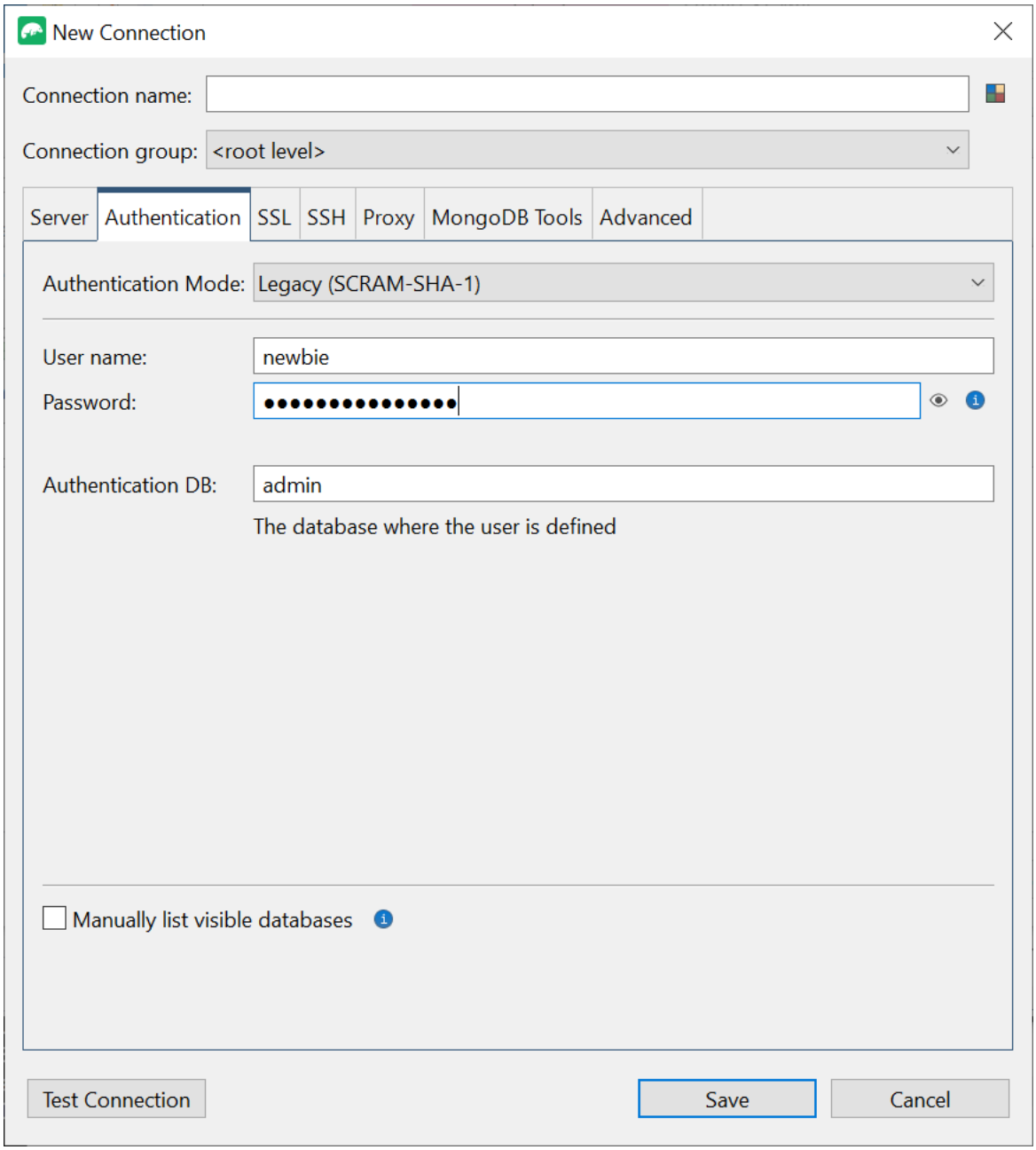


3. サーバー タブの サーバー フィールドに、クラスターエンドポイント情報を入力します。

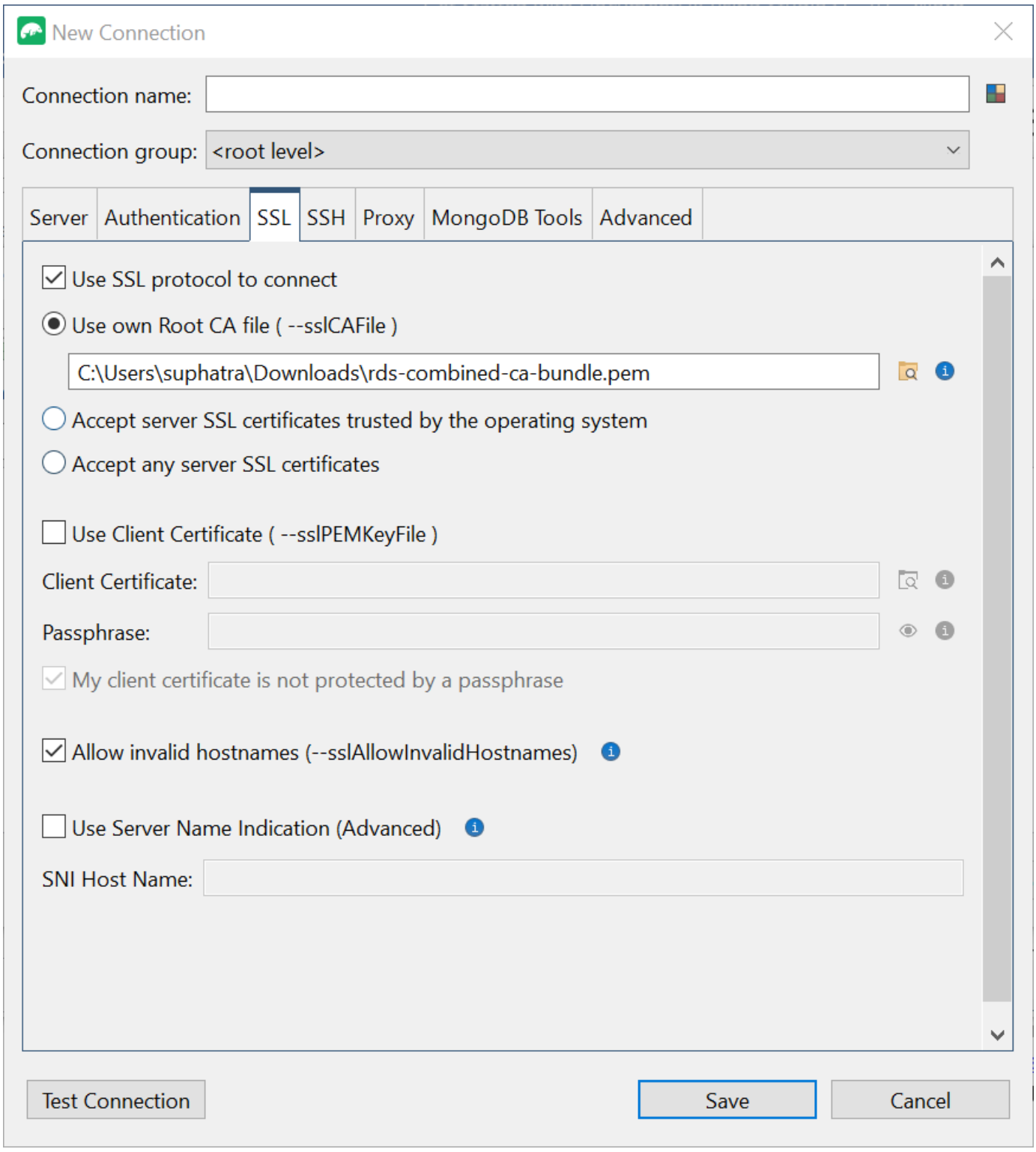


> **Note**
>
> クラスターエンドポイントが見つからない場合。こちらの ステップに従ってください。

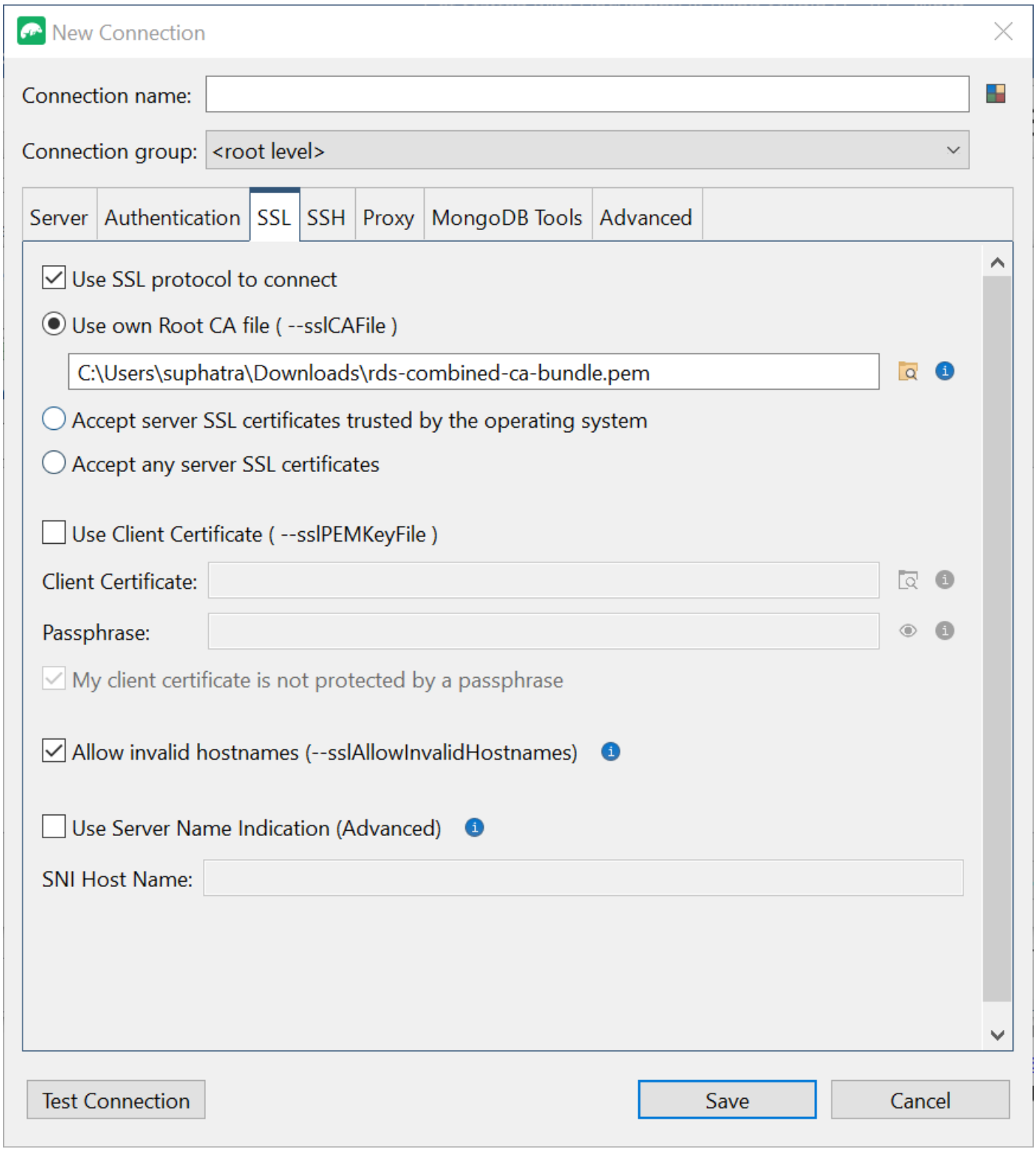
4. 認証 タブを選択し、認証モード のドロップダウンメニューで レガシー を選択します。



5. ユーザー名 フィールドと パスワード フィールドにユーザーネームと資格情報を入力します。
6. SSL タブを選択し、SSLプロトコルを使用して接続する チェックボックスをオンにします。



7. 独自のルート CA ファイルを使用する を選択します。次に、Amazon DocumentDB 証明書を追加します (DocumentDB クラスターで SSL が無効になっている場合は、このステップをスキップできます)。無効なホスト名 を許可するには、チェックボックスをオンにします。



Note

証明書を持っていない場合。次のコマンドでダウンロードできます。

```
wget https://truststore.pki.rds.amazonaws.com/global/global-
bundle.pem
```

8. Amazon VPC 外のクライアントマシンから接続する場合は、SSH トンネルを作成する必要があります。SSH タブでこれを行います。

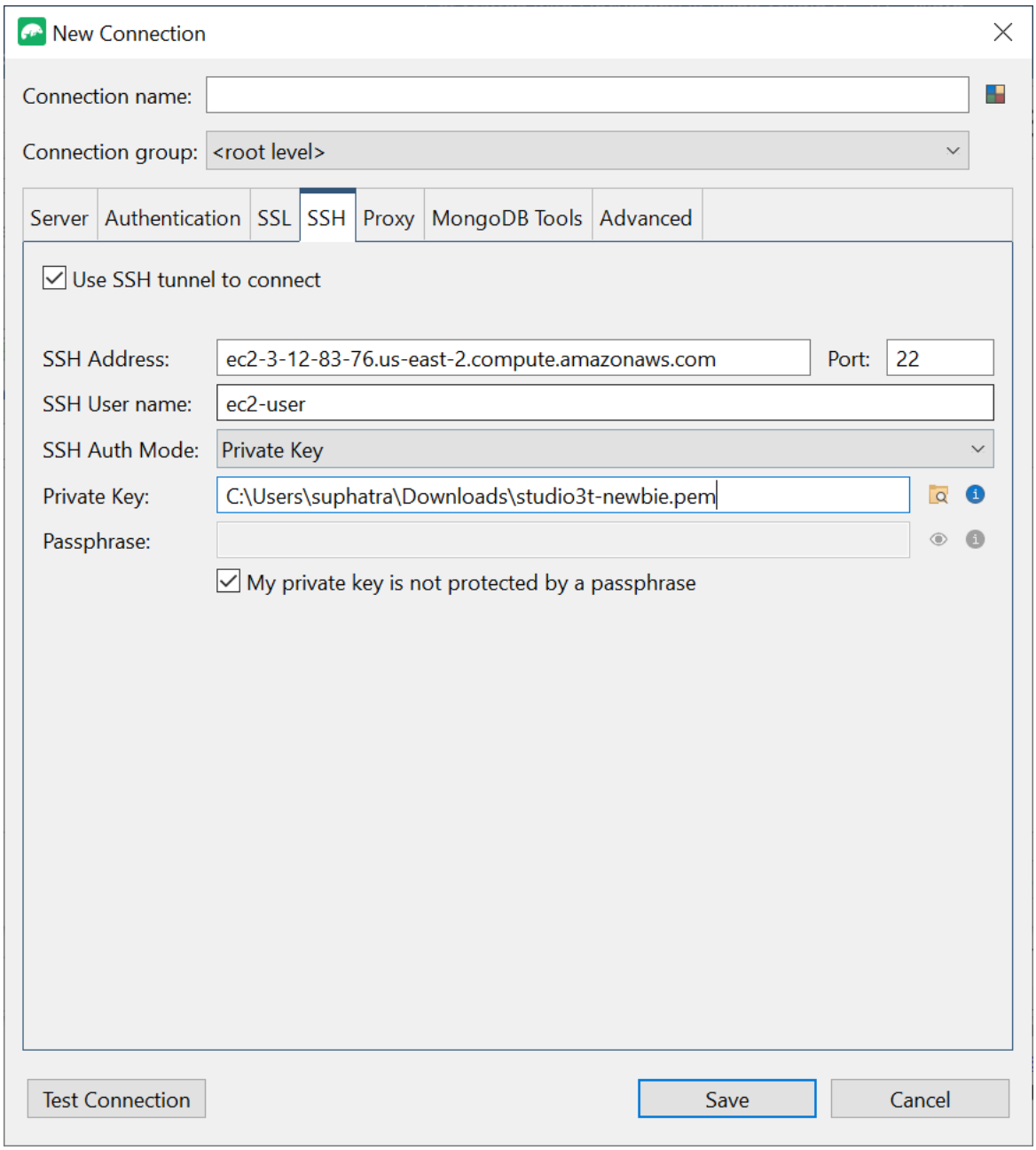
   a. SSHトンネルを使用する のチェックボックスをオンにして、SSHアドレス フィールドに SSH アドレスを入力します。これはインスタンスの公開 DNS (IPV4) です。この URL は Amazon EC2 マネジメントコンソール から取得できます。

   b. ユーザーネームを入力します。これは Amazon EC2 インスタンスのユーザーネームです。

   c. SSH 認証モードで、プライベートキー を選択します。プライベートキー フィールドで、ファイルファインダーアイコンを選択し、Amazon EC2 インスタンスのプライベートキーを選択します。これは、Amazon EC2 コンソールでインスタンスを作成するときに保存した.pem ファイル (キーペア) です。

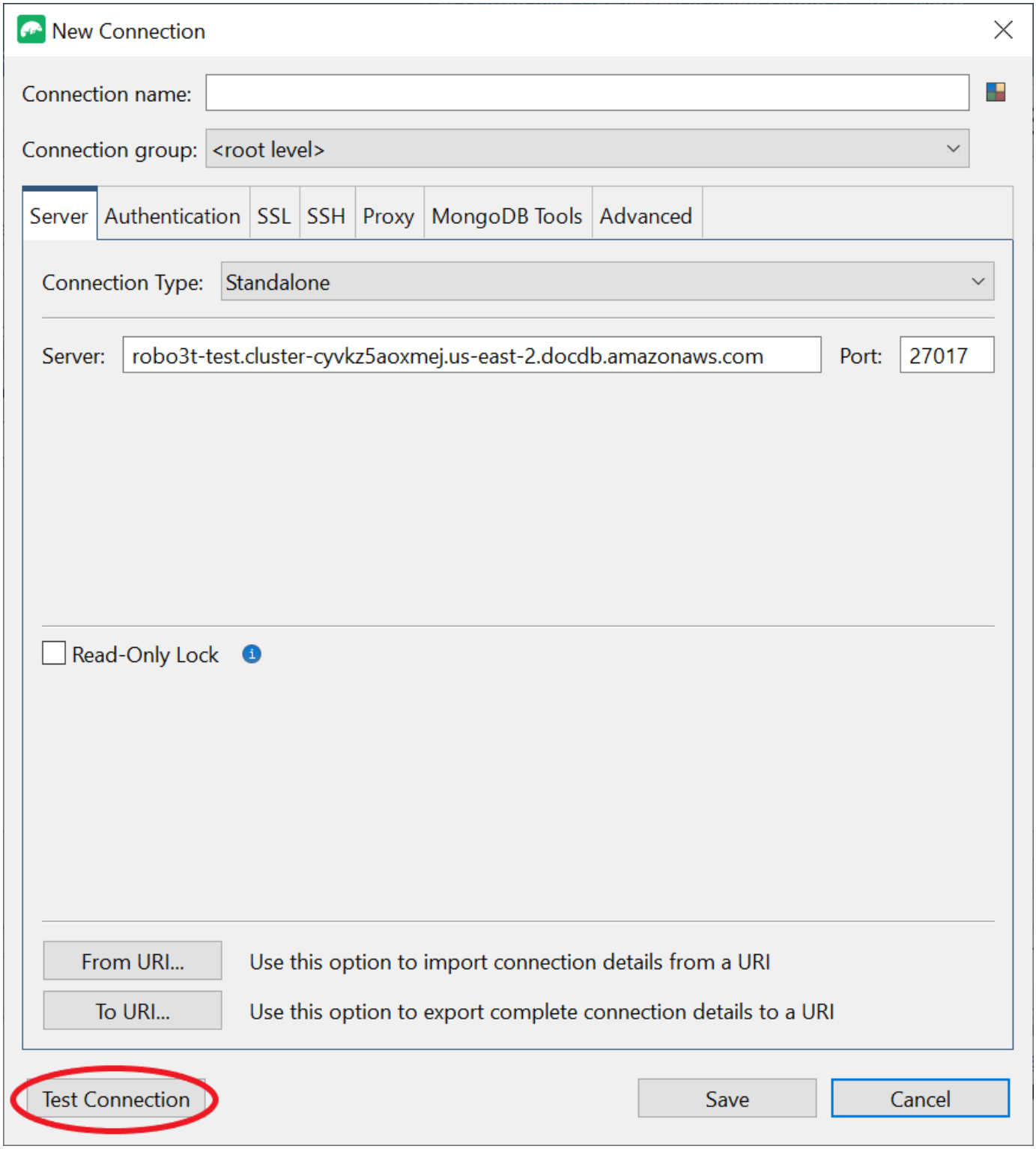
   d. Linux/macOS クライアントマシンを使用している場合は、次のコマンドを使用してプライベートキーの許可を変更する必要があることがあります。

      ```
      chmod 400 /fullPathToYourPemFile/<yourKey>.pem
      ```

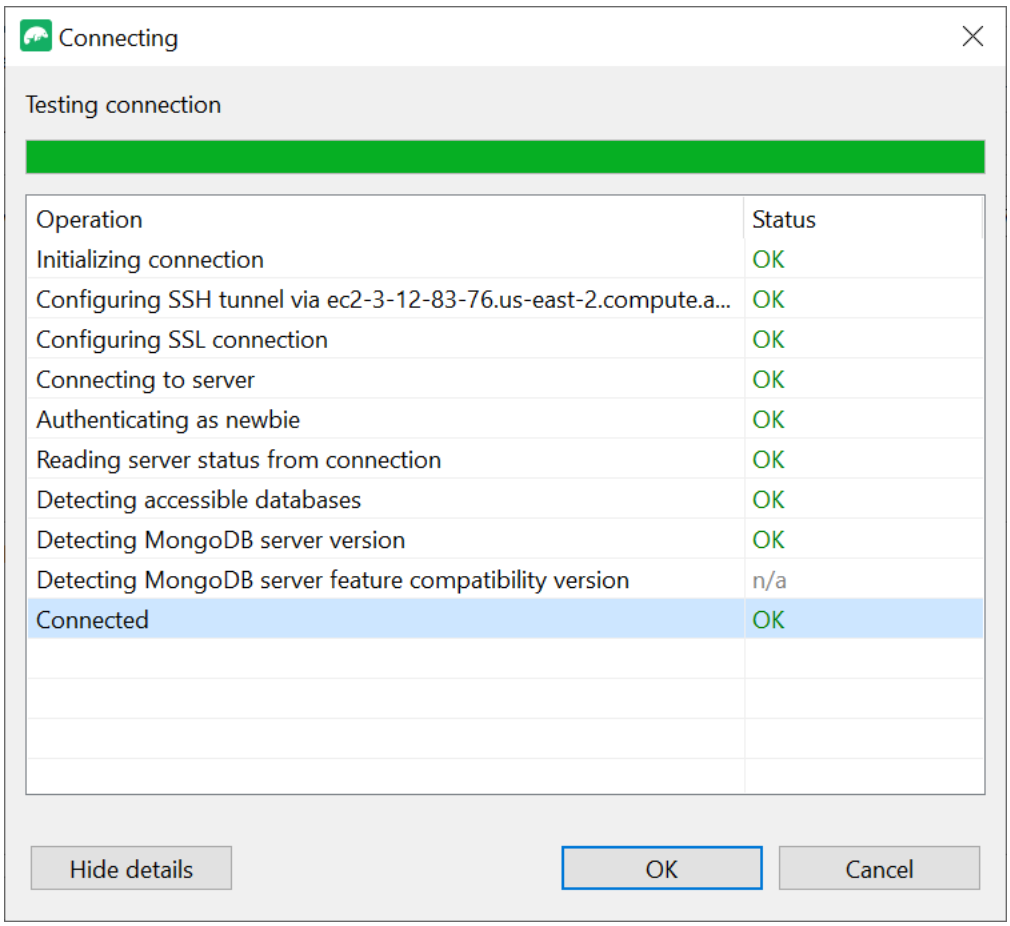


> **Note**
>
> この Amazon EC2 インスタンスは、DocumentDB クラスターと同じ Amazon VPC およびセキュリティグループ内に存在する必要があります。SSH アドレス、ユーザーネーム、プライベートキーは、Amazon EC2 マネジメントコンソール から取得できます。

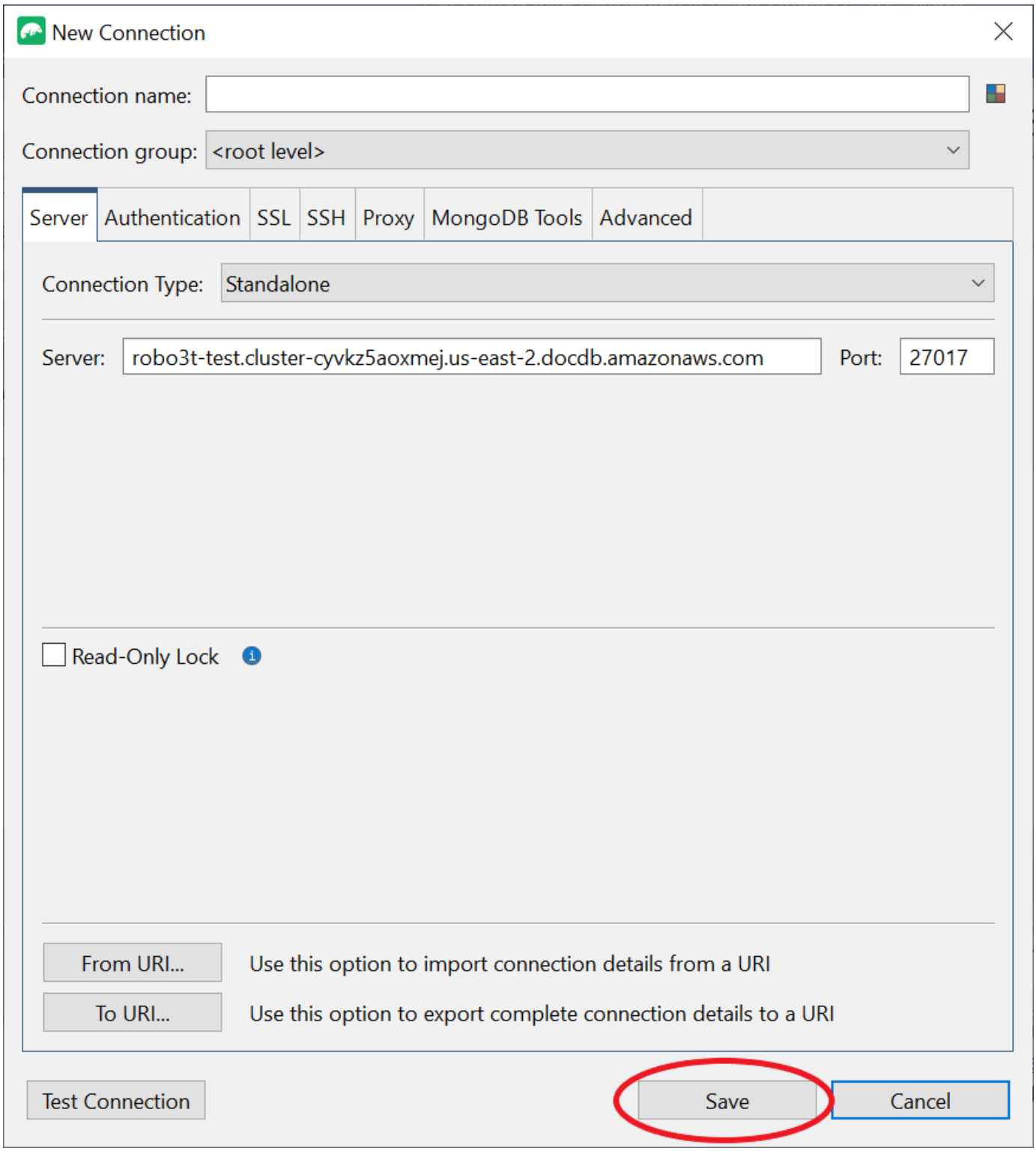
9. 次に 接続のテスト ボタンを選択して構成をテストします。



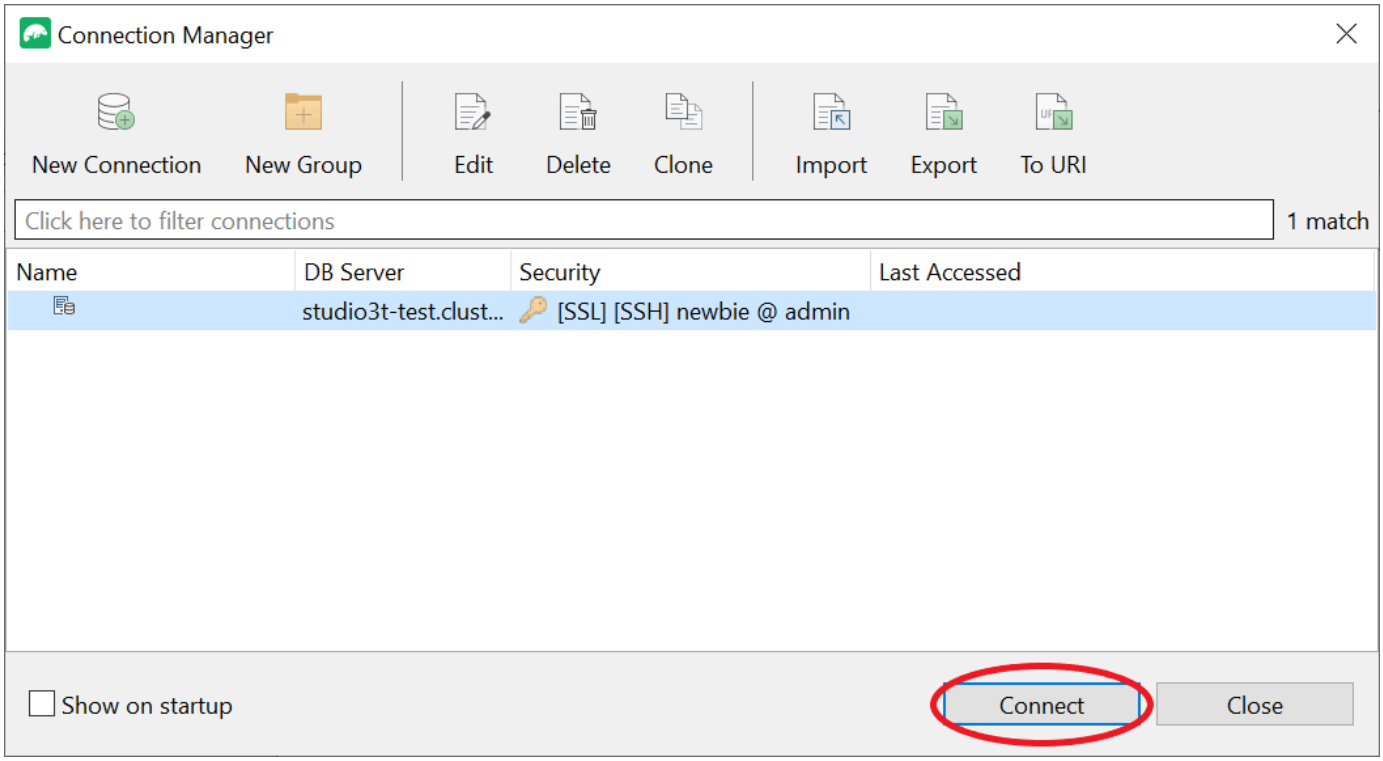
10. 診断ウィンドウには、テストが成功したことを示す緑色のバーがロードされます。次に OK を選択して、診断ウィンドウを閉じます。



11. 保存 を選択して、将来使用するために接続を保存します。



12. 次にクラスターを選択し、接続 を選択します。



お疲れ様でした。Studio 3T を介して Amazon DocumentDB クラスターに正常に接続されました。

DataGrip を使用して Amazon DocumentDB に接続する

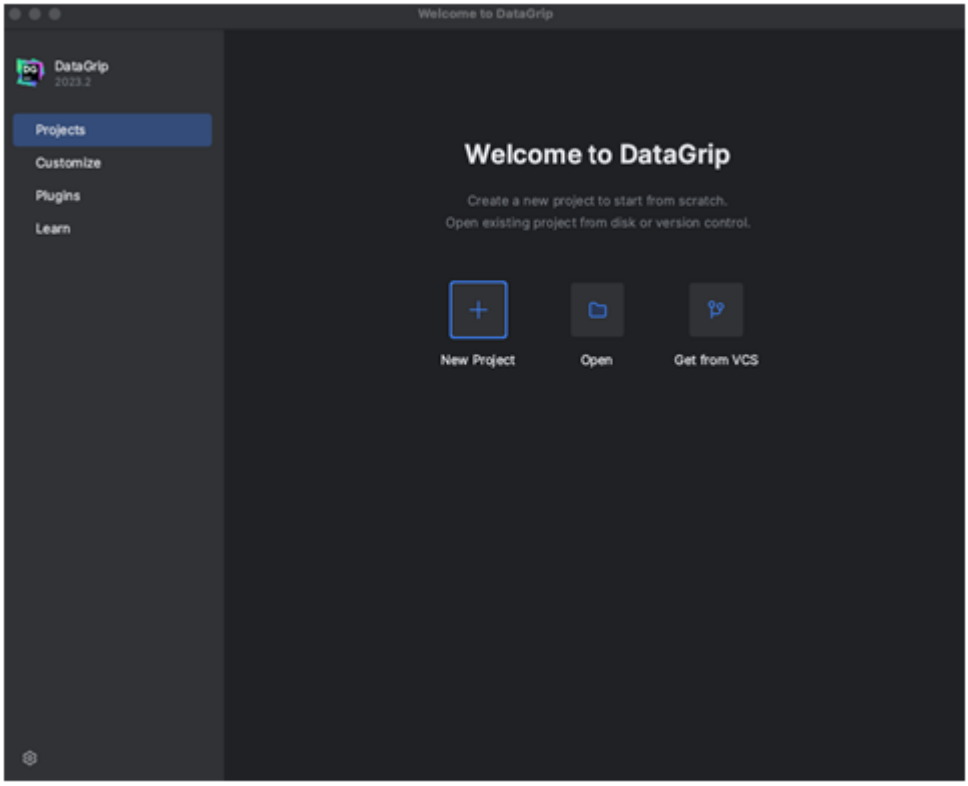
DataGrip は、Amazon DocumentDB を含むさまざまなデータベースシステムをサポートする強力な統合開発環境 (IDE) です。このセクションでは、DataGrip を使用して Amazon DocumentDB クラスターに接続する手順を説明します。これにより、グラフィカルインターフェイスを使用してデータを簡単に管理およびクエリできます。

前提条件

- DataGrip IDE がマシンにインストールされています。JetBrains からダウンロードできます。
- Amazon DocumentDB クラスターと同じ VPC で実行中の Amazon EC2 インスタンス。このインスタンスを使用して、ローカルマシンから Amazon DocumentDBCluster までの安全なトンネルを確立します。Amazon EC2 を使用した接続 方法に関する指示に従ってください。
- Amazon EC2 インスタンス、VPN 接続、または安全な VPN を使用して AWS インフラストラクチャに既にアクセスしている場合の代替方法。このオプションを使用する場合は、 を使用して Amazon DocumentDB に安全にアクセスする AWS Client VPN手順に従ってください。

DataGrip を使用して接続する

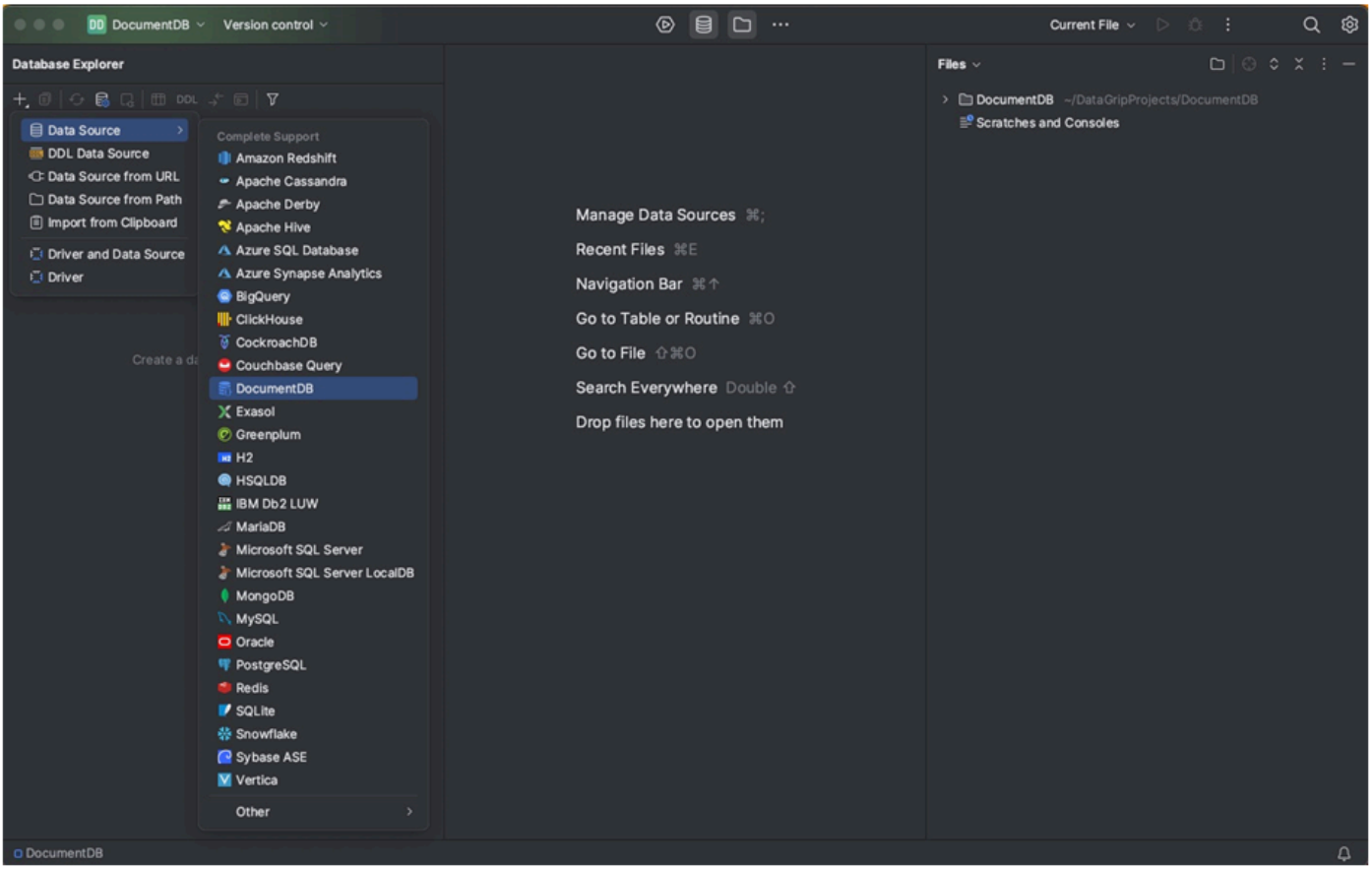
1. コンピューターで DataGrip を起動し、[新しいプロジェクト]を作成します。



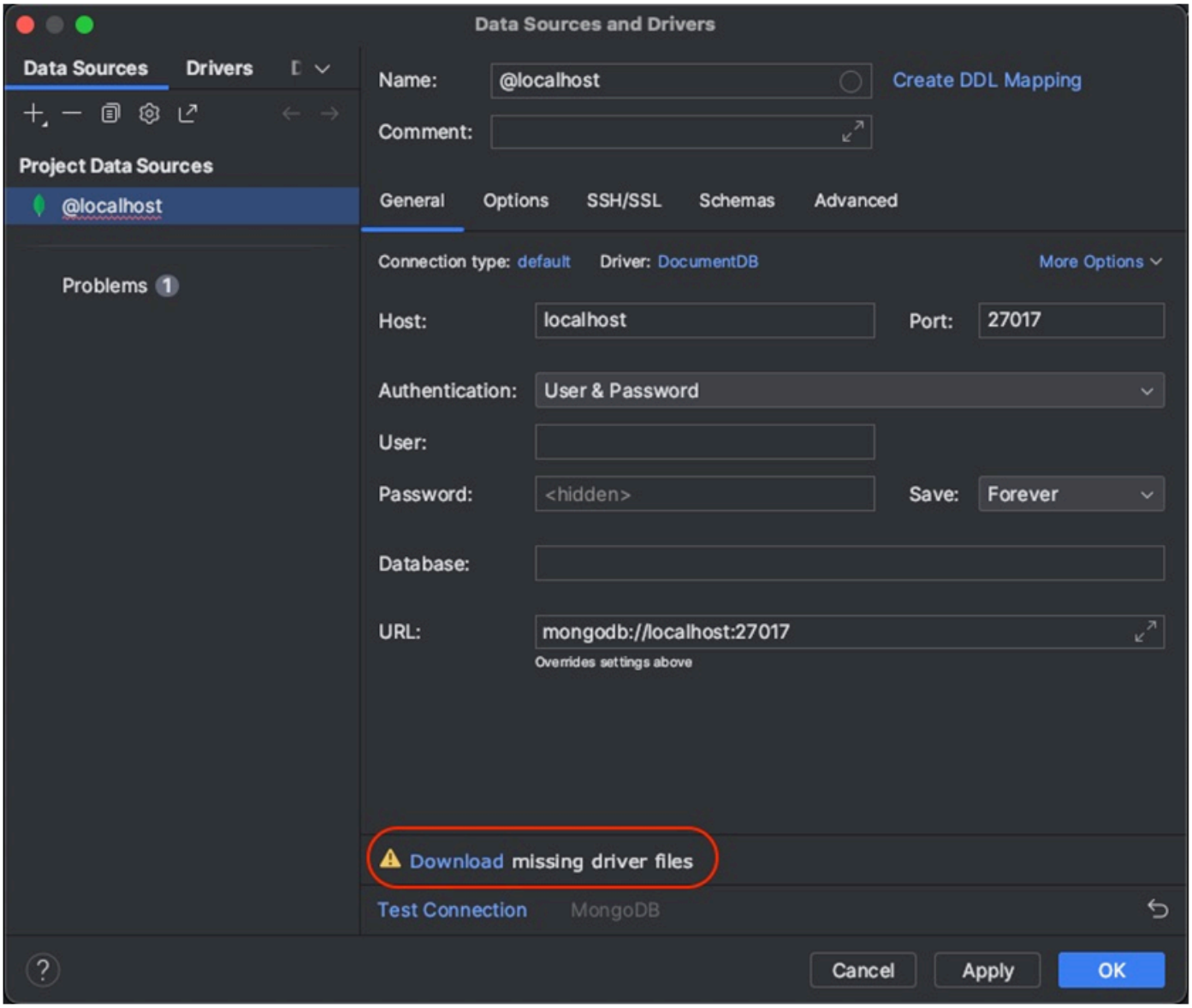
2. 以下のいずれかの方法を使用して、新しいデータソースを追加します。

   a. メインメニューから [ファイル]-[新規]-[データソース] に移動し、[DocumentDB] を選択します。

   b. [データベースエクスプローラー]で、ツールバーの新規アイコン (+) をクリックします。[データソース] に移動し、[DocumentDB] を選択します。



3. [全般] タブの [データソース] ページで、接続設定領域の下部に [見つからないドライバファイルのダウンロード] リンクがあるかどうかを確認します。このリンクをクリックして、データベースとの通信に必要なドライバーをダウンロードします。直接ダウンロードリンクについては、JetBrains JDBC ドライバーを参照してください。



4. [一般] タブで、接続の詳細を指定します。

   a. [ホスト] フィールドで、Amazon DocumentDB クラスターエンドポイントを指定します。

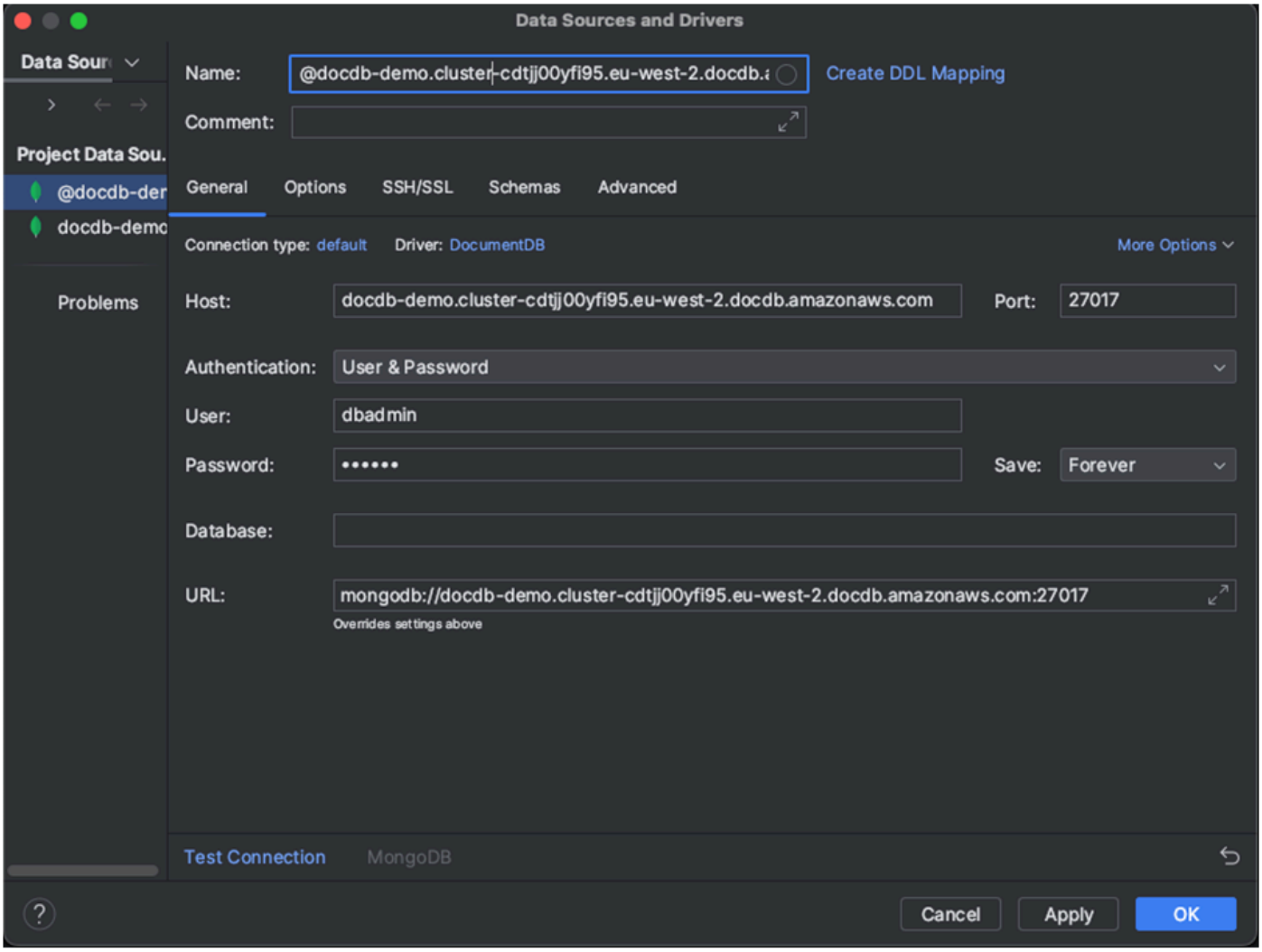
   b. [ポート] は既に 27017 に設定されています。クラスターが別のポートにデプロイされている場合は変更してください。

   c. [認証] で、[ユーザーとパスワード] を選択します。

   d. 自分のユーザー名とパスワードを入力します。

   e. [データベース] フィールドはオプションです。接続先のデータベースを指定できます。

   f. 上記の詳細を追加すると、[URL] フィールドが自動補完されます。



5. [SSH/SSL] タブで [SSH トンネルを使用する] を有効にし、アイコンをクリックして [SSH 設定] ダイアログを開きます。次の情報を入力します。

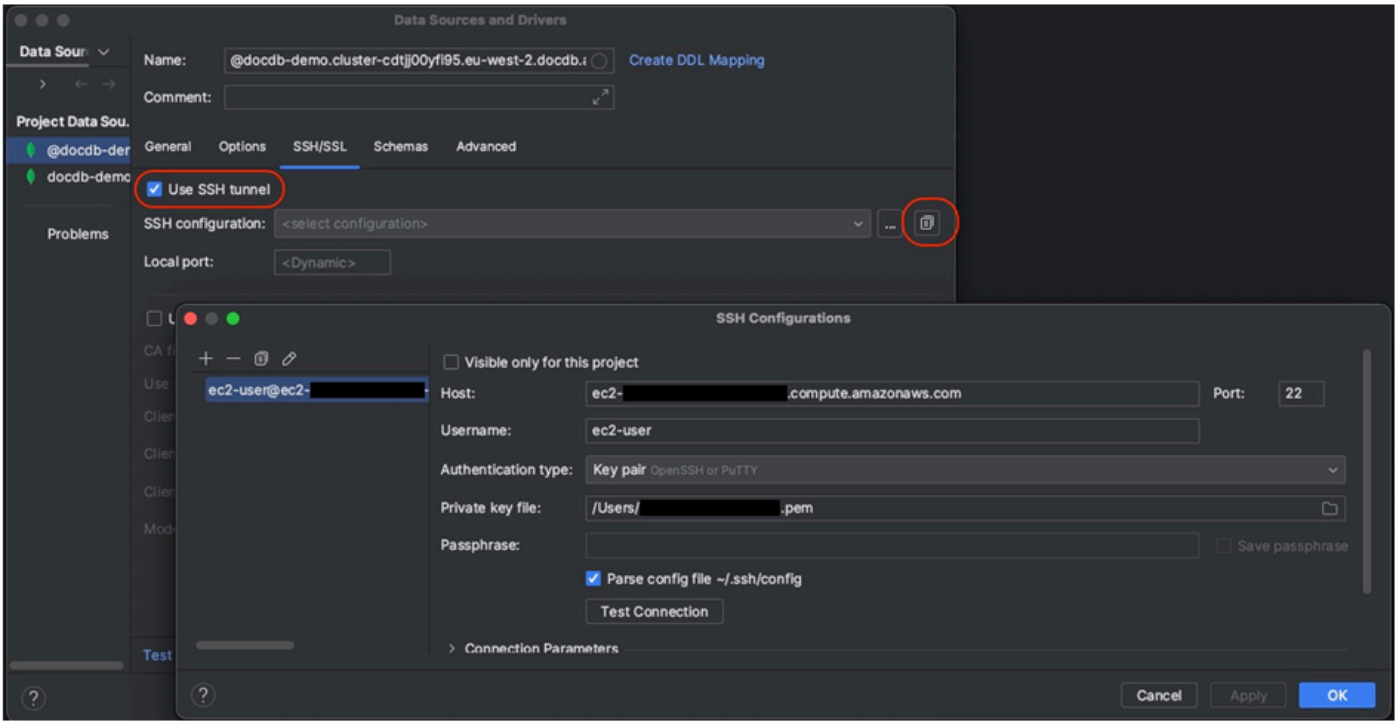
   a. [ホスト] フィールドに、Amazon EC2 インスタンスのホスト名を入力します。

   b. Amazon EC2 インスタンスのユーザーネームとパスワードを入力します。

   c. [認証タイプ] で、[キーペア] を選択します。

   d. [プライベートキーファイル] を入力します。

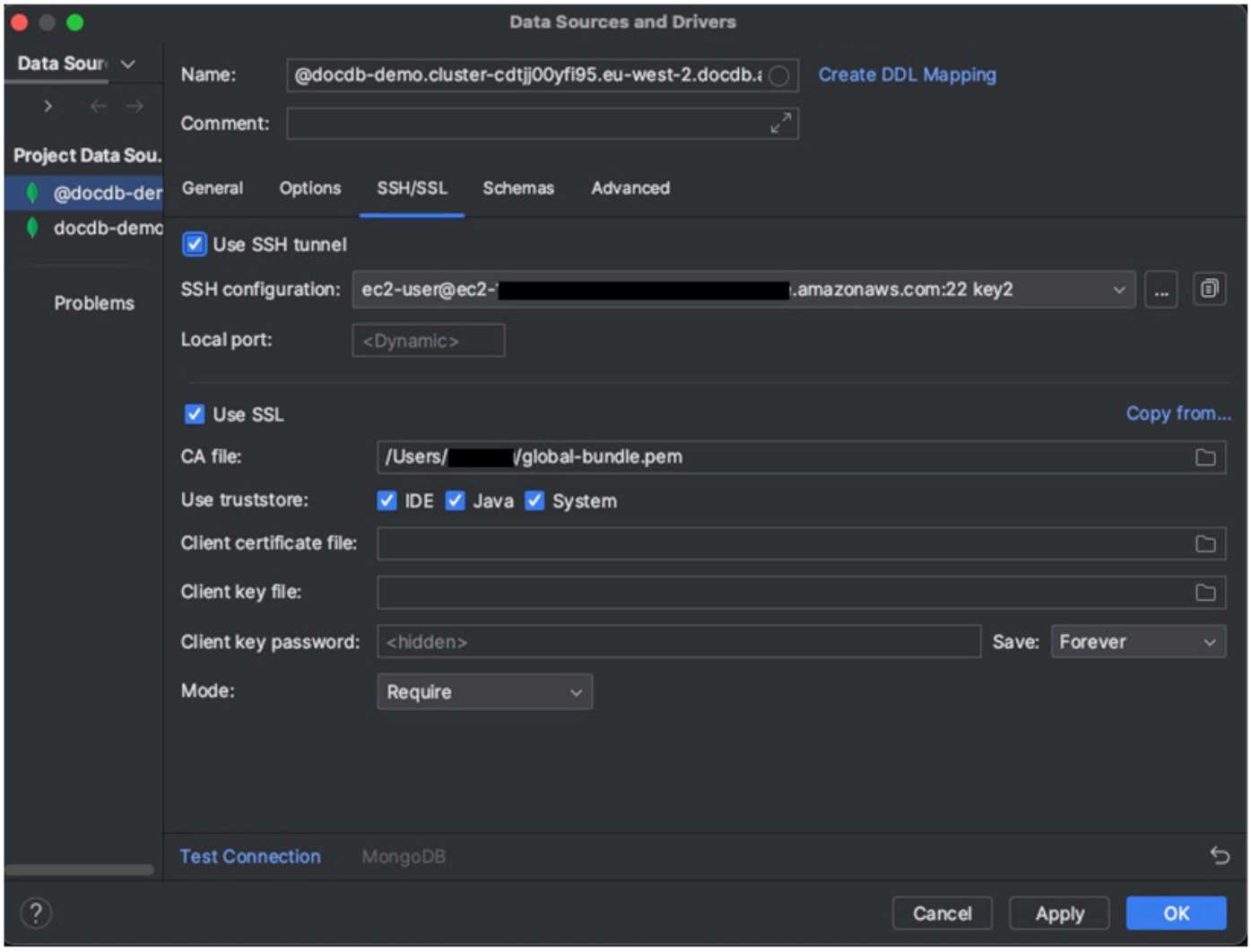
> **Note**
>
> VPN オプションを使用している場合は、SSH トンネルを設定する必要はありません。



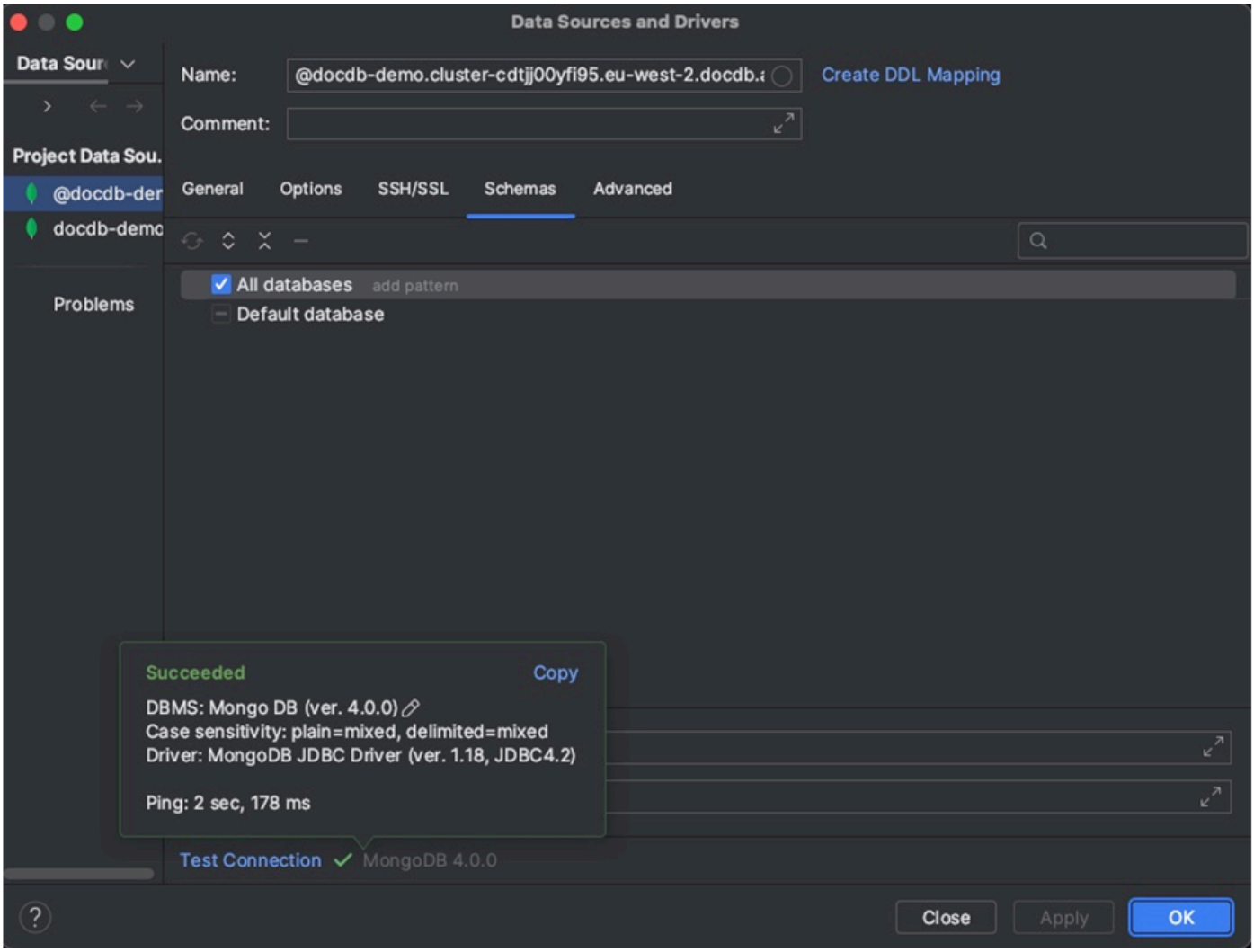
6. [SSH/SSL] タブで、[SSL を使用する] を有効にします。[CA ファイル] フィールドに、`global-bundle.pem` コンピューター上のファイルの場所を入力します。[モード] は [必須] オプションのままにしておきます。

> **Note**
>
> 証明書は次の場所からダウンロードするか、wgethttps://truststore.pki.rds.amazonaws.com/global/global-bundle.pem コマンドを使用してダウンロードできます。

> **Note**
>
> Amazon DocumentDB エラスティッククラスターに接続する場合、CA ファイルを指定する必要はありません。[SSLを使用する] オプションはオンのままにして、他のすべてのオプションはデフォルト値のままにします。



7. [スキーマ] タブの [すべてのデータベース] を選択するか、[スキーマパターン] フィールドに「*: *」というフィルターを入力します。[接続テスト] リンクをクリックして接続をテストします。



8. 接続が正常にテストされたら、[OK] をクリックしてデータソース設定を保存します。

DataGrip の機能

DataGrip には Amazon DocumentDB を使用して効率的に作業するためのさまざまな機能が用意されています。

- SQL エディター — DataGrip の SQL エディターを使用してDocumentDB コレクションに SQL のようなクエリを記述して実行します。
- ビジュアルクエリビルダー — ビジュアルクエリビルダーを使用すると、SQL コードを記述せずにグラフィカルにクエリを作成できます。
- スキーマ・マネジメント — コレクションの作成、変更、削除など、データベーススキーマを簡単に管理できます。

- データ・ビジュアライゼーション — DataGrip のさまざまな視覚化ツールを使用して、データを表示および分析します。
- データのエクスポートとインポート — DataGrip のエクスポートおよびインポート機能を使用して、Amazon DocumentDB と他のデータベース間でデータを転送します。

Amazon DocumentDB やその他のデータベースシステムの操作に関するより高度な機能やヒントについては、公式 DataGrip ドキュメントを参照してください。

Amazon EC2 を使用した接続

このセクションでは、Amazon DocumentDB クラスターと Amazon EC2 間の接続を設定し、Amazon EC2 インスタンスから Amazon DocumentDB クラスターにアクセスする方法について説明します。

EC2 接続を設定する方法は複数あります。

- EC2 インスタンスを Amazon DocumentDB データベースに自動的に接続する — EC2 コンソールの自動接続機能を使用して、EC2 インスタンスと新規または既存の Amazon DocumentDB データベース間の接続を自動で設定できます。この接続により、EC2 インスタンスと Amazon DocumentDB データベース間でやり取りされるトラフィックを確立できます。このオプションは通常、新しいセキュリティグループのテストと作成に使用されます。
- EC2 インスタンスを Amazon DocumentDB データベースに手動で接続する — 自動接続機能によって作成されるのと同じ設定を再現することにより、セキュリティグループを手動で設定して割り当て、EC2 インスタンスと Amazon DocumentDB データベース間の接続を設定できます。このオプションは、通常、高度な設定を変更したり、既存のセキュリティグループを使用するために使用されます。

前提条件

いずれのオプションを使用する場合でも、最初の Amazon DocumentDB クラスターを作成する前に、以下の操作を行う必要があります。

Amazon Web Services (AWS) アカウントを作成する

Amazon DocumentDB を使用する前に、Amazon Web Services (AWS) アカウントを持っている必要があります。 AWS アカウントは無料です。使用しているサービスとリソースに対してのみ料金をお支払いいただきます。

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

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

   サインアップ手順の一環として、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。

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

必要に応じて、必要な AWS Identity and Access Management (IAM) アクセス許可を設定します。

クラスター、インスタンス、クラスターパラメータグループなどの Amazon DocumentDB リソースを管理するには、 がリクエストの認証 AWS に使用できる認証情報が必要です。詳細については、「Amazon DocumentDB の Identity and Access Management」を参照してください。

1. の検索バーで AWS Management Console、IAM と入力し、表示されるドロップダウンメニューで IAM を選択します。
2. IAM コンソールにアクセスしたら、ナビゲーションペインから [ユーザー] を選択します。
3. ユーザーネームを選択します。
4. [アクセス許可の追加] ボタンをクリックします。
5. [Attach existing policies directly (既存のポリシーを直接アタッチ)] を選択します。
6. 検索バーに `AmazonDocDBFullAccess` と入力し、検索結果に表示されたら、それを選択します。
7. [Next: レビュー] と表示されている青いボタンをクリックします。
8. [アクセス許可の追加] となっている下の青いボタンをクリックします。

Amazon VPC (Amazon Virtual Private Cloud) を作成する

どの AWS リージョン にあるかによって、デフォルトの VPC が既に作成されているかどうかがわかります。デフォルトの VPC がない場合は、Amazon VPC のユーザーガイド の Amazon VPC の開始方法 のステップ 1 を完了させます。これには 5 分もかかりません。

Amazon EC2 への自動接続

トピック

- EC2 インスタンスを新しい Amazon DocumentDB データベースに自動接続する
- EC2 インスタンスを既存の Amazon DocumentDB データベースに自動的に接続する
- EC2 インスタンスとの自動接続の概要
- 接続中のコンピューティングリソースを表示する

EC2 インスタンスと新しい Amazon DocumentDB データベース間の接続を設定する前に、「EC2 インスタンスとの自動接続の概要」に記載される要件を満たしているか確認してください。接続の設定後にこれらのセキュリティグループに変更を加えると、EC2 インスタンスと Amazon DocumentDB データベース間の接続に影響を及ぼす可能性があります。

> **Note**
>
> AWS Management Consoleを使用することでのみ、EC2 インスタンスと Amazon DocumentDB データベース間の接続を自動的に設定できます。 AWS CLI または Amazon DocumentDB API を使用して接続を自動的にセットアップすることはできません。

EC2 インスタンスを新しい Amazon DocumentDB データベースに自動接続する

以下のプロセスは、前提条件 トピック記載のステップが完了済みであることを前提としています。

ステップ

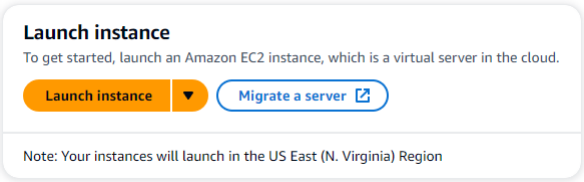
- ステップ 1: Amazon EC2 インスタンスを作成する
- ステップ 2: Amazon DocumentDB クラスターを作成する
- ステップ 3: Amazon EC2 インスタンスに接続する
- ステップ 4: MongoDB シェルをインストールする
- ステップ 5: Amazon DocumentDB TLS の管理

- ステップ 6: Amazon DocumentDB クラスターに接続する
- ステップ 7: データの挿入とクエリ
- ステップ 8: 使ってみる

ステップ 1: Amazon EC2 インスタンスを作成する

このステップでは、後で Amazon DocumentDB クラスターのプロビジョニングに使用するのと同じリージョンと Amazon VPC に Amazon EC2 インスタンスを作成します。

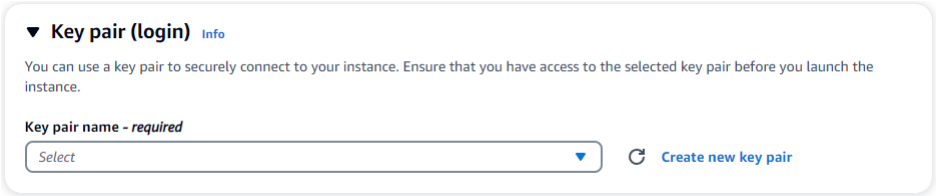
1. Amazon EC2 コンソールで、[インスタンスを起動] を選択します。



2. [名前とタグ] セクション内の [名前] フィールドに、名前または識別子を入力します。
3. [Amazon マシンイメージ (AMI)] ドロップダウンリストで、[Amazon Linux 2 AMI] を選択します。



4. [インスタンスタイプ] のドロップダウンリストで、[t3.micro] を選択します。
5. [キーペア (ログイン)] セクションで、既存のキーペアの識別子を入力するか、もしくは [新しいキーペアの作成] を選択します。



   Amazon EC2 キーペアを指定する必要があります。

   Amazon EC2 のキーペアがある場合 :

   a. キーペアを選択し、リストからご自身のキーペアを選択します。
   b. Amazon EC2 インスタンスにログインするには、あからじめプライベートキーファイル (.pem もしくは.ppk ファイル) を利用可能にしておく必要があります。

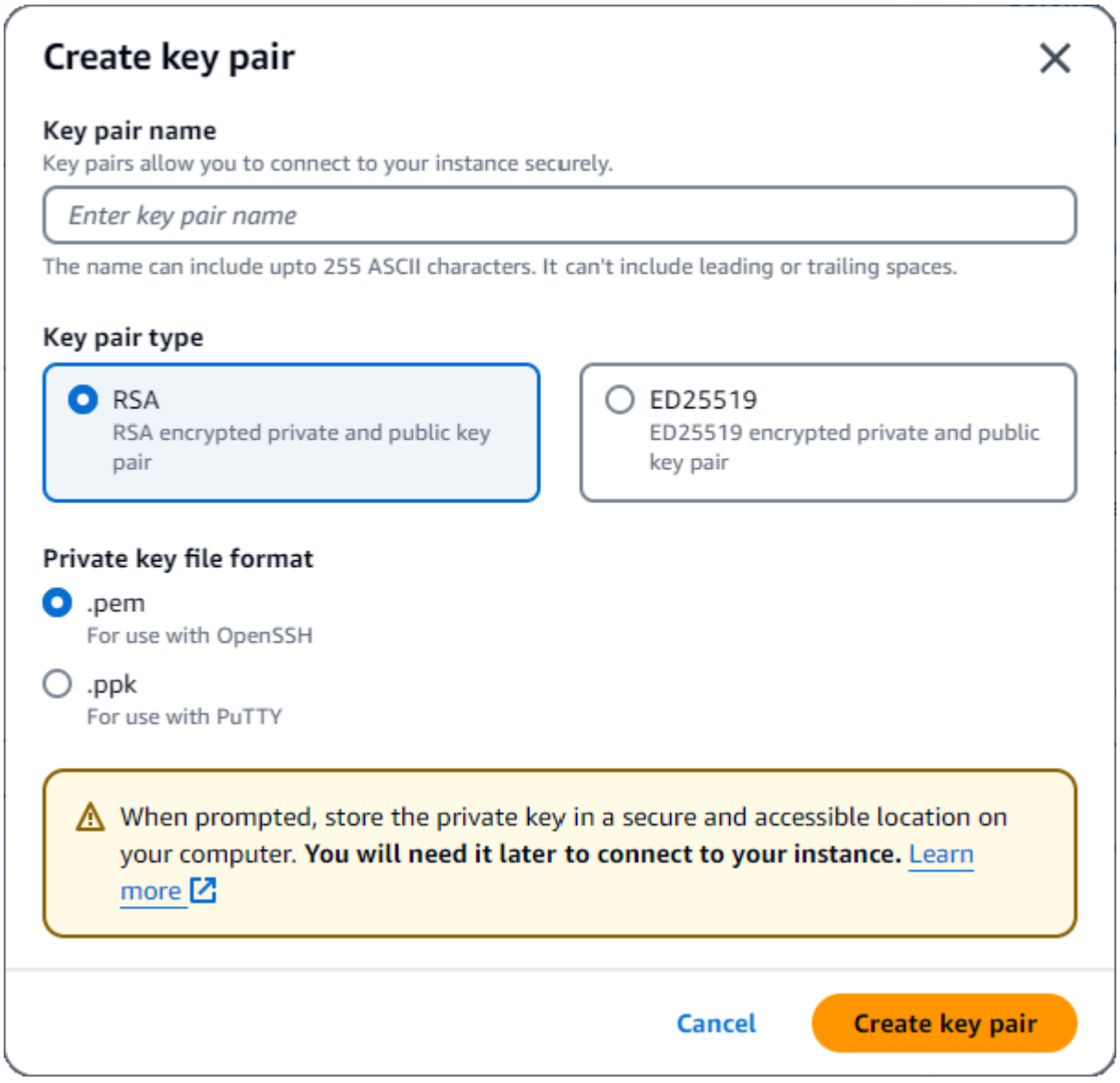
Amazon EC2 のキーペアがない場合:

a. [新しいキーペアの作成] を選択すると、[キーペアの作成] ダイアログボックスが表示されます。

b. [キーペア名] フィールドに名前を入力します。

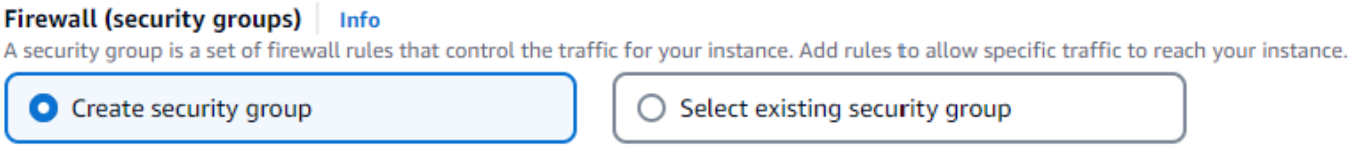
c. [キーペアタイプ] と [プライベートキーファイル形式] を選択します。

d. [キーペアの作成] を選択してください。



> **Note**
>
> セキュリティ上の理由から、EC2 インスタンスへの SSH とインターネット接続の両方にキーペアを使用することを強くお勧めします。

6. オプション： ネットワーク設定セクションのファイアウォール (セキュリティグループ) で、セキュリティグループの作成を選択します。



セキュリティグループの作成 (EC2 接続に適用されるすべてのトラフィック許可ルールを確認してください) を選択します。

> **Note**
>
> 既存のセキュリティグループを使用する場合は、「」の手順に従いますAmazon EC2 を手動で接続する。

7. 概要セクションで、EC2 設定を確認し、正しい場合はインスタンスを起動を選択します。

ステップ 2: Amazon DocumentDB クラスターを作成する

Amazon EC2 インスタンスのプロビジョニング中に、Amazon DocumentDB クラスターを作成します。

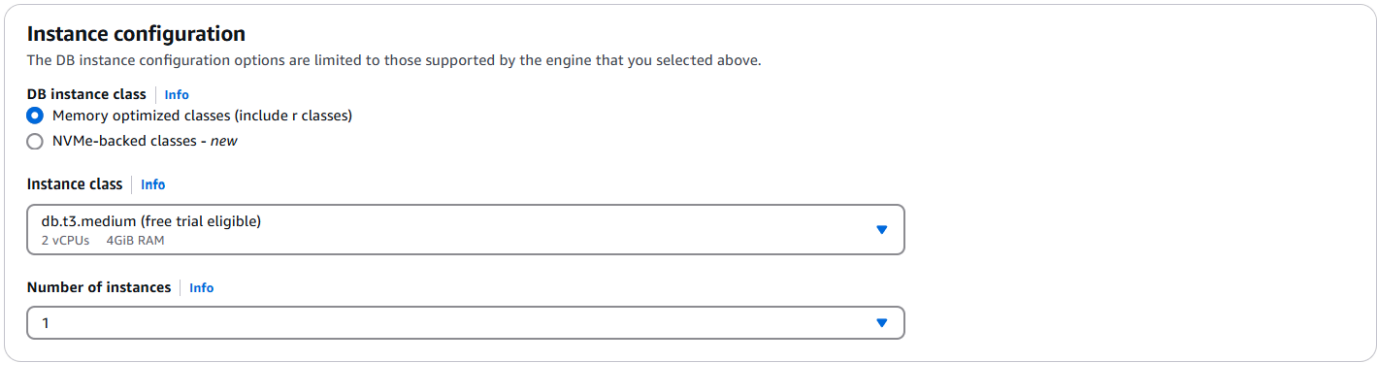
1. Amazon DocumentDB コンソールに移動し、ナビゲーションペインから クラスター を選択します。
2. [作成] を選択します。
3. [クラスタータイプ] 設定は、デフォルトの [インスタンスベースのクラスター] のままにしておきます。
4. クラスター設定で、クラスター識別子に一意の名前を入力します。コンソール操作では、入力方法に関係なくクラスター名のすべての文字が小文字に変換されることにご留意ください。

   エンジンバージョンはデフォルト値の 5.0.0 のままにします。

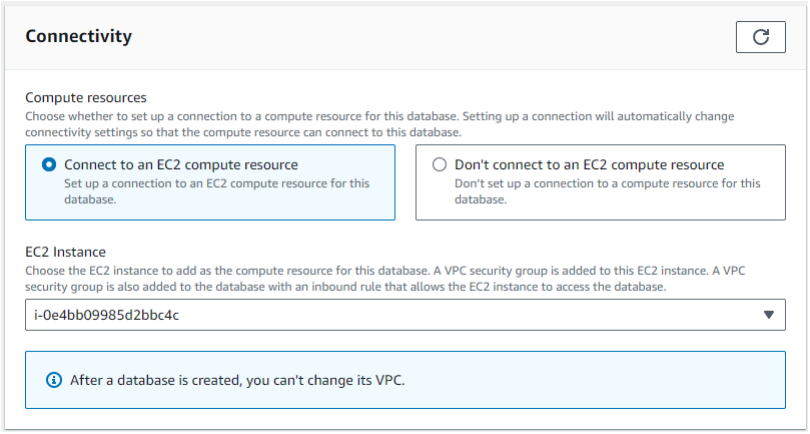
5. クラスターストレージ設定の場合、Amazon DocumentDB Standard のデフォルト設定のままにします。
6. インスタンス設定：

   - DB インスタンスクラスの場合は、メモリ最適化クラス (r クラスを含む) を選択します (デフォルト)。

     もう 1 つのインスタンスオプションは NVMe-backed クラスです。詳細については[NVMe-backed インスタンス]を参照してください。
   - インスタンスクラスで、ニーズに合ったインスタンスタイプを選択します。インスタンスクラスの詳細については、「」を参照してください[インスタンスクラスの仕様]。
   - インスタンスの数については、ニーズに最も適した数を選択します。数値が低いほど、コストが低くなり、クラスターで管理できる読み取り/書き込みボリュームが低くなることに注意してください。



7. [接続] で、[EC2 コンピューティングリソースに接続する] を選択します。これは、ステップ 1 で作成した EC2 インスタンスです。



> **Note**
>
> EC2 コンピューティングリソースに接続すると、Amazon DocumentDB クラスターへの EC2 コンピューティングリソース接続向けセキュリティグループが自動的に作成されます。クラスターの作成完了後、新しく作成したセキュリティグループを確認したい場合は、クラスターリストに移動して該当するクラスター識別子を選択します。[接続とセキュリティ] タブ内で [セキュリティグループ] に移動して、[セキュリティグループ名 (ID)] で見たいグループを見つけます。クラスターの名前が含まれ、 のようになりますdocdb-ec2-docdb-2023-12-11-21-33-41:i-0e4bb09985d2bbc4c (sg-0238e0b0bf0f73877)。

8. 認証セクションで、プライマリユーザーのユーザー名を入力し、セルフマネージドを選択します。パスワードを入力して確認します。

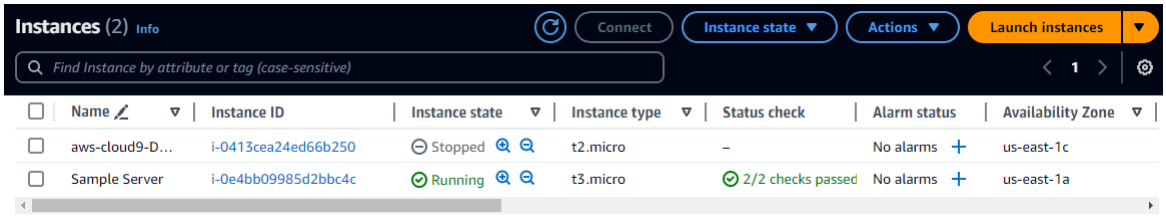
   代わりに Managed in AWS Secrets Manager を選択した場合は、詳細については[Amazon DocumentDB と によるパスワード管理 AWS Secrets Manager]「」を参照してください。

9. [クラスターを作成] を選択します。

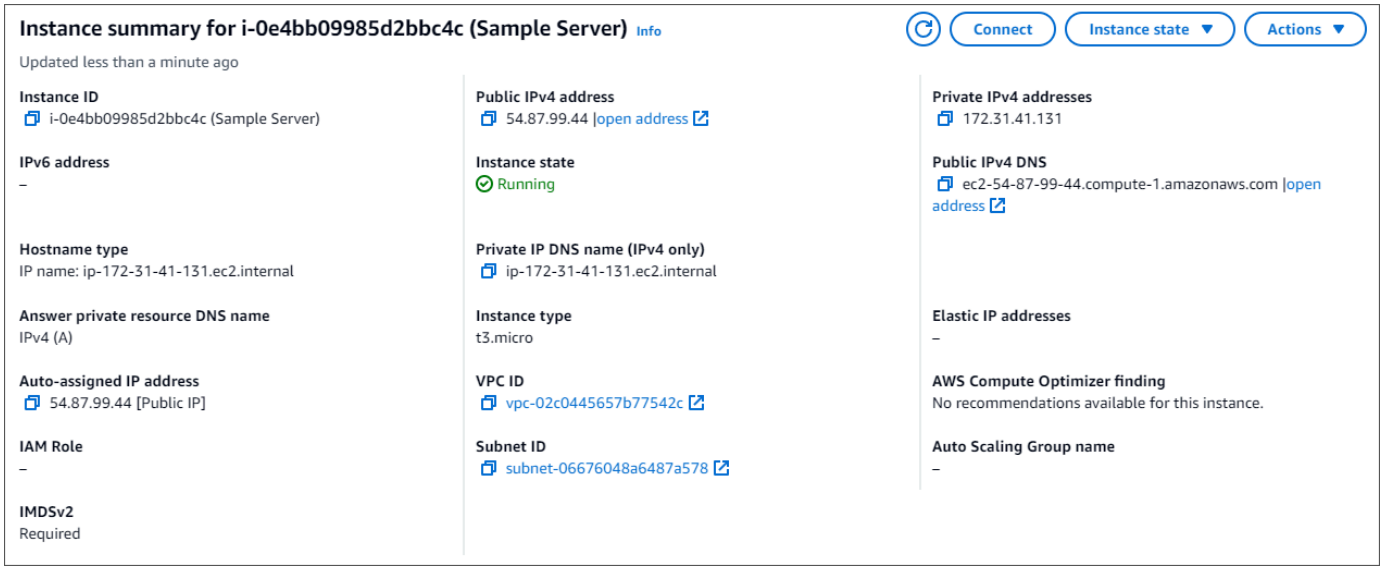
ステップ 3: Amazon EC2 インスタンスに接続する

Mongo Shell をインストールするには、まず Amazon EC2 インスタンスに接続する必要があります。Mongo シェルをインストールすると、Amazon DocumentDB クラスターに接続してクエリを実行できます。以下のステップを実行します。

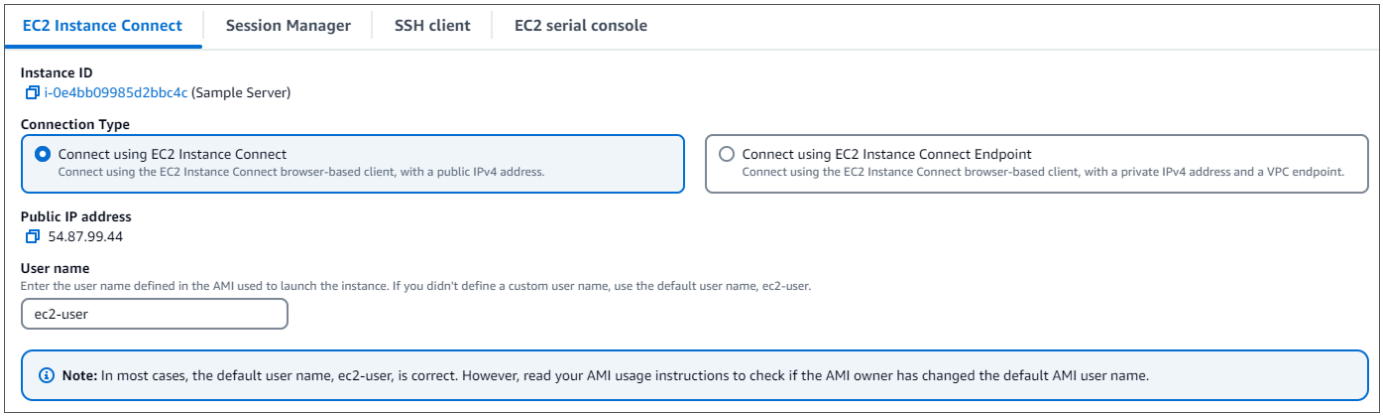
1. Amazon EC2 コンソールで、インスタンスに移動し、作成したインスタンスが実行中かどうかを確認します。実行されている場合は、インスタンス ID をクリックしてインスタンスを選択します。



2. [接続] を選択します。



3. 接続方法として、Amazon EC2 インスタンス Connect、セッションマネージャー、SSH クライアント、EC2 シリアルコンソールの 4 つのオプションがタブ形式で表示されます。いずれかを選択し、その指示に従わなければなりません。完了したら、[接続する] を選択します。



> **Note**
>
> このウォークスルーの開始後に IP アドレスが変更された場合、または後で環境に戻る場合は、demoEC2セキュリティグループのインバウンドルールを更新して、新しい API アドレスからのインバウンドトラフィックを有効にする必要があります。

ステップ 4: MongoDB シェルをインストールする

Amazon DocumentDB クラスターの接続とクエリに使用するコマンドラインユーティリティである MongoDB シェルをインストールできるようになりました。 Amazon DocumentDB 現在、MongoDB

シェルには最新バージョンの mongosh と以前のバージョンの mongo シェルの 2 つのバージョンがあります。

> **Important**
>
> バージョン 6.13.1 より前の Node.js ドライバーには既知の制限がありますが、現在 Amazon DocumentDB の IAM ID 認証ではサポートされていません。Node.js ドライバーを使用する Node.js ドライバーとツール (mongosh など) は、Node.js ドライバーバージョン 6.13.1 以降を使用するようにアップグレードする必要があります。

以下の手順に従って、オペレーティングシステムの MongoDB シェルをインストールします。

On Amazon Linux

Amazon Linux に MongoDB シェルをインストールするには

IAM 認証を使用しておらず、最新の MongoDB シェル (mongosh) を使用して Amazon DocumentDB クラスターに接続する場合は、次の手順に従います。

1. リポジトリファイルを作成します。作成した EC2 インスタンスのコマンドラインで、次のコマンドを実行します。

```
echo -e "[mongodb-org-5.0] \nname=MongoDB Repository\nbaseurl=https://
repo.mongodb.org/yum/amazon/2023/mongodb-org/5.0/x86_64/\ngpgcheck=1 \nenabled=1
 \ngpgkey=https://pgp.mongodb.com/server-5.0.asc" | sudo tee /etc/yum.repos.d/
mongodb-org-5.0.repo
```

2. 完了したら、コマンドプロンプトで次の 2 つのコマンドオプションのいずれかを使用して mongosh をインストールします。

   オプション 1 — Amazon EC2 設定中にデフォルトの Amazon Linux 2023 を選択した場合は、次のコマンドを入力します。

```
sudo yum install -y mongodb-mongosh-shared-openssl3
```

   オプション 2 — Amazon EC2 設定中に Amazon Linux 2 を選択した場合は、次のコマンドを入力します。

```
sudo yum install -y mongodb-mongosh
```

IAM 認証を使用している場合は、以前のバージョンの MongoDB シェル (5.0) を使用して Amazon DocumentDB クラスターに接続する必要があります。次の手順を実行します。

1. リポジトリファイルを作成します。作成した EC2 インスタンスのコマンドラインで、次のコマンドを実行します。

```
echo -e "[mongodb-org-5.0] \nname=MongoDB Repository\nbaseurl=https://
repo.mongodb.org/yum/amazon/2023/mongodb-org/5.0/x86_64/\ngpgcheck=1 \nenabled=1
 \ngpgkey=https://pgp.mongodb.com/server-5.0.asc" | sudo tee /etc/yum.repos.d/
mongodb-org-5.0.repo
```

2. 完了したら、コマンドプロンプトで次のコマンドオプションを使用して mongodb 5.0 シェルをインストールします。

```
sudo yum install -y mongodb-org-shell
```

On Ubuntu

Ubuntu に mongosh をインストールするには

1. パッケージ管理システムで使用されるパブリックキーをインポートします。

```
curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
share/keyrings/mongodb-server-5.0.gpg
```

2. Ubuntu のバージョンに適切なコマンドを使用して、MongoDB の リストファイル `mongodb-org-5.0.list` を作成します。

```
echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-
server-5.0.gpg ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/5.0
 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list
```

3. 次のコマンドを使用して、ローカルパッケージデータベースをインポートおよび更新します。

```
sudo apt-get update
```

4. mongosh をインストールします。

```
sudo apt-get install -y mongodb-mongosh
```

以前のバージョンの MongoDB の Ubuntu システムへのインストールについては、「 Install MongoDB Community Edition on Ubuntu 」を参照してください。

On other operating systems

mongo シェルを他のオペレーティングシステムにインストールするには、MongoDB ドキュメントの「 MongoDB Community Edition のインストール 」を参照してください。

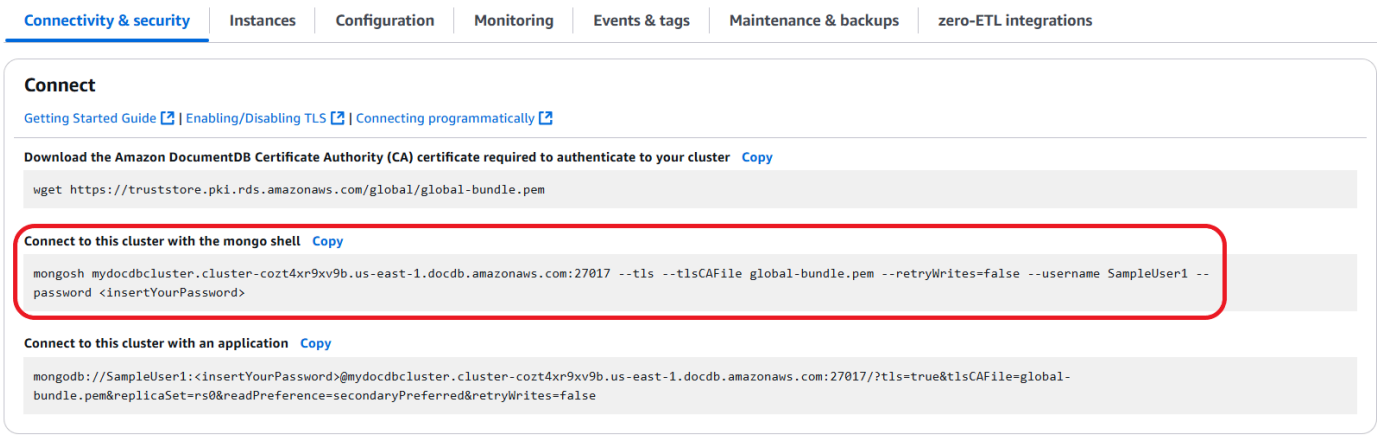
ステップ 5: Amazon DocumentDB TLS の管理

次のコード `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem` を使用して Amazon DocumentDB の CA 証明書をダウンロードします。

> **Note**
>
> 全ての新しい Amazon DocumentDB クラスターに対して、デフォルトで Transport Layer Security (TLS) が有効になっています。詳細については、「Amazon DocumentDB クラスター TLS 設定の管理」を参照してください。

ステップ 6: Amazon DocumentDB クラスターに接続する

1. Amazon DocumentDB コンソールの [クラスター] で、接続したいクラスターを見つけます。そのクラスターのクラスター識別子をクリックして、作成したクラスターを選択します。
2. [接続とセキュリティ] タブの [接続する] ボックス内で、[mongo シェルを使用してこのクラスターに接続する] を見つけます。



表示された接続文字列をコピーし、ターミナルに貼り付けます。

これに以下の変更を加えます。

a. 文字列に正しいユーザー名が含まれているのを確認してください。

b. <insertYourPassword> は省略します。これにより、接続時に mongo シェルによりパスワードの入力を求められます。

c. オプション: IAM 認証を使用している場合、または以前のバージョンの MongoDB シェルを使用している場合は、次のように接続文字列を変更します。

```
mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --username SampleUser1 --password
```

をクラスターの同じ情報mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1に置き換えます。

3. ターミナル上で Enter キーを押します。ここで、パスワードの入力を求められます。パスワードを入力します。

4. パスワードを入力し、rs0 [direct: primary] <env-name>> プロンプトが表示されたら、Amazon DocumentDB クラスターに正常に接続されています。

接続に問題がありますか。Amazon DocumentDB のトラブルシューティング を参照してください。

ステップ 7: データの挿入とクエリ

クラスターに接続できたので、いくつかのクエリを実行して、ドキュメントデータベースの使用に慣れることができます。

1. 1 つのドキュメントを挿入するには、次のように入力します。

```
db.collection.insertOne({"hello":"DocumentDB"})
```

次の出力が返ります。

```
{
  acknowledged: true,
  insertedId: ObjectId('673657216bdf6258466b128c')
}
```

2. findOne() コマンドで書き込んだドキュメントを読み取ることができます (一つのドキュメントしか返さないため)。以下を入力します。

```
db.collection.findOne()
```

次の出力が返ります。

```
{ "_id" : ObjectId("5e401fe56056fda7321fbd67"), "hello" : "DocumentDB" }
```

3. さらにいくつかのクエリを実行するには、ゲームプロファイルのユースケースを検討してください。まず、「profiles」というタイトルのコレクションにエントリをいくつか挿入します。以下を入力します。

```
db.profiles.insertMany([{ _id: 1, name: 'Matt', status: 'active', level: 12, score:
 202 },
      { _id: 2, name: 'Frank', status: 'inactive', level: 2, score: 9 },
      { _id: 3, name: 'Karen', status: 'active', level: 7, score: 87 },
      { _id: 4, name: 'Katie', status: 'active', level: 3, score: 27 }
])
```

次の出力が返ります。

```
{ acknowledged: true, insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4 } }
```

4. find() コマンドを使用して、プロファイルコレクション内のすべてのドキュメントを返します。以下を入力します。

```
db.profiles.find()
```

ステップ 3 で入力したデータと一致する出力を取得します。

5. フィルターを使用して 1 つのドキュメントに対して1つのクエリを使用します。以下を入力します。

```
db.profiles.find({name: "Katie"})
```

次の出力が返ります。

```
{ "_id" : 4, "name" : "Katie", "status": "active", "level": 3, "score":27}
```

6. それでは、プロファイルを探して、それを findAndModify コマンドで変更してみましょう。次のコードを使用して、ユーザー Matt にさらに 10 ポイントを与えます。

```
db.profiles.findAndModify({
        query: { name: "Matt", status: "active"},
        update: { $inc: { score: 10 } }
    })
```

次のような出力が得られます (Mattのスコアはまだ上がっていないことに注意してください) 。

```
{
    [{_id : 1, name : 'Matt', status: 'active', level: 12, score: 202}]
```

7. 次のクエリを使用して、Matt のスコアが変更されたことを確認できます。

```
db.profiles.find({name: "Matt"})
```

次の出力が返ります。

```
{ "_id" : 1, "name" : "Matt", "status" : "active", "level" : 12, "score" : 212 }
```

ステップ 8: 使ってみる

お疲れ様でした。Amazon DocumentDB のクイックスタートガイドを正常に完了しました。

次のステップ この強力なデータベースを一般的な特徴をいくつか使って十分に活用する方法を学びましょう。

- Amazon DocumentDB の管理

- スケーリング
- バックアップと復元

> **Note**
>
> コストを節約するために、Amazon DocumentDB クラスターを停止してコストを削減するか、クラスターを削除することができます。デフォルトでは、30 分間非アクティブになると、 AWS Cloud9 環境は基盤となる Amazon EC2 インスタンスを停止します。

EC2 インスタンスを既存の Amazon DocumentDB データベースに自動的に接続する

次の手順では、既存の Amazon DocumentDB クラスターと既存の Amazon EC2 インスタンスを利用できることを前提としています。

Amazon DocumentDB クラスターにアクセスし、Amazon EC2 接続をセットアップする

1. Amazon DocumentDB クラスターにアクセスします。

   a. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。

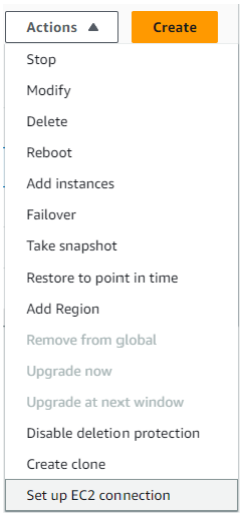
   b. ナビゲーションペインで クラスター を選択します。

      > **Tip**
      >
      > 画面の左側にナビゲーションペインが表示されない場合は、ページの左上隅にあるメニューアイコン (≡) を選択します。

   c. クラスター名の左側にあるボタンを選択して、アクセスしたいクラスターを指定します。

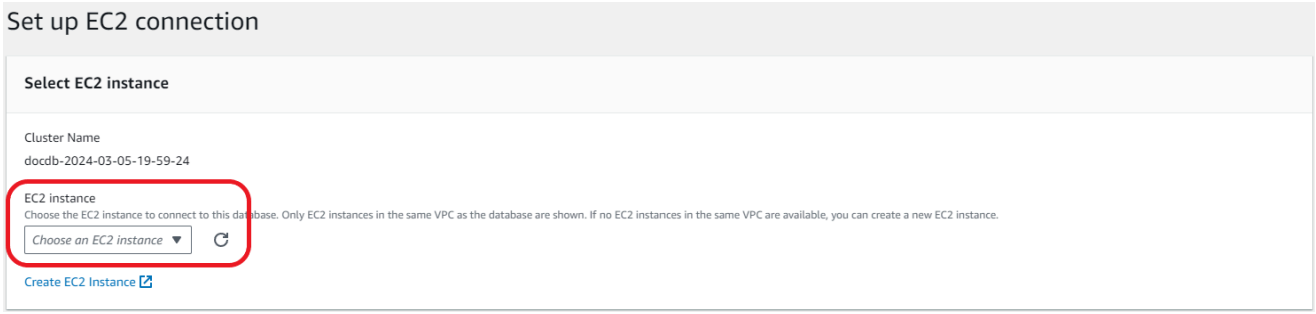
2. Amazon EC2 接続を設定します。

   a. [アクション] をクリックし、[EC2 接続の設定] をクリックします。



[EC2 接続の設定] ダイアログが開きます。

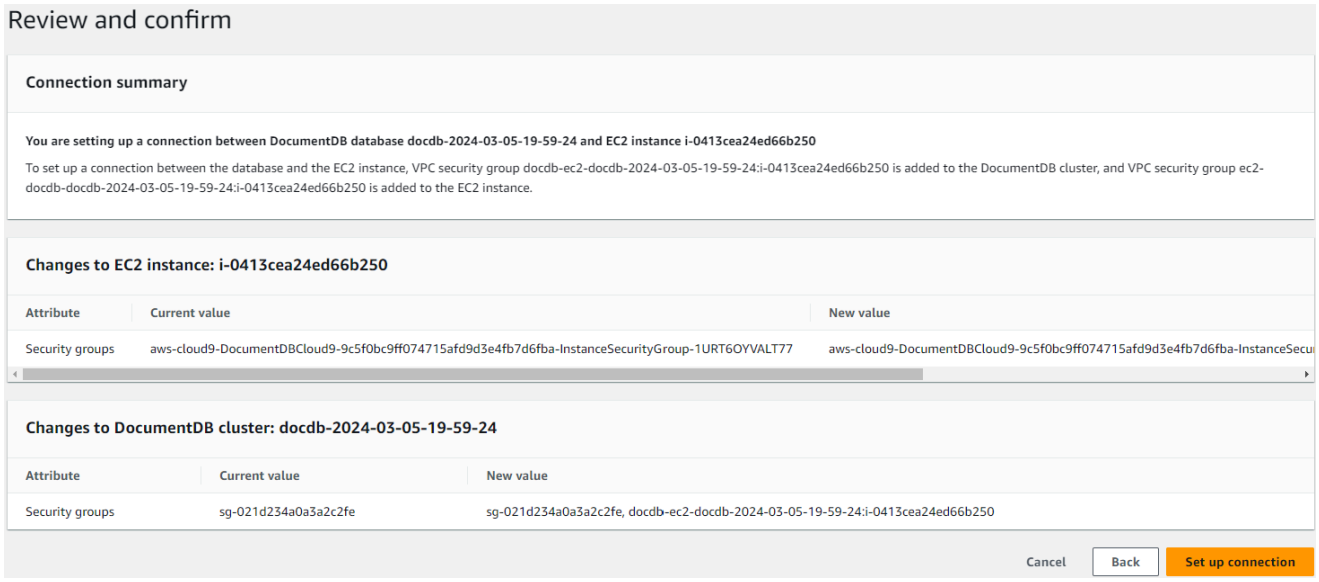
b. [EC2 インスタンス] フィールドで、クラスターに接続する EC2 インスタンスを選択します。



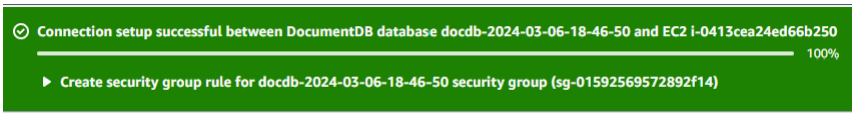
c. [続行] をクリックしてください。

[確認と確定] ダイアログページが表示されます。

d. 変更内容が正しいことを確認します。続いて、[接続の設定] を選択します。



正常に接続が設定されると、以下の検証結果が表示されます。



EC2 インスタンスとの自動接続の概要

EC2 インスタンスと Amazon DocumentDB データベース間の接続を設定すると、Amazon DocumentDB ではその EC2 インスタンスと Amazon DocumentDB データベース向けの VPC セキュリティグループを自動的に設定します。

EC2 インスタンスと Amazon DocumentDB データベースを接続するための要件は次のとおりです。

- EC2 インスタンスは、Amazon DocumentDB データベースと同じ VPC 内に存在する必要があります。

  同じ VPC に EC2 インスタンスが存在しない場合、コンソールには EC2 インスタンス作成用のリンクが表示されます。
- 接続を設定するユーザーには、以下の Amazon EC2 オペレーションを実行するアクセス許可が必要です。
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`

- ec2:DescribeNetworkInterfaces
- ec2:DescribeSecurityGroups
- ec2:ModifyNetworkInterfaceAttribute
- ec2:RevokeSecurityGroupEgress

DB インスタンスと EC2 インスタンスが異なるアベイラビリティーゾーンにある場合、アベイラビリティーゾーン間のコストが発生する可能性があります。

EC2 インスタンスへの接続を設定すると、Amazon DocumentDB は、次の表に示すように、その Amazon DocumentDB データベースと EC2 インスタンスに関連付けられているセキュリティグループの現在の設定に従って動作します。

現在の Amazon DocumentDB セキュリティグループ設定	現在の EC2 セキュリティグループの設定	Amazon DocumentDB の動作
パターン DocumentDB-ec2-n に一致する名前をもつ Amazon DocumentDB データベースに関連付けられたセキュリティグループが 1 つ以上存在します。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	パターン DocumentDB-ec2-n (n は数字) に一致する名前の EC2 インスタンスに関連付けられたセキュリティグループが 1 つ以上存在します。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、Amazon DocumentDB データベースの VPC セキュリティグループをソースとするアウトバウンドルールが 1 つのみ存在します。	Amazon DocumentDB はアクションを実行しません。EC2 インスタンスと Amazon DocumentDB データベース間の接続は、既に自動で設定されています。EC2 インスタンスと Amazon DocumentDB データベース間には既に接続が存在するため、セキュリティグループの変更は行われません。
次の条件のいずれかが適用されます。 • パターン DocumentDB-ec2-n に一致する名前をもつ Amazon DocumentDB データベースに関連付けら	次の条件のいずれかが適用されます。 • EC2 インスタンスに関連付けされた、パターン ec2-DocumentDB-n に一致	Amazon DocumentDB アクション: 新しいセキュリティグループを作成する

現在の Amazon DocumentDB セキュリティグループ設定	現在の EC2 セキュリティグループの設定	Amazon DocumentDB の動作
れたセキュリティグループは存在しません。 • パターン DocumentDB-ec2-n に一致する名前をもつ Amazon DocumentDB に関連付けられたセキュリティグループが 1 つ以上存在します。ただし、Amazon DocumentDB は、これらのセキュリティグループのいずれも、EC2 インスタンスとの接続には使用できません。Amazon DocumentDB は、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つも存在しないセキュリティグループを使用できません。また、Amazon DocumentDB は、変更が加えられたセキュリティグループを使用できません。変更の例としては、ルールの追加や、既存ルールのポート変更などがあります。	する名前のセキュリティグループは存在しません。 • EC2 インスタンスに関連付けられた、パターン ec2-DocumentDB-n に一致する名前のセキュリティグループが 1 つまたは複数存在します。ただし、Amazon DocumentDB は、これらのセキュリティグループのいずれも、Amazon DocumentDB データベースとの接続には使用できません。Amazon DocumentDB は、Amazon DocumentDB データベースの VPC セキュリティグループをソースとするアウトバウンドルールが 1 つもないセキュリティグループを使用できません。また、Amazon DocumentDB は、変更が加えられたセキュリティグループを使用できません。	

現在の Amazon DocumentDB セキュリティグループ設定	現在の EC2 セキュリティグループの設定	Amazon DocumentDB の動作
パターン `DocumentDB-ec2-n` に一致する名前をもつ Amazon DocumentDB データベースに関連付けられたセキュリティグループが 1 つ以上存在します。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	EC2 インスタンスに関連付けられた、パターン `ec2-DocumentDB-n` に一致する名前のセキュリティグループが 1 つまたは複数存在します。ただし、Amazon DocumentDB は、これらのセキュリティグループのいずれも、Amazon DocumentDB データベースとの接続には使用できません。Amazon DocumentDB は、Amazon DocumentDB データベースの VPC セキュリティグループをソースとするアウトバウンドルールが 1 つもないセキュリティグループを使用できません。また、Amazon DocumentDB は、変更が加えられたセキュリティグループを使用できません。	Amazon DocumentDB アクション: 新しいセキュリティグループを作成する

現在の Amazon DocumentDB セキュリティグループ設定	現在の EC2 セキュリティグループの設定	Amazon DocumentDB の動作
パターン `DocumentDB-ec2-n` に一致する名前をもつ Amazon DocumentDB データベースに関連付けられたセキュリティグループが 1 つ以上存在します。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	接続に有効な EC2 セキュリティグループは存在しますが、EC2 インスタンスに関連付けられていません。このセキュリティグループには、パターン `DocumentDB-ec2-n` に一致する名前が付いています。このセキュリティグループには変更が加えられていません。このセキュリティグループには、Amazon DocumentDB データベースの VPC セキュリティグループをソースとするアウトバウンドルールが 1 つのみ存在します。	Amazon DocumentDB アクション: EC2 セキュリティグループを関連付ける

現在の Amazon DocumentDB セキュリティグループ設定	現在の EC2 セキュリティグループの設定	Amazon DocumentDB の動作
次の条件のいずれかが適用されます。 • パターン DocumentDB-ec2-n に一致する名前をもつ Amazon DocumentDB データベースに関連付けられたセキュリティグループは存在しません。 • パターン DocumentDB-ec2-n に一致する名前をもつ Amazon DocumentDB データベースに関連付けられたセキュリティグループが 1 つ以上存在します。ただし、Amazon DocumentDB は、これらのセキュリティグループのいずれも、EC2 インスタンスとの接続には使用できません。Amazon DocumentDB は、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つも存在しないセキュリティグループを使用できません。また、Amazon DocumentDB は、変更が加えられたセキュリティグループを使用できません。	EC2 インスタンスに関連付けられた、パターン DocumentDB-ec2-n に一致する名前のセキュリティグループが 1 つまたは複数存在します。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、Amazon DocumentDB データベースの VPC セキュリティグループをソースとするアウトバウンドルールが 1 つのみ存在します。	Amazon DocumentDB アクション: 新しいセキュリティグループを作成する

Amazon DocumentDB アクション: 新しいセキュリティグループを作成する

Amazon DocumentDB は以下のアクションを実行します。

- パターン `DocumentDB-ec2-n` に一致する新しい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には、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが存在します。このセキュリティグループは Amazon DocumentDB データベースに関連付けられており、EC2 インスタンスが Amazon DocumentDB データベースにアクセスするのを許可します。
- パターン `ec2-DocumentDB-n` に一致する新しい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には、Amazon DocumentDB データベースの VPC セキュリティグループをソースとするアウトバウンドルールが存在します。このセキュリティグループには EC2 インスタンスに関連付けられており、EC2 インスタンスが Amazon DocumentDB データベースにトラフィックを送信するのを許可します。

Amazon DocumentDB アクション: EC2 セキュリティグループを関連付ける

Amazon DocumentDB は、有効な既存の EC2 セキュリティグループを EC2 インスタンスに関連付けます。このセキュリティグループは、EC2 インスタンスが Amazon DocumentDB データベース にトラフィックを送信するのを許可します。

接続中のコンピューティングリソースを表示する

を使用して AWS Management Console 、Amazon DocumentDB データベースに接続されているコンピューティングリソースを表示できます。表示されるリソースには、自動的に設定されたコンピューティングリソース接続が含まれます。コンピューティングリソースとの接続は、次の方法で自動的に設定できます。

- データベースを作成するときに、コンピューティングリソースを選択できます。詳細については、「Amazon DocumentDB クラスターの作成」および「マルチ AZ DB クラスターの作成」を参照してください。
- 既存のデータベースとコンピューティングリソース間の接続を設定できます。詳細については、「Amazon EC2 への自動接続」を参照してください。

コンピューティングリソースリストには、手動でデータベースに接続されたものは含まれていません。例えば、データベースに関連付けられた VPC セキュリティグループにルールを追加することで、コンピューティングリソースがデータベースに手動でアクセスできるようになります。

コンピューティングリソースをリスト化するには、次の条件を満たしている必要があります。

- コンピューティングリソースに関連付けられているセキュリティグループの名前がパターン `ec2-DocumentDB-n` (n は数字) と一致する。
- コンピューティングリソースに関連付けられたセキュリティグループには、Amazon DocumentDB データベースが使用するポート向けに設定されたポート範囲を持つアウトバウンドルールがある。
- コンピューティングリソースに関連付けられたセキュリティグループには、Amazon DocumentDB データベースに関連付けられたセキュリティグループをソースとするアウトバウンドルールがある。
- Amazon DocumentDB データベースに関連付けられたセキュリティグループの名前が、パターン `DocumentDB-ec2-n` (n は数字) に一致する。
- Amazon DocumentDB データベースに関連付けられたセキュリティグループには、Amazon DocumentDB データベースが使用するポート向けに設定されたポート範囲を持つインバウンドルールがある。
- Amazon DocumentDB データベースに関連付けられたセキュリティグループには、コンピューティングリソースに関連付けられたセキュリティグループをソースとするインバウンドルールがある。

Amazon DocumentDB データベースに接続されているコンピューティングリソースを確認するには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/docdb で Amazon DocumentDB コンソールを開きます。
2. ナビゲーションペインで、[データベース] を選択し、確認したい Amazon DocumentDB データベースの名前を選択します。
3. [接続とセキュリティ] タブ内の [接続されたコンピューティングリソース] で、接続されているコンピューティングリソースを確認します。

Amazon EC2 を手動で接続する

トピック

- ステップ 1: Amazon EC2 インスタンスを作成する
- ステップ 2: セキュリティグループを作成する
- ステップ 3: Amazon DocumentDB クラスターを作成する
- ステップ 4: Amazon EC2 インスタンスに接続します
- ステップ 5: MongoDB シェルをインストールする
- ステップ 6: Amazon DocumentDB TLS の管理

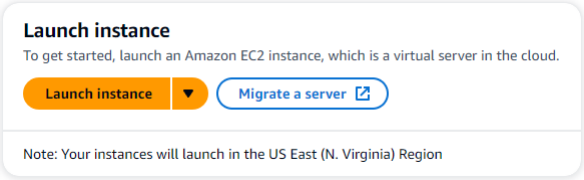
- ステップ 7: Amazon DocumentDB クラスターに接続する
- ステップ 8: データの挿入とクエリ
- ステップ 9: 環境情報の確認

以下の手順は、「前提条件」トピックに記載されるステップが完了済みであることを前提としています。

ステップ 1: Amazon EC2 インスタンスを作成する

このステップでは、後で Amazon DocumentDB クラスターのプロビジョニングに使用するのと同じリージョンと Amazon VPC に Amazon EC2 インスタンスを作成します。

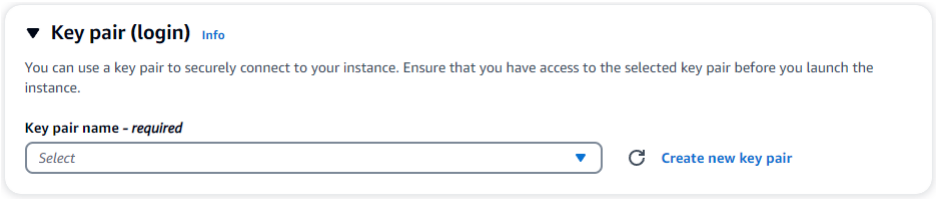
1. Amazon EC2 コンソールで、[インスタンスを起動] を選択します。



2. [名前とタグ] セクション内の [名前] フィールドに、名前または識別子を入力します。
3. [Amazon マシンイメージ (AMI)] ドロップダウンリストで、[Amazon Linux 2 AMI] を選択します。



4. [インスタンスタイプ] のドロップダウンリストで、[t3.micro] を選択します。
5. [キーペア (ログイン)] セクションで、既存のキーペアの識別子を入力するか、もしくは [新しいキーペアの作成] を選択します。



Amazon EC2 キーペアを指定する必要があります。

Amazon EC2 のキーペアがある場合 :

a. キーペアを選択し、リストからご自身のキーペアを選択します。

b. Amazon EC2 インスタンスにログインするには、あからじめプライベートキーファイル (.pem もしくは.ppk ファイル) を利用可能にしておく必要があります。

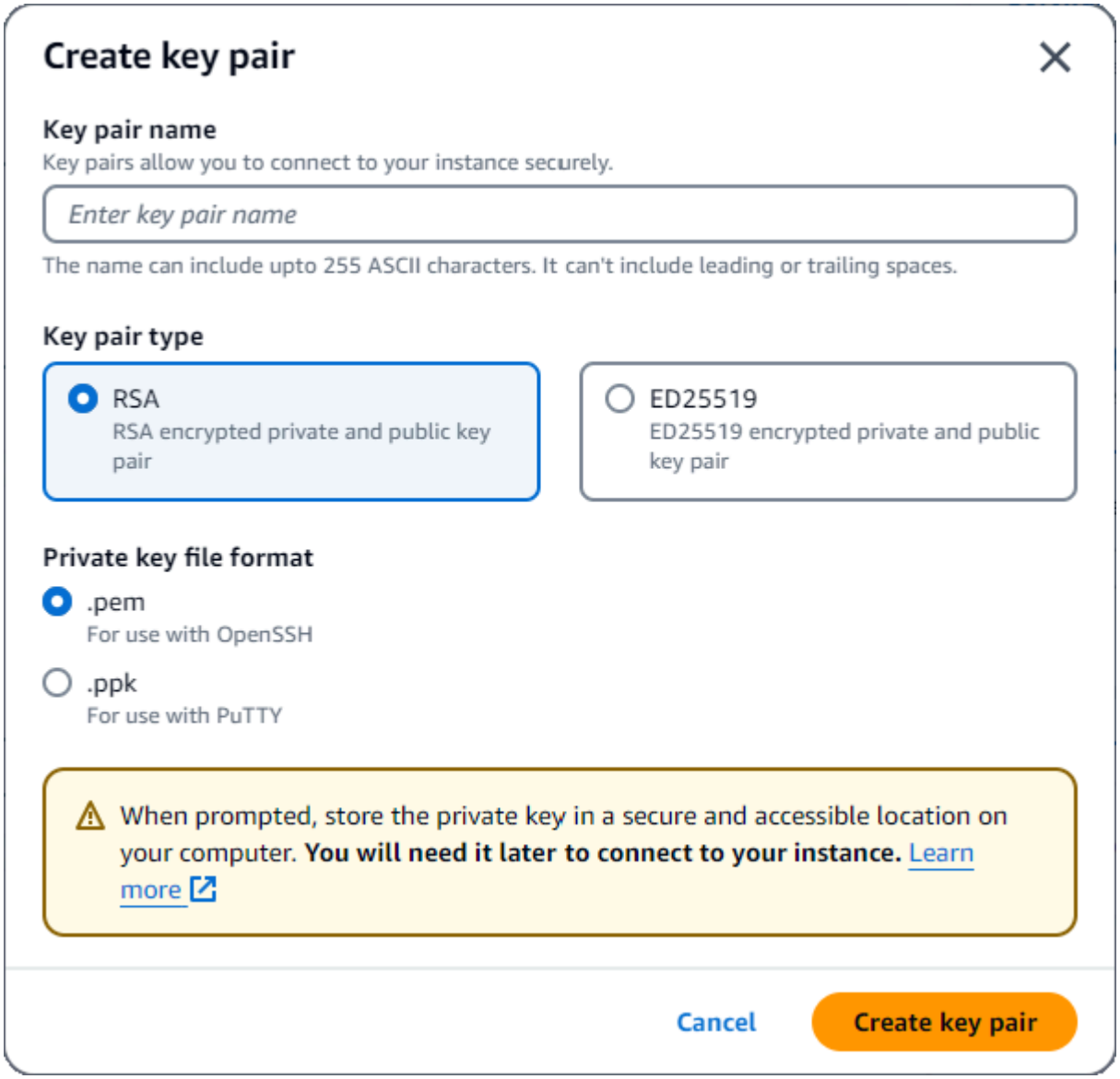
Amazon EC2 のキーペアがない場合:

a. [新しいキーペアの作成] を選択すると、[キーペアの作成] ダイアログボックスが表示されます。

b. [キーペア名] フィールドに名前を入力します。

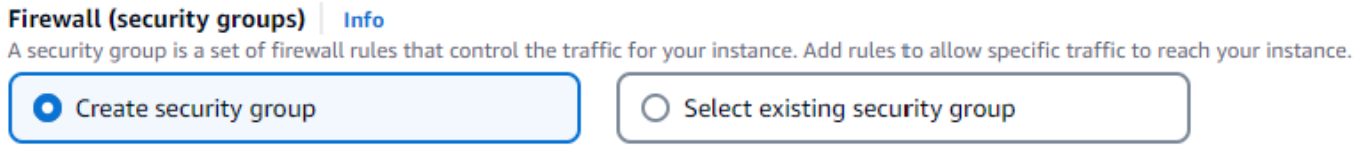
c. [キーペアタイプ] と [プライベートキーファイル形式] を選択します。

d. [キーペアの作成] を選択してください。



> **Note**
>
> セキュリティ上の理由から、EC2 インスタンスへの SSH とインターネット接続の両方にキーペアを使用することを強くお勧めします。

6. ネットワーク設定セクションのファイアウォール (セキュリティグループ) で、セキュリティグループの作成または既存のセキュリティグループの選択を選択します。



既存のセキュリティグループを選択する場合は、[共通セキュリティグループ] ドロップダウンリストから選択します。

新しいセキュリティグループの作成を選択する場合は、以下を実行します。

a. EC2 への接続に適用されるすべてのトラフィック許可ルールを確認します。

b. IP フィールドで、My IP を選択するか、Custom を選択して CIDR ブロック、プレフィックスリスト、またはセキュリティグループのリストから選択します。ここで [Anywhere] を選択するとどの IP アドレスからでも EC2 インスタンス へのアクセスが可能になるため、EC2 インスタンスが分離されたネットワーク上にある場合を除き、[Anywhere] を選択することはお勧めしません。

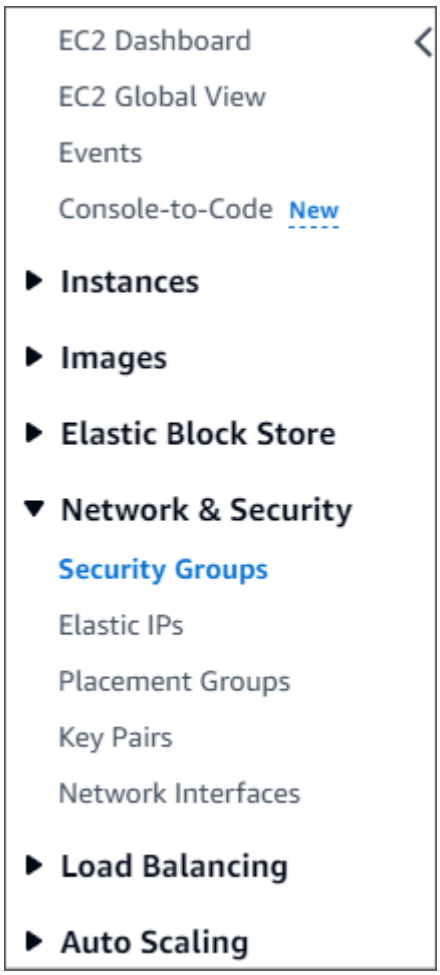


7. 概要セクションで、EC2 設定を確認し、正しい場合はインスタンスを起動を選択します。

ステップ 2: セキュリティグループを作成する

次に、デフォルトの Amazon VPC に新しいセキュリティグループを作成します。セキュリティグループ demoDocDB では、Amazon EC2 インスタンスからポート 27017 (Amazon DocumentDB のデフォルトポート) で Amazon DocumentDB クラスターに接続できます。

1. Amazon EC2 マネジメントコンソール 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。



2. [セキュリティグループの作成] を選択してください。



3. 基本の詳細セクションで、次の操作を行います。

   a. [Security group name (セキュリティグループ名)] に「 demoDocDB 」と入力します。

   b. [説明] に説明を入力します。

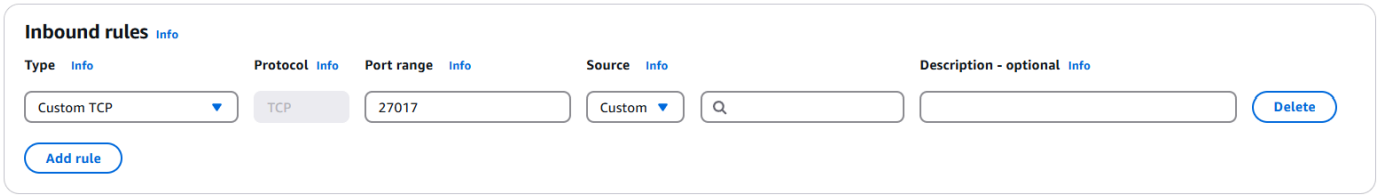
   c. VPC については、デフォルトの VPC の使用を受け入れます。

4. [インバウンドルール] セクションで、[ルールの追加] を選択します。

   a. Type で、カスタム TCP ルール (デフォルト) を選択します。

   b. [ポート範囲] に 27017 と入力します。

   c. [ソース] で [カスタム] を選択します。その横の フィールドで、ステップ 1 で作成したセキュリティグループを検索します。 のソース名を自動入力するには、Amazon EC2 コンソールでブラウザを更新することが必要な場合があります。



5. 他のすべてのデフォルトを受け入れ、[セキュリティグループの作成] を選択します。



ステップ 3: Amazon DocumentDB クラスターを作成する

Amazon EC2 インスタンスのプロビジョニング中は、Amazon DocumentDB クラスターを作成します。

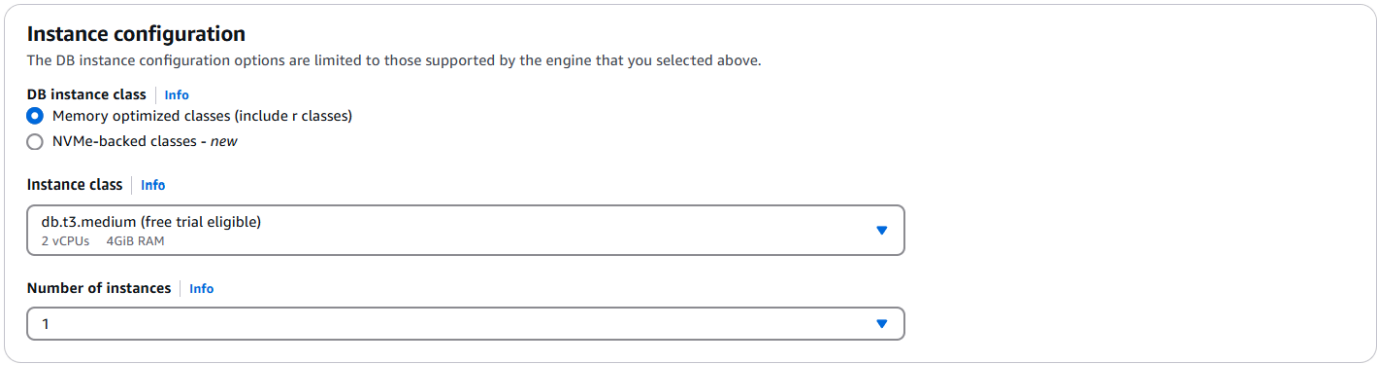
1. Amazon DocumentDB コンソールに移動し、ナビゲーションペインから クラスター を選択します。
2. [作成] を選択します。
3. [クラスタータイプ] 設定は、デフォルトの [インスタンスベースのクラスター] のままにしておきます。
4. クラスター設定で、クラスター識別子に一意の名前を入力します。コンソール操作では、入力方法に関係なくクラスター名のすべての文字が小文字に変換されることにご留意ください。

   エンジンバージョンはデフォルト値の 5.0.0 のままにします。
5. クラスターストレージ設定の場合、Amazon DocumentDB Standard のデフォルト設定のままにします。
6. インスタンス設定：

   - DB インスタンスクラスの場合は、メモリ最適化クラス (r クラスを含む) を選択します (デフォルト)。

     もう 1 つのインスタンスオプションは NVMe-backed クラスです。詳細については[NVMe-backed インスタンス]を参照してください。
   - インスタンスクラスで、ニーズに合ったインスタンスタイプを選択します。インスタンスクラスの詳細については、「」を参照してください[インスタンスクラスの仕様]。

- インスタンスの数については、ニーズに最も適した数を選択します。数値が低いほど、コストが低くなり、クラスターで管理できる読み取り/書き込みボリュームが低くなることに注意してください。



7. [接続] は、デフォルト設定の [EC2 コンピューティングリソースに接続しない] のままにしておきます。

> **Note**
>
> EC2 コンピューティングリソースに接続すると、クラスターへの接続のセキュリティグループが自動的に作成されます。前のステップでセキュリティグループを既に手動で作成しているため、追加のセキュリティグループのセットが作成されないように、[EC2 コンピューティングリソースに接続しない] を選択しておく必要があります。

8. 認証セクションで、プライマリユーザーのユーザー名を入力し、セルフマネージドを選択します。パスワードを入力して確認します。

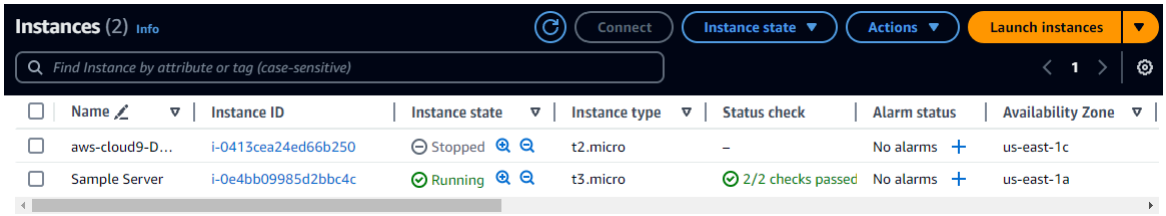
   代わりに Managed in AWS Secrets Manager を選択した場合は、詳細については[Amazon DocumentDB と によるパスワード管理 AWS Secrets Manager]「」を参照してください。

9. [クラスターを作成] を選択します。

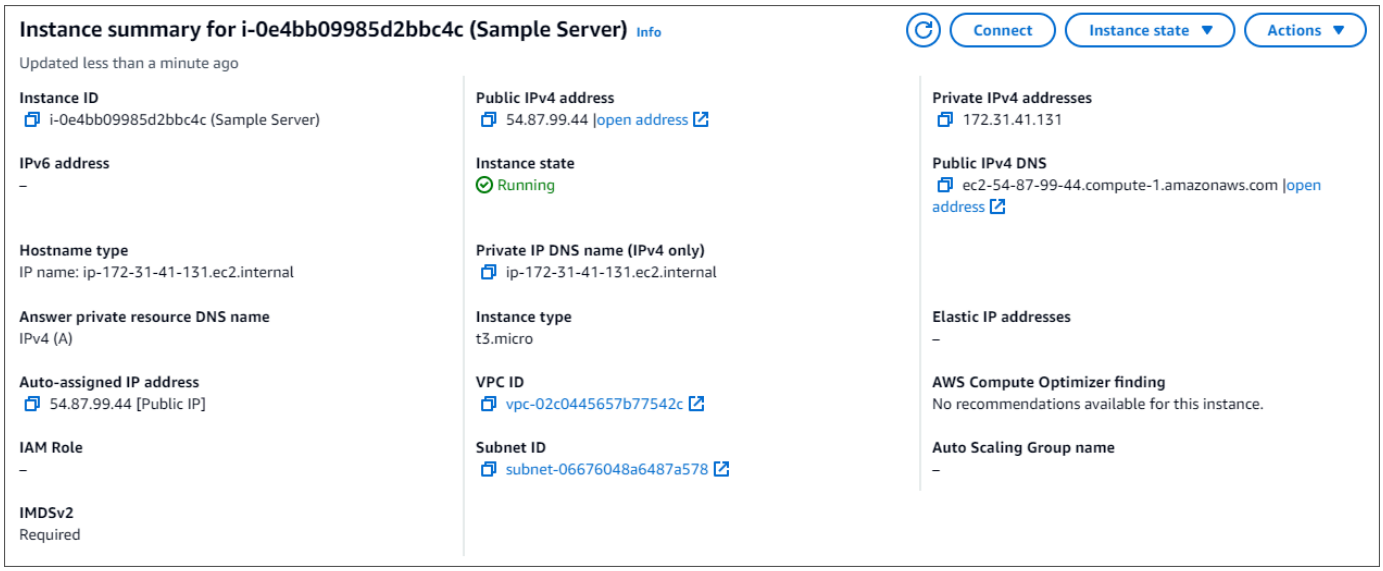
ステップ 4: Amazon EC2 インスタンスに接続します

Amazon EC2 インスタンスに接続すると、MongoDB シェルをインストールできます。Mongo シェルをインストールすると、Amazon DocumentDB クラスターに接続してクエリを実行できます。以下のステップを実行します。

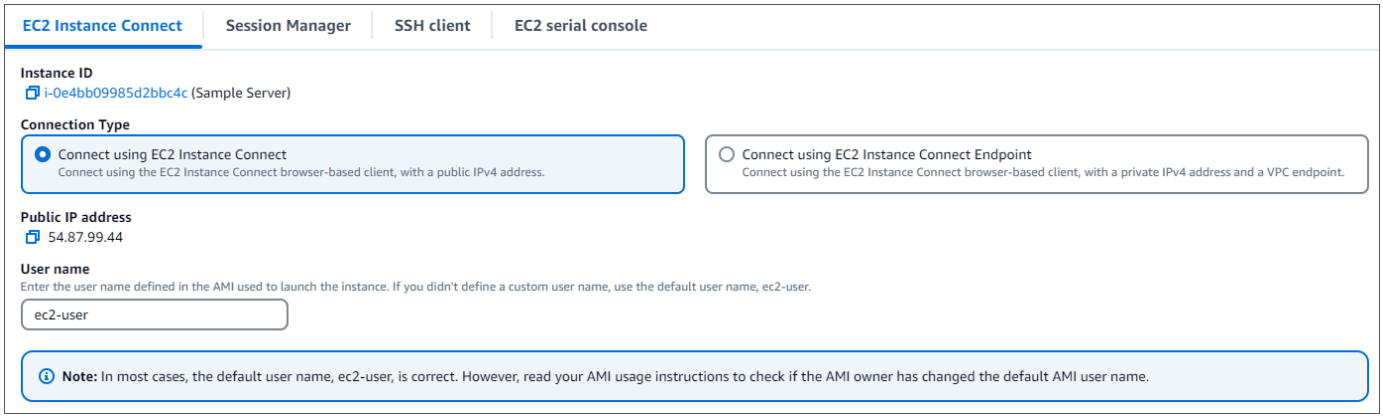
1. Amazon EC2 コンソールで、インスタンスに移動し、作成したインスタンスが実行中かどうかを確認します。実行されている場合は、インスタンス ID をクリックしてインスタンスを選択します。



2. [接続] を選択します。



3. 接続方法として、Amazon EC2 インスタンス Connect、セッションマネージャー、SSH クライアント、EC2 シリアルコンソールの 4 つのオプションがタブ形式で表示されます。いずれかを選択し、その指示に従わなければなりません。完了したら、[接続する] を選択します。



> **Note**
>
> このウォークスルーの開始後に IP アドレスが変更された場合、または後で環境に戻る場合は、demoEC2セキュリティグループのインバウンドルールを更新して、新しい API アドレスからのインバウンドトラフィックを有効にする必要があります。

ステップ 5: MongoDB シェルをインストールする

Amazon DocumentDB クラスターの接続とクエリに使用するコマンドラインユーティリティである MongoDB シェルをインストールできるようになりました。 Amazon DocumentDB 現在、MongoDB シェルには最新バージョンの mongosh と以前のバージョンの mongo シェルの 2 つのバージョンがあります。

> **Important**
>
> バージョン 6.13.1 より前の Node.js ドライバーには既知の制限がありますが、現在 Amazon DocumentDB の IAM ID 認証ではサポートされていません。Node.js ドライバーを使用する Node.js ドライバーとツール (mongosh など) は、Node.js ドライバーバージョン 6.13.1 以降を使用するようにアップグレードする必要があります。

以下の手順に従って、オペレーティングシステムの MongoDB シェルをインストールします。

On Amazon Linux

Amazon Linux に MongoDB シェルをインストールするには

IAM を使用しておらず、最新の MongoDB シェル (mongosh) を使用して Amazon DocumentDB クラスターに接続する場合は、次の手順に従います。

1. リポジトリファイルを作成します。作成した EC2 インスタンスのコマンドラインで、次のコマンドを実行します。

```
echo -e "[mongodb-org-5.0] \nname=MongoDB Repository\nbaseurl=https://
repo.mongodb.org/yum/amazon/2023/mongodb-org/5.0/x86_64/\ngpgcheck=1 \nenabled=1
 \ngpgkey=https://pgp.mongodb.com/server-5.0.asc" | sudo tee /etc/yum.repos.d/
mongodb-org-5.0.repo
```

2. 完了したら、コマンドプロンプトで次の 2 つのコマンドオプションのいずれかを使用して mongosh をインストールします。

   オプション 1 — Amazon EC2 設定中にデフォルトの Amazon Linux 2023 を選択した場合は、次のコマンドを入力します。

   ```
   sudo yum install -y mongodb-mongosh-shared-openssl3
   ```

   オプション 2 — Amazon EC2 設定中に Amazon Linux 2 を選択した場合は、次のコマンドを入力します。

   ```
   sudo yum install -y mongodb-mongosh
   ```

IAM を使用している場合は、以前のバージョンの MongoDB シェル (5.0) を使用して Amazon DocumentDB クラスターに接続する必要があります。次の手順を実行します。

1. リポジトリファイルを作成します。作成した EC2 インスタンスのコマンドラインで、次のコマンドを実行します。

   ```
   echo -e "[mongodb-org-5.0] \nname=MongoDB Repository\nbaseurl=https://
   repo.mongodb.org/yum/amazon/2023/mongodb-org/5.0/x86_64/\ngpgcheck=1 \nenabled=1
    \ngpgkey=https://pgp.mongodb.com/server-5.0.asc" | sudo tee /etc/yum.repos.d/
   mongodb-org-5.0.repo
   ```

2. 完了したら、コマンドプロンプトで次のコマンドオプションを使用して mongodb 5.0 シェルをインストールします。

   ```
   sudo yum install -y mongodb-org-shell
   ```

On Ubuntu

Ubuntu に mongosh をインストールするには

1. パッケージ管理システムで使用されるパブリックキーをインポートします。

   ```
   curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
   share/keyrings/mongodb-server-5.0.gpg
   ```

2. Ubuntu のバージョンに適切なコマンドを使用して、MongoDB の リストファイル `mongodb-org-5.0.list` を作成します。

```
echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-
server-5.0.gpg ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/5.0
 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list
```

3. 次のコマンドを使用して、ローカルパッケージデータベースをインポートおよび更新します。

```
sudo apt-get update
```

4. mongosh をインストールします。

```
sudo apt-get install -y mongodb-mongosh
```

以前のバージョンの MongoDB の Ubuntu システムへのインストールについては、「Install MongoDB Community Edition on Ubuntu」を参照してください。

On other operating systems

mongo シェルを他のオペレーティングシステムにインストールするには、MongoDB ドキュメントの「MongoDB Community Edition のインストール」を参照してください。

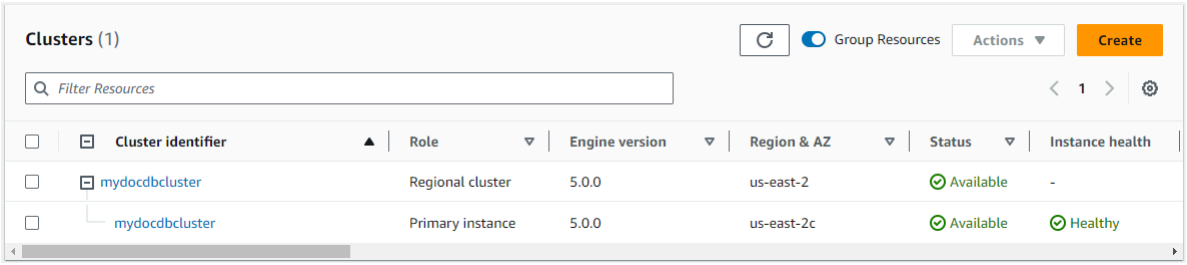
ステップ 6: Amazon DocumentDB TLS の管理

次のコード `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem` を使用して Amazon DocumentDB の CA 証明書をダウンロードします。

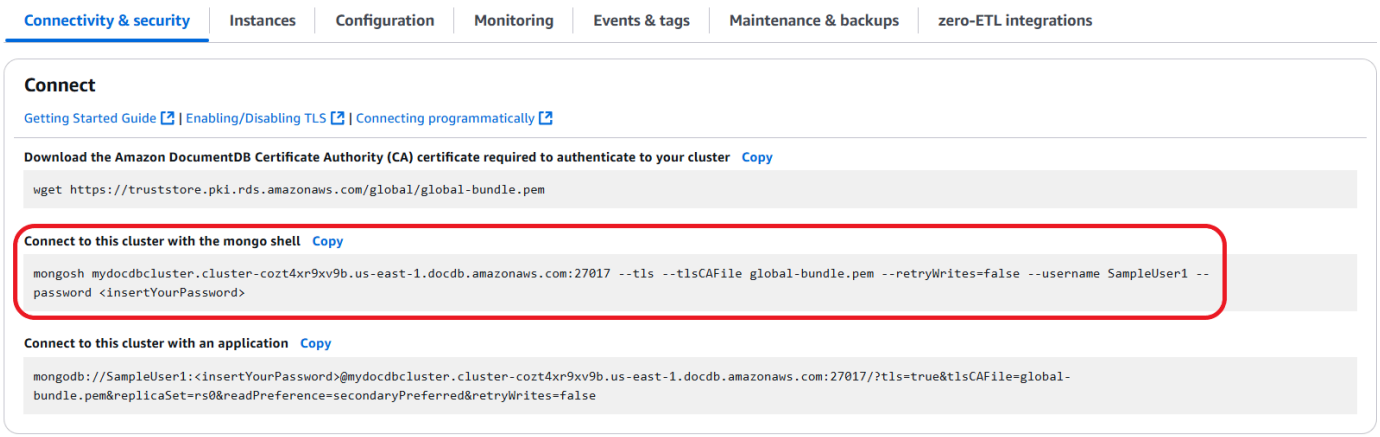
> **Note**
>
> 全ての新しい Amazon DocumentDB クラスターに対して、デフォルトで Transport Layer Security (TLS) が有効になっています。詳細については、「Amazon DocumentDB クラスター TLS 設定の管理」を参照してください。

ステップ 7: Amazon DocumentDB クラスターに接続する

1. Amazon DocumentDB コンソールの [クラスター] で、接続したいクラスターを見つけます。そのクラスターのクラスター識別子をクリックして、作成したクラスターを選択します。



2. [接続とセキュリティ] タブの [接続する] ボックス内で、[mongo シェルを使用してこのクラスターに接続する] を見つけます。



表示された接続文字列をコピーし、ターミナルに貼り付けます。

これに以下の変更を加えます。

a. 文字列に正しいユーザー名が含まれているのを確認してください。

b. <insertYourPassword> は省略します。これにより、接続時に mongo シェルによりパスワードの入力を求められます。

c. オプション: IAM 認証を使用している場合、または以前のバージョンの MongoDB シェルを使用している場合は、次のように接続文字列を変更します。

```
mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-
east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --
username SampleUser1 --password
```

をクラスターの同じ情報mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1に置き換えます。

3. ターミナル上で Enter キーを押します。ここで、パスワードの入力を求められます。パスワードを入力します。
4. パスワードを入力し、rs0 [direct: primary] <env-name>> プロンプトが表示されたら、Amazon DocumentDB クラスターに正常に接続されています。

接続に問題がありますか。Amazon DocumentDB のトラブルシューティング を参照してください。

ステップ 8: データの挿入とクエリ

クラスターに接続できたので、いくつかのクエリを実行して、ドキュメントデータベースの使用に慣れることができます。

1. 1 つのドキュメントを挿入するには、次のように入力します。

```
db.collection.insertOne({"hello":"DocumentDB"})
```

次の出力が返ります。

```
{
  acknowledged: true,
  insertedId: ObjectId('673657216bdf6258466b128c')
}
```

2. findOne() コマンドで書き込んだドキュメントを読み取ることができます (一つのドキュメントしか返さないため)。以下を入力します。

```
db.collection.findOne()
```

次の出力が返ります。

```
{ "_id" : ObjectId("5e401fe56056fda7321fbd67"), "hello" : "DocumentDB" }
```

3. さらにいくつかのクエリを実行するには、ゲームプロファイルのユースケースを検討してください。まず、「profiles」というタイトルのコレクションにエントリをいくつか挿入します。以下を入力します。

```
db.profiles.insertMany([{ _id: 1, name: 'Matt', status: 'active', level: 12, score:
 202 },
      { _id: 2, name: 'Frank', status: 'inactive', level: 2, score: 9 },
      { _id: 3, name: 'Karen', status: 'active', level: 7, score: 87 },
      { _id: 4, name: 'Katie', status: 'active', level: 3, score: 27 }
])
```

次の出力が返ります。

```
{ acknowledged: true, insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4 } }
```

4. find() コマンドを使用して、プロファイルコレクション内のすべてのドキュメントを返します。以下を入力します。

```
db.profiles.find()
```

ステップ 3 で入力したデータと一致する出力を取得します。

5. フィルターを使用して 1 つのドキュメントに対して1つのクエリを使用します。以下を入力します。

```
db.profiles.find({name: "Katie"})
```

次の出力が返ります。

```
{ "_id" : 4, "name" : "Katie", "status": "active", "level": 3, "score":27}
```

6. それでは、プロファイルを探して、それを findAndModify コマンドで変更してみましょう。次のコードを使用して、ユーザー Matt にさらに 10 ポイントを与えます。

```
db.profiles.findAndModify({
        query: { name: "Matt", status: "active"},
        update: { $inc: { score: 10 } }
    })
```

次のような出力が得られます (Mattのスコアはまだ上がっていないことに注意してください) 。

```
{
    [{_id : 1, name : 'Matt', status: 'active', level: 12, score: 202}]
```

7. 次のクエリを使用して、Matt のスコアが変更されたことを確認できます。

   ```
   db.profiles.find({name: "Matt"})
   ```

   次の出力が返ります。

   ```
   { "_id" : 1, "name" : "Matt", "status" : "active", "level" : 12, "score" : 212 }
   ```

ステップ 9: 環境情報の確認

お疲れ様でした。Amazon DocumentDB のクイックスタートガイドを正常に完了しました。

次のステップ この強力なデータベースを一般的な特徴をいくつか使って十分に活用する方法を学びましょう。

- Amazon DocumentDB の管理
- スケーリング
- バックアップと復元

> **Note**
>
> コストを節約するために、Amazon DocumentDB クラスターを停止してコストを削減するか、クラスターを削除することができます。デフォルトでは、30 分間非アクティブになると、 AWS Cloud9 環境は基盤となる Amazon EC2 インスタンスを停止します。

Amazon DocumentDB JDBC ドライバーを使用して接続する

Amazon DocumentDB 用の JDBC ドライバーは、デベロッパーのための SQL リレーショナルインターフェイスを提供し、Tableau や DBVisualizer などの BI ツールからの接続を可能にします。

詳細については、GitHub 上の Amazon DocumentDB JDBC ドライバーのドキュメント を参照してください。

トピック

- 入門
- Tableau デスクトップから Amazon DocumentDB に接続する

- DbVisualizer から Amazon DocumentDB に接続する
- JDBC スキーマの自動生成
- SQL サポートと制限
- トラブルシューティング

入門

ステップ 1. Amazon DocumentDB クラスターの作成

Amazon DocumentDB クラスターが作成されていない場合は、Amazon DocumentDB デベロッパーガイドの 開始方法 のセクションの手順を使用してクラスターを作成します。

> **Note**
>
> Amazon DocumentDB は仮想プライベートクラウド (VPC) 専用サービスです。クラスターの VPC 外にあるローカルマシンから接続する場合は、Amazon EC2 インスタンスへの SSH 接続を作成する必要があります。この場合、EC2 を使用して接続 の手順に従ってクラスターを起動します。SSH トンネリングおよび必要な場合の詳細については、「SSH トンネルを使って Amazon DocumentDB に接続する」を参照してください。

ステップ 2. JRE または JDK インストール

BI アプリケーションによっては、64 ビット JRE または JDK インストールバージョン 8 以降がコンピュータにインストールされていることを確認する必要がある場合があります。Java SE ランタイム環境 8 をダウンロードは こちら。

ステップ 3. DocumentDB JDBC ドライバーのダウンロード

here(こちら) からのDocumentDB JDBC ドライバーのダウンロード ドライバーは単一の JAR ファイルとしてパッケージ化されています (例: documentdb-jdbc-1.0.0-all.jar)。

ステップ 4. SSH トンネルを使用して Amazon DocumentDB に接続する

Amazon DocumentDB (MongoDB との互換性) クラスターは、Amazon Virtual Private Cloud (Amazon VPC) 内にデプロイされます。これらは、Amazon EC2 インスタンス、または同じ Amazon VPC にデプロイされている他の AWS サービスから直接アクセスできます。さらに、Amazon DocumentDB には、同じリージョンまたは他の AWS リージョンの異なる VPCs 内の EC2a インスタンスまたは他の AWS サービスが VPC ピアリングを介してアクセスできます。

SSH トンネリング (ポート転送とも呼ばれます) を使用して、クラスターの VPC 外部から Amazon DocumentDB リソースにアクセスできます。これは、ほとんどのユーザーが DocumentDB クラスターと同じ VPC 内の仮想マシンでアプリケーションを実行していない場合です。

SSH トンネルを作成するには、Amazon DocumentDB クラスターと同じ Amazon VPC で実行中の Amazon EC2 インスタンスが必要です。クラスターと同じ VPC で既存の EC2 インスタンスを使用するか、新しい EC2 インスタンスを作成できます。その場合、ローカルコンピュータで以下のコマンドを実行することにより Amazon DocumentDB クラスター `sample-cluster.node.us-east-1.docdb.amazonaws.com` に SSH トンネルを設定できます。

```
ssh -i "ec2Access.pem" -L 27017:sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017 ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com -
N
```

ローカルポートの転送を行う場合は、-L フラグを使用します。これは、VPC 外のクライアントで実行されている BI ツールに接続するための前提条件です。上記のステップを実行したら、選択した BI ツールの次のステップに進むことができます。

SSH トンネリングの詳細については、SSH トンネルを使用して Amazon DocumentDB に接続する のドキュメントを参照してください。

Tableau デスクトップから Amazon DocumentDB に接続する

トピック

- Amazon DocumentDB JDBC ドライバーの追加
- Tableau を使用して Amazon DocumentDB に接続する - SSH トンネル

Amazon DocumentDB JDBC ドライバーの追加

Tableau デスクトップから Amazon DocumentDB に接続するには、DocumentDB JDBC ドライバーと DocumentDB Tableau コネクタをダウンロードしてインストールする必要があります。

1. Amazon DocumentDB JDBC ドライバーリポジトリから Amazon DocumentDB JDBC ドライバー JAR ファイルをダウンロードし、オペレーティングシステムに応じてこれらのディレクトリのいずれかにコピーします。

- Windows – `C:\Program Files\Tableau\Drivers`
- MacOS – `~/Library/Tableau/Drivers`

2. Tableau Exchange ウェブサイトから DocumentDB Tableau コネクタ (TACO ファイル) をダウンロードし、My Tableau Repository/Connectors ディレクトリにコピーします。

   - Windows – `C:\Users\[user]\Documents\My Tableau Repository\Connectors`
   - MacOS – `/Users/[user]/Documents/My Tableau Repository/Connectors`

詳細 (追加情報) については、Tableau のドキュメント を参照してください。

> **Note**
>
> 新しい CA 証明書を使用している場合は、JDBC ドライバーを v1.4.5 (この AWS GitHub リポジトリで利用可能) にアップグレードしてください。

Tableau を使用して Amazon DocumentDB に接続する - SSH トンネル

DocumentDB クラスターの VPC 外のクライアントマシンから Tableau に接続するには、以下のステップを実行する前に SSH トンネルを設定する必要があります。

1. Tableau デスクトップアプリケーションを起動します。
2. Connect > To A Server > More に移動します。
3. [インストールされたコネクタ] で Amazon Web Services の Amazon DocumentDB を選択します。

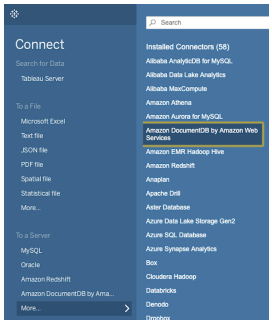
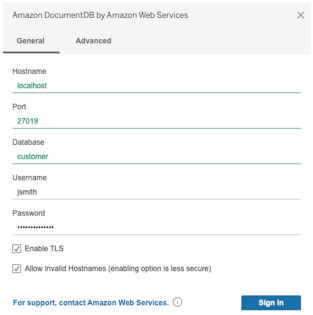


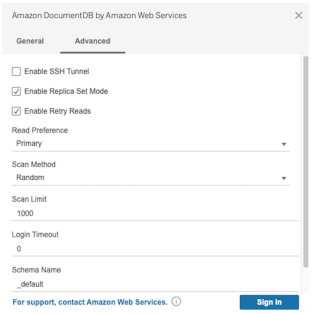
Tableau を使用して Amazon DocumentDB に接続する - 外部 SSH トンネル

1. 必要な接続パラメータを入力します。Hostname、ポート、データベース、ユーザーネーム、パスワード です。以下の例の接続パラメータは、JDBC 接続文字列に相当します。

   ユーザーネームとパスワードのパラメータを伴う `jdbc:documentdb://localhost:27019/test?tls=true&tlsAllowInvalidHostnames=true&scanMethod=random&scanLimit=1000&logir` は、プロパティコレクションで別々に渡されました。接続文字列パラメータの詳細については、Amazon DocumentDB JDBC ドライバー github ドキュメント を参照してください。



2. (オプション) より高度なオプションは、アドバンスト タブで見つけることができます。

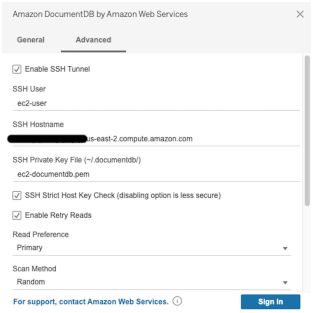


3. [Sign in] (サインイン) を選択します。

Tableau を使用して Amazon DocumentDB に接続する - SSH トンネル

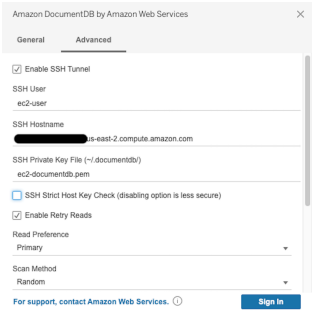
> **Note**
>
> 端末を使用して SSH トンネルをセットアップしたくない場合は、Tableau GUI を使用して、JDBC ドライバーが本質的に SSH トンネルの作成に使用する EC2 インスタンスの詳細を指定できます。

1. アドバンスト タブで、SSH トンネルオプションを有効にする を選択して、その他のプロパティを確認します。



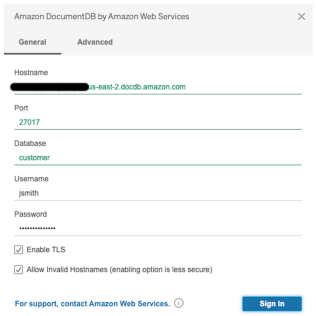
2. SSH ユーザー、SSH ホスト名、SSH プライベートキーファイル を入力します。
3. (オプション) SSH 厳密ホストキーチェック オプションを無効にできます。これを使用すると、既知のホストファイルに対するホストキーのチェックをバイパスします。

> **Note**
>
> このオプションを無効にすると、中間者 攻撃につながり得るので、安全性が低くなります。



4. 必要な接続パラメータを入力します。Hostname、ポート、データベース、ユーザーネーム、パスワード です。

> **Note**
>
> 内部 SSH トンネルオプションを使用する場合は、ローカルホストではなく DocumentDB クラスターエンドポイントを確実に使用してください。



5. [サインイン] を選択します。

DbVisualizer から Amazon DocumentDB に接続する

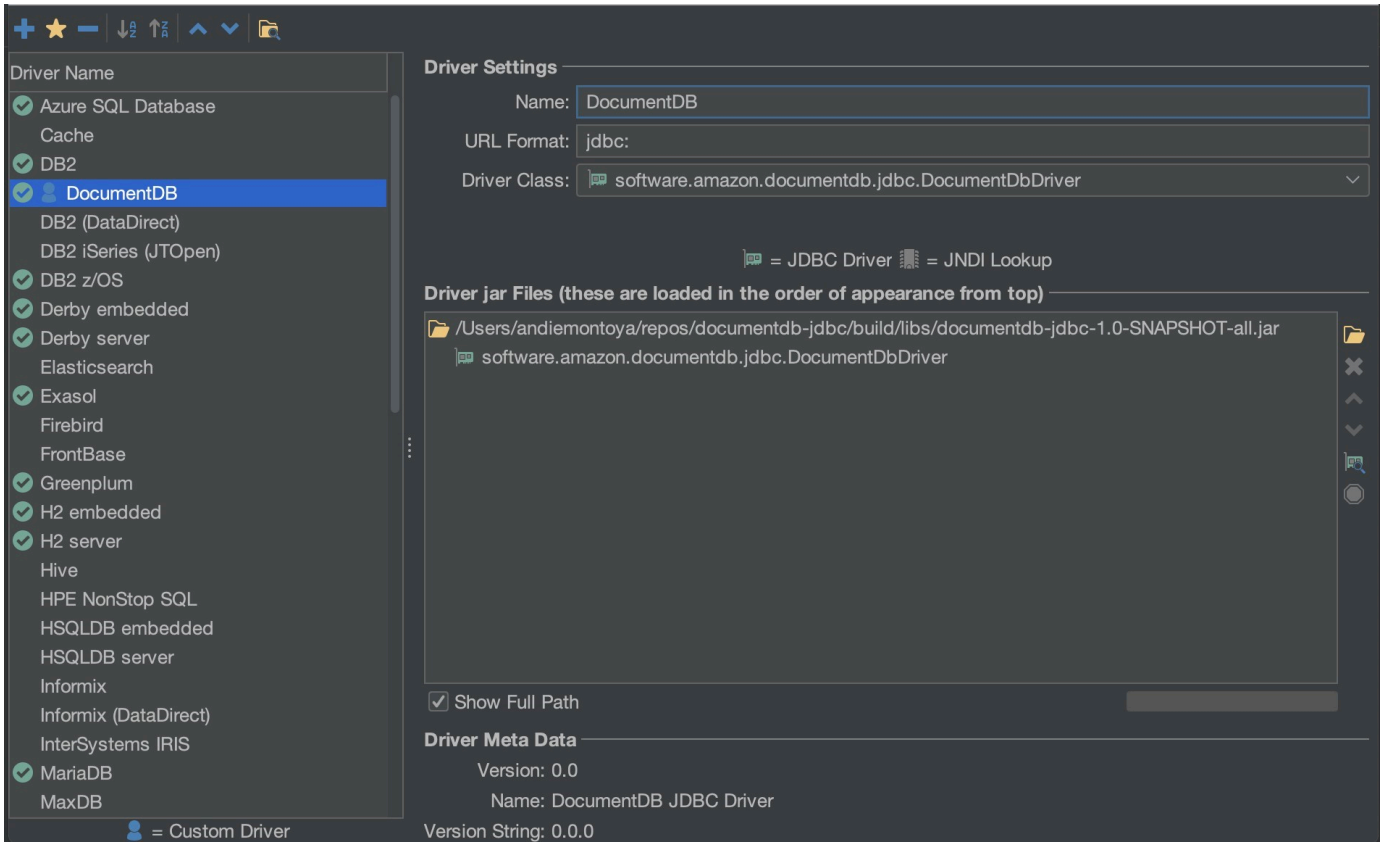
トピック

- Amazon DocumentDB JDBC ドライバーの追加
- DBVisualizer を使用して Amazon DocumentDB に接続する

Amazon DocumentDB JDBC ドライバーの追加

DbVisualizer から Amazon DocumentDB に接続するには、まずAmazon DocumentDB JDBC ドライバーをインポートする必要があります。

1. DbVisualizer アプリケーションをスタートし、メニューパスに移動します : ツール > ドライバーマネージャー...
2. + を選択します (またはメニューで、ドライバ ー> ドライバーを作成 を選択) 。
3. [Name] を [DocumentDB] に設定します。
4. URL 形式 を `jdbc:documentdb://<host>[:port]/<database>[?option=value[&option=value[...]]]` に設定
5. フォルダ ボタンを選択し、Amazon DocumentDB JDBC ドライバー JAR ファイルを選択し、オープン ボタンを選択します。
6. ドライバークラス フィールドが `software.amazon.documentdb.jdbc.DocumentDbDriver` になっていることを確認します。DocumentDB のドライバーマネージャの設定は、次の例のようになります。



7. ダイアログを閉じます。Amazon DocumentDB JDBC ドライバーがセットアップされ、使用できる状態になります。

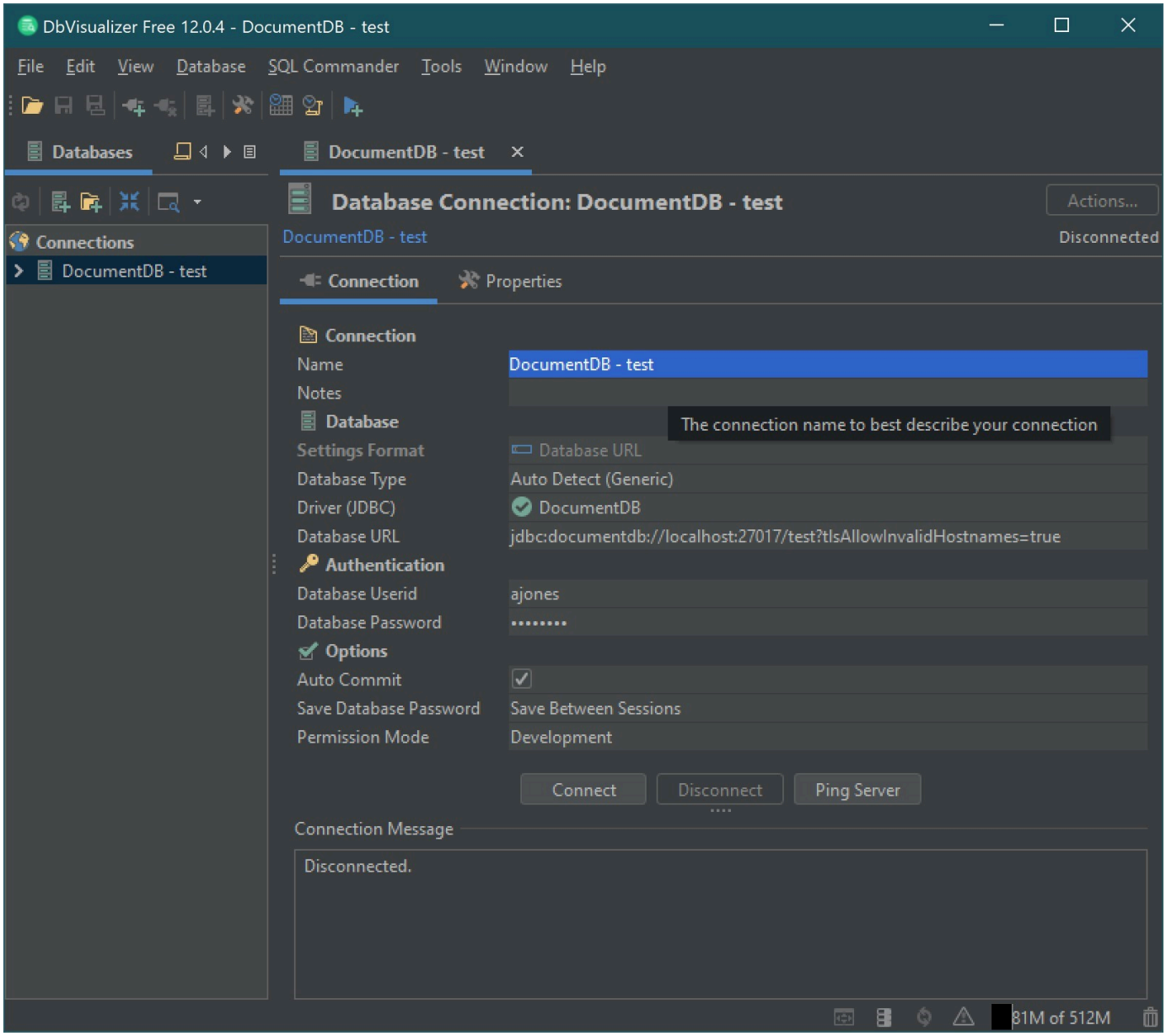
DBVisualizer を使用して Amazon DocumentDB に接続する

DBVisualizer を使用して Amazon DocumentDB に接続する

1. Amazon DocumentDB クラスターの VPC の外部から接続する場合は、SSH トンネルが設定されていることを確認してください。
2. トップレベルメニューから、データベース > データベース接続の作成 を選択します。
3. 名前 フィールドに、わかりやすい名前を入力します。
4. ドライバー (JDBC) を前のセクションで作成した DocumentDB ドライバーに設定します。
5. データベース URL を JDBC 接続文字列に設定します。

   例: `jdbc:documentdb://localhost:27017/database?tlsAllowInvalidHostnames=true`
6. データベースユーザー ID を Amazon DocumentDB ユーザー ID に設定します。
7. データベースパスワード を、ユーザー ID に対応するパスワードに設定します。

データベース接続ダイアログは、次のダイアログのようになります。



8. [接続]を選択してください。

JDBC スキーマの自動生成

Amazon DocumentDB はドキュメントデータベースであるため、テーブルとスキーマの概念はありません。ただし、Tableau などの BI ツールは、データベースがスキーマを提示することを期待します。具体的には、JDBC ドライバー接続でデータベース内のコレクションのスキーマを取得する必要がある場合は、データベース内のすべてのコレクションをポーリングします。ドライバーは、そのコレクションのスキーマのキャッシュバージョンがすでに存在するかどうかを判別します。キャッシュ

されたバージョンが存在しない場合は、ドキュメントのコレクションをサンプリングし、次の動作に基づいてスキーマを作成します。

トピック

- スキーマ生成の制限
- スキャン方法のオプション
- Amazon DocumentDB データ型
- スカラードキュメントフィールドのマッピング
- オブジェクトと配列のデータ型の処理

スキーマ生成の制限

DocumentDB JDBC ドライバーは識別子の長さに 128 文字の制限を課しています。スキーマジェネレータは、生成された識別子 (テーブル名とカラム名) の長さを切り捨てて、その制限に適合するようにします。

スキャン方法のオプション

サンプリング動作は、接続文字列またはデータソースオプションを使用して変更できます。

- scanMethod=<option>
  - random - (デフォルト) - サンプルドキュメントはランダムな順序で返されます。
  - idForward - サンプルドキュメントは ID の順序で返されます。
  - idReverse - サンプルドキュメントは id の逆の順序で返されます。
  - all - コレクション内のすべてのドキュメントをサンプリングします。
- ScanLimit=<n> - サンプリングするドキュメントの数。値は正の整数である必要があります。デフォルト値は 1000 です。scanMethod が all に設定されている場合、このオプションは無視されます。

Amazon DocumentDB データ型

Amazon DocumentDB サーバーはいくつかの MongoDB データ型をサポートしています。サポートされているデータ型とそれに関連付けられた JDBC データ型を以下に示します。

MongoDB データ型	DocumentDB でサポートされています	JDBC データ型
バイナリデータ	あり	VARBINARY
ブール値	あり	BOOLEAN
倍精度	あり	DOUBLE
32 ビット整数	あり	INTEGER
64 ビット整数	あり	BIGINT
String	あり	VARCHAR
ObjectId	あり	VARCHAR
日付	あり	TIMESTAMP
Null	あり	VARCHAR
正規表現	あり	VARCHAR
Timestamp	あり	VARCHAR
MinKey	あり	VARCHAR
MaxKey	あり	VARCHAR
オブジェクト	あり	仮想テーブル
配列	あり	仮想テーブル
Decimal128	いいえ	DECIMAL
JavaScript	いいえ	VARCHAR
JavaScript (スコープ付き)	いいえ	VARCHAR
未定義	いいえ	VARCHAR
記号	いいえ	VARCHAR

MongoDB データ型	DocumentDB でサポートされています	JDBC データ型
dbPointer (4.0+)	いいえ	VARCHAR

スカラードキュメントフィールドのマッピング

コレクションからドキュメントのサンプルをスキャンする場合、JDBC ドライバーは、コレクション内のサンプルを表す 1 つ以上のスキーマを作成します。一般に、ドキュメントのスカラーフィールドは、テーブルスキーマの列にマップされます。例えば、team というコレクションと 1 つのドキュメント `{ "_id" : "112233", "name" : "Alastair", "age": 25 }` の場合、これはスキーマにマップされます。

テーブル名	Column Name (列名)	データタイプ	キー
team	チームID	VARCHAR	PK
team	名前	VARCHAR	
team	年齢	INTEGER	

データ型の競合プロモーション

サンプリングされたドキュメントをスキャンするときに、フィールドのデータ型がドキュメント間で一貫していない可能性があります。この場合、JDBC ドライバーは JDBC データ型を、サンプリングされたドキュメントのすべてのデータ型に適合する共通データ型に昇格します。

例:

```
{
"_id" : "112233",
"name" : "Alastair", "age" : 25
}

{
"_id" : "112244",
"name" : "Benjamin",
"age" : "32"
```

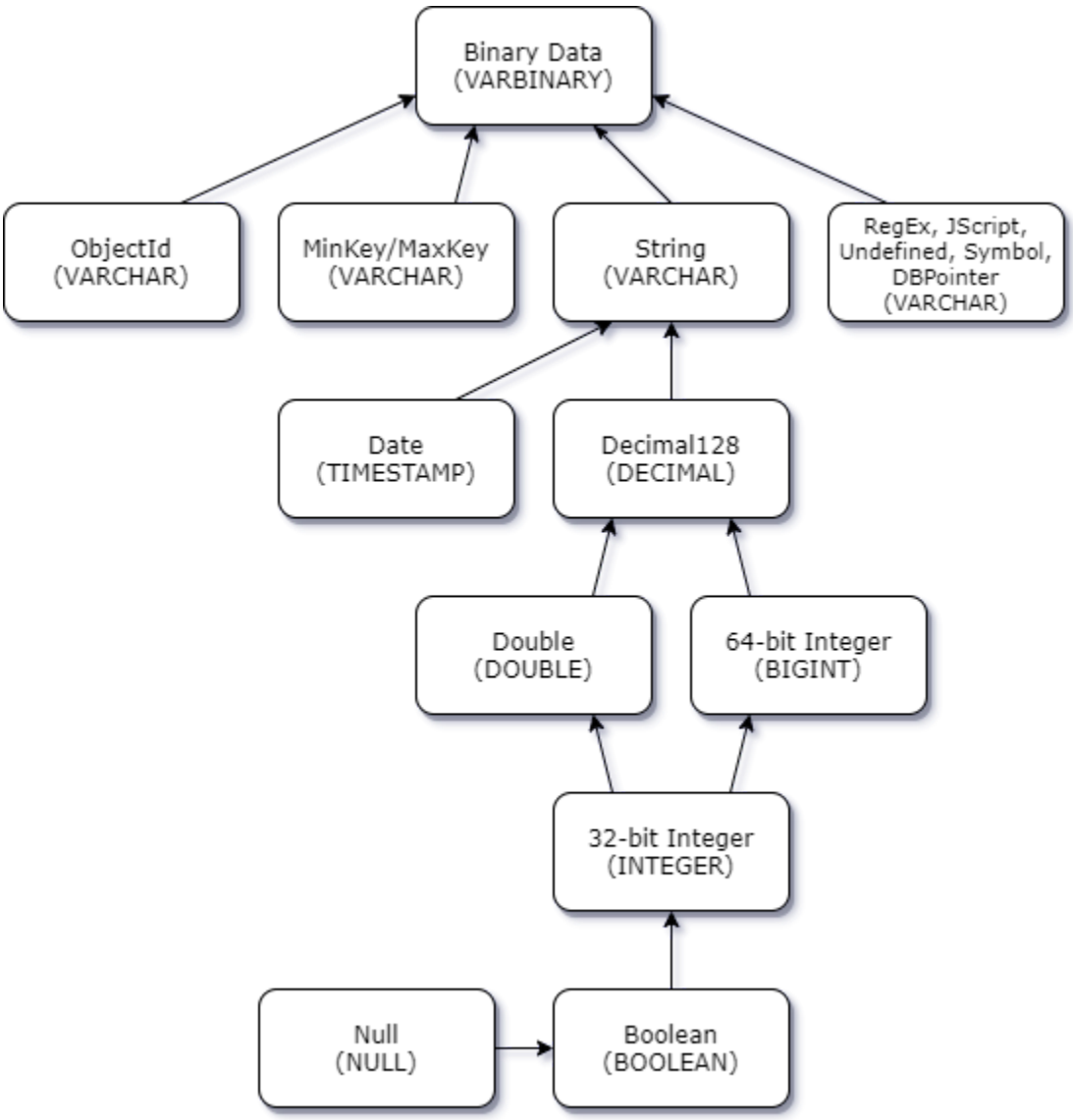
```
}
```

年齢 フィールドは、最初のドキュメントでは 32 ビット整数ですが、2 番目のドキュメントでは文字列です。ここで、JDBC ドライバーは JDBC データ型を VARCHAR に昇格させ、いずれかのデータ型を検出したときに処理します。

テーブル名	Column Name (列名)	データタイプ	キー
team	チームID	VARCHAR	PK
team	名前	VARCHAR	
team	年齢	VARCHAR	

スカラー - スカラー競合プロモーション

次の図表は、スカラー - スカラーデータ型の競合を解決する方法を示しています。



スカラー - 複合型競合プロモーション

スカラー· スカラー型の競合と同様に、異なるドキュメント内の同じフィールドは、複素数 (配列とオブジェクト) とスカラー (整数、ブールなど) の間で競合するデータ型を持つことができます。これらの競合はすべて、それらのフィールドについて VARCHAR に解決 (昇格) されます。この場合、配列とオブジェクトのデータは JSON 表現として返されます。

埋め込み配列 - 文字列フィールドの競合の例 :

```
{
   "_id":"112233",
   "name":"George Jackson",
   "subscriptions":[
```

```
      "Vogue",
      "People",
      "USA Today"
   ]
}
{
   "_id":"112244",
   "name":"Joan Starr",
   "subscriptions":1
}
```

上記の例は、customer2 テーブルのスキーマにマッピングされています。

テーブル名	Column Name (列名)	データタイプ	キー
customer2	Customer2 ID	VARCHAR	PK
customer2	名前	VARCHAR	
customer2	サブスクリプション	VARCHAR	

customer1_subscriptions 仮想テーブルは次のとおりです。

テーブル名	Column Name (列名)	データタイプ	キー
customer1_subscriptions	customer1 ID	VARCHAR	PK/FK
customer1_subscriptions	subscriptions_index_lvl0	BIGINT	PK
customer1_subscriptions	値	VARCHAR	
customer_address	city	VARCHAR	
customer_address	region	VARCHAR	
customer_address	country	VARCHAR	

テーブル名	Column Name (列名)	データタイプ	キー
customer_address	コード	VARCHAR	

オブジェクトと配列のデータ型の処理

これまでは、スカラーデータ型のマッピング方法のみについて説明しました。オブジェクトデータ型と配列データ型は (現在) 仮想テーブルにマッピングされています。JDBC ドライバーは、ドキュメント内のオブジェクトまたは配列フィールドを表す仮想テーブルを作成します。マッピングされた仮想テーブルの名前は、元のコレクションの名前の後にフィールドの名前をアンダースコア (「_」) で区切って連結します。

ベーステーブルの主要なキー (「_id」) は、新しい仮想テーブル内の新しい名前を取得し、関連付けられたベーステーブルの外部キーとして提供されます。

埋め込み配列型フィールドの場合、配列の各レベルの配列へのインデックスを表すインデックス列が生成されます。

埋め込みオブジェクトフィールドの例

ドキュメント内のオブジェクトフィールドの場合、仮想テーブルへのマッピングは JDBC ドライバーによって作成されます。

```
{
   "Collection: customer",
   "_id":"112233",
   "name":"George Jackson",
   "address":{
      "address1":"123 Avenue Way",
      "address2":"Apt. 5",
      "city":"Hollywood",
      "region":"California",
      "country":"USA",
      "code":"90210"
   }
}
```

上記の例は、customer テーブルのスキーマにマッピングされます。

テーブル名	Column Name (列名)	データタイプ	キー
カスタマー	カスタマー ID	VARCHAR	PK
カスタマー	名前	VARCHAR	

および顧客アドレス仮想テーブルは次のとおりです。

テーブル名	Column Name (列名)	データタイプ	キー
customer_address	カスタマー ID	VARCHAR	PK/FK
customer_address	address1	VARCHAR	
customer_address	address2	VARCHAR	
customer_address	city	VARCHAR	
customer_address	region	VARCHAR	
customer_address	country	VARCHAR	
customer_address	コード	VARCHAR	

埋め込み配列フィールドの例

ドキュメント内の配列フィールドの場合、JDBC ドライバーによって仮想テーブルへのマッピングも作成されます。

```
{
   "Collection: customer1",
   "_id":"112233",
   "name":"George Jackson",
   "subscriptions":[
      "Vogue",
      "People",
      "USA Today"
   ]
}
```

上記の例は、customer1 テーブルのスキーマにマッピングします。

テーブル名	Column Name (列名)	データタイプ	キー
customer1	customer1 ID	VARCHAR	PK
customer1	名前	VARCHAR	

customer1_subscriptions 仮想テーブルは次のとおりです。

テーブル名	Column Name (列名)	データタイプ	キー
customer1_subscriptions	customer1 ID	VARCHAR	PK/FK
customer1_subscriptions	subscriptions_index_lvl0	BIGINT	PK
customer1_subscriptions	値	VARCHAR	
customer_address	city	VARCHAR	
customer_address	region	VARCHAR	
customer_address	country	VARCHAR	
customer_address	コード	VARCHAR	

SQL サポートと制限

Amazon DocumentDB JDBC ドライバーは、SQL-92 のサブセットといくつかの一般的な拡張を Support (サポート)する読み取り専用ドライバーです。詳細については、SQL の制限のドキュメントと JDBC の制限のドキュメント を参照してください。

トラブルシューティング

Amazon DocumentDB JDBC ドライバーの使用に問題がある場合は、トラブルシューティングガイド を参照してください。

Amazon DocumentDB JDBC ドライバーを使用して接続する

Amazon DocumentDB 用の ODBC ドライバーは、デベロッパーのための SQL リレーショナルインターフェイスを提供し、Power BI Desktop や Microsoft Excel などの BI ツールからの接続を可能にします。

詳細については、GitHub 上の Amazon DocumentDB JDBC ドライバーのドキュメントを参照してください。

トピック

- 入門
- Windows での Amazon DocumentDB ODBC ドライバーのセットアップ
- Microsoft Excel から Amazon DocumentDB に接続
- Microsoft Power BI Desktop から Amazon DocumentDB に接続します
- 自動スキーマ生成
- SQL サポートと制限
- トラブルシューティング

入門

ステップ 1. Amazon DocumentDB クラスターの作成

Amazon DocumentDB クラスターをまだお持ちでない場合、始める方法はいくつかあります。

> **Note**
>
> Amazon DocumentDB は仮想プライベートクラウド (VPC) 専用サービスです。クラスターの VPC 外にあるローカルマシンから接続する場合は、Amazon EC2 インスタンスへの SSH 接続を作成する必要があります。この場合、EC2 を使用して接続 の手順に従ってクラスターを起動します。SSH トンネリングおよび必要な場合の詳細については、「SSH トンネルを使って Amazon DocumentDB に接続する」を参照してください。

ステップ 2. JRE または JDK インストール

BI アプリケーションによっては、64 ビット JRE または JDK インストールバージョン 8 以降がコンピュータにインストールされていることを確認する必要がある場合があります。Java SE ランタイム環境 8 をダウンロードは こちら。

ステップ 3 Amazon DocumentDB ODBC ドライバーのダウンロード

Amazon DocumentDB ODBC ドライバーはこちらからダウンロードしてください。適切なインストーラー (たとえば documentdb-odbc-1.0.0.msi) を選択します。インストールガイドに従います。

ステップ 4。SSH トンネルを使用して Amazon DocumentDB に接続する

Amazon DocumentDB クラスターは、Amazon Virtual Private Cloud (Amazon VPC) 内にデプロイされます。これらは、Amazon EC2 インスタンス、または同じ Amazon VPC にデプロイされている他の AWS サービスから直接アクセスできます。さらに、Amazon DocumentDB には、同じ AWS リージョンまたは他のリージョンの異なる VPCs の Amazon EC2 インスタンスまたは他の AWS サービスが VPC ピアリングを介してアクセスできます。

ただし、お客様のユースケースで、お客様 (またはお客様のアプリケーション) が、クラスターの VPC の外部から Amazon DocumentDB リソースにアクセスする必要があるとします。これは、ほとんどのユーザーが Amazon DocumentDB クラスターと同じ VPC 内の仮想マシンでアプリケーションを実行していない場合です。VPC 外から接続する場合、SSH トンネリング (ポート転送とも呼ばれる) を使用して Amazon DocumentDB リソースにアクセスできます。

SSH トンネルを作成するには、Amazon DocumentDB クラスターと同じ Amazon VPC で実行中の Amazon EC2 インスタンスが必要です。クラスターと同じ VPC で既存の EC2 インスタンスを使用するか、新しい EC2 インスタンスを作成できます。その場合、ローカルコンピュータで以下のコマンドを実行することにより Amazon DocumentDB クラスター `sample-cluster.node.us-east-1.docdb.amazonaws.com` に SSH トンネルを設定できます。

```
ssh -i "ec2Access.pem" -L 27017:sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017 ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com -
N
```

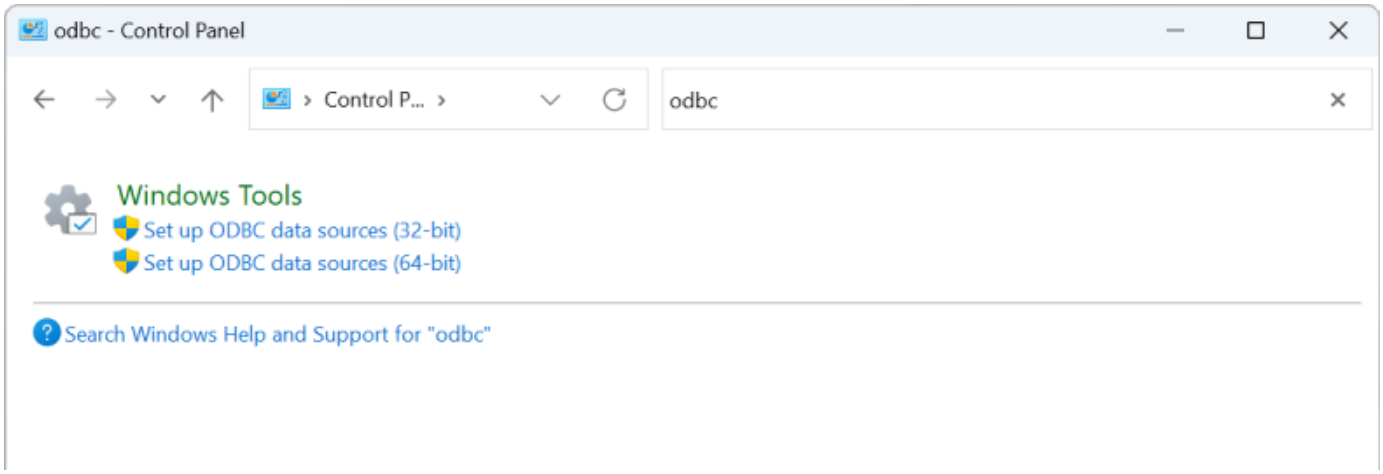
ローカルポートの転送を行う場合は、-L フラグを使用します。これは、VPC 外のクライアントで実行されている BI ツールに接続するための前提条件です。上記のステップを実行したら、選択した BI ツールの次のステップに進むことができます。

SSH トンネリングの詳細については、「SSH トンネルを使用して Amazon DocumentDB に接続する」ドキュメントを参照してください。

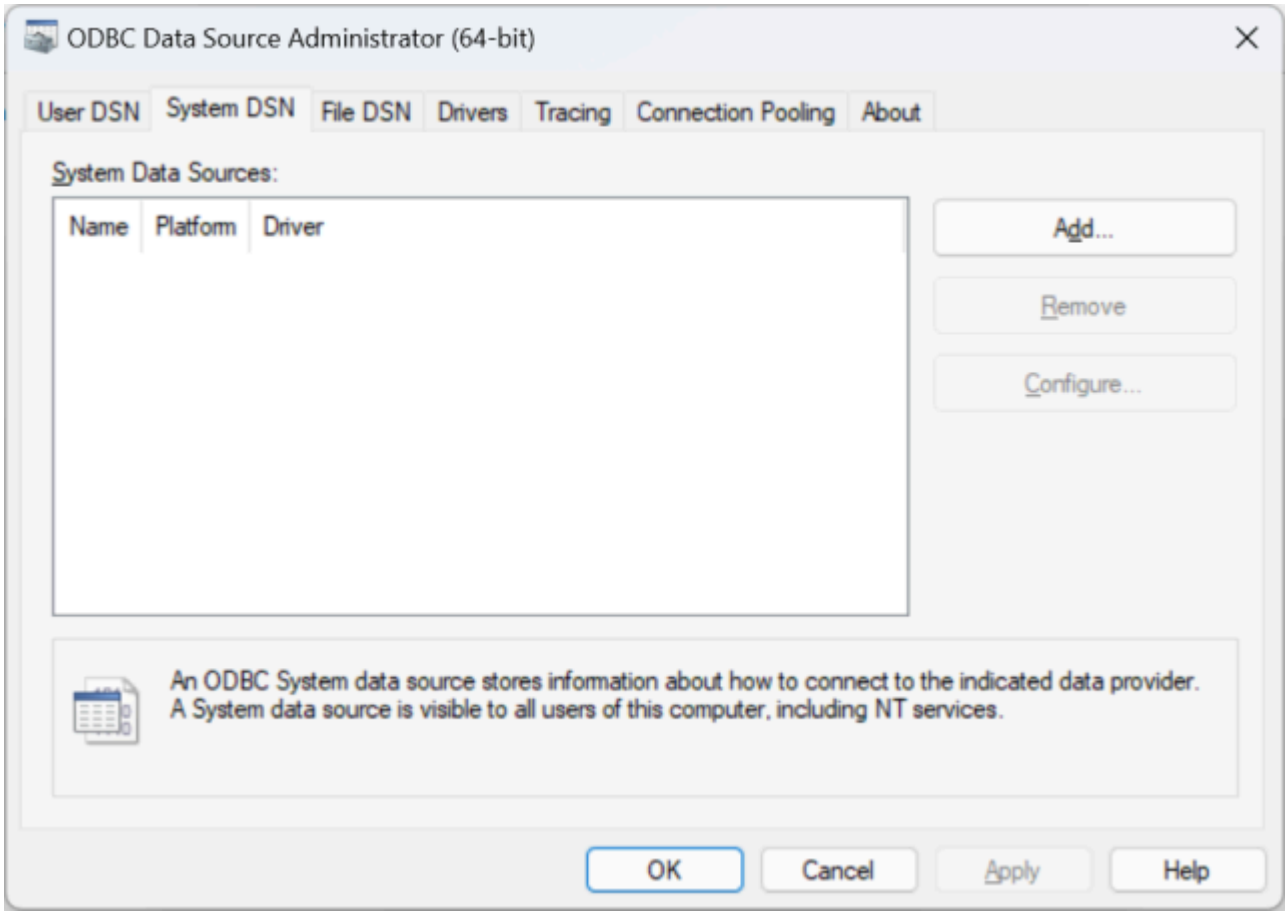
Windows での Amazon DocumentDB ODBC ドライバーのセットアップ

Windows で Amazon DocumentDB ODBC ドライバーを設定するには、次の手順に従います。

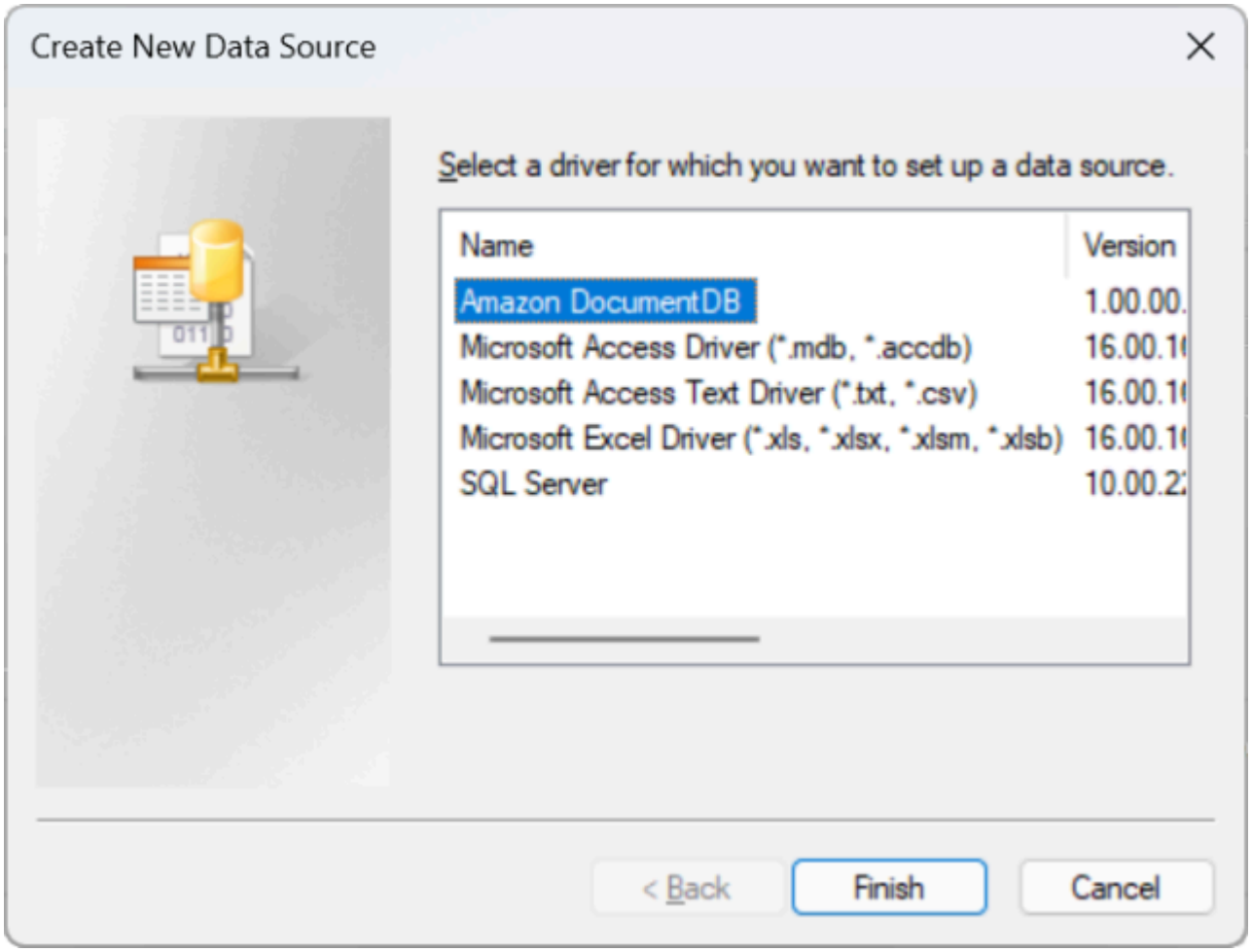
1. Windows で [コントロールパネル] を開き、ODBC を検索します (またはメニューで [Windows ツール] > [ODBC データソース (32 ビット)] または [ODBC データソース (64 ビット)] を選択)。



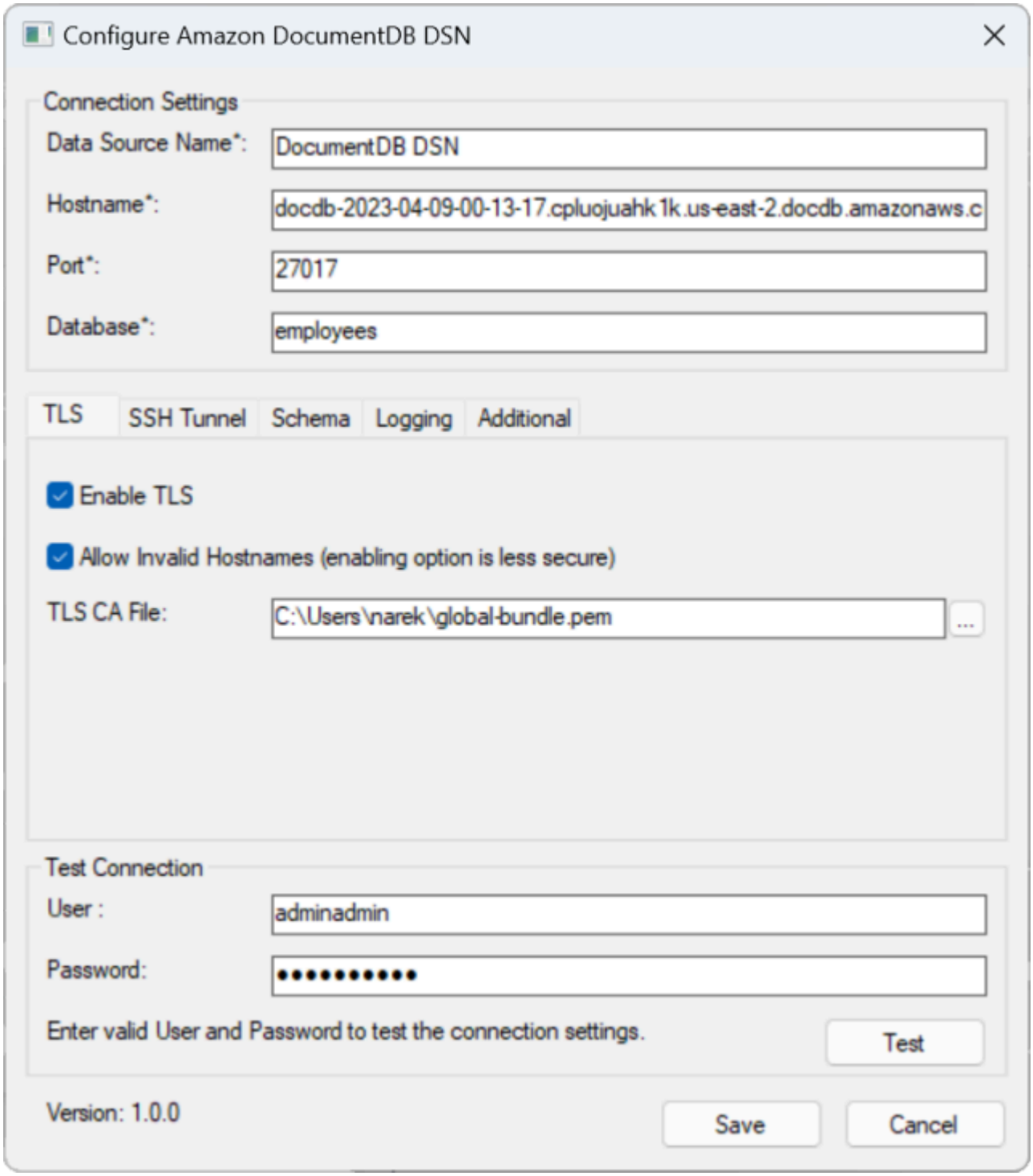
2. 適切な ODBC ドライバデータソース管理者を選択します。インストールされていれば 32 ビットバージョンを、インストールされていない場合は 64 ビットバージョンを選択します。
3. [システム DSN] タブを選択し、[追加...] をクリックして新しいDSNを追加します。



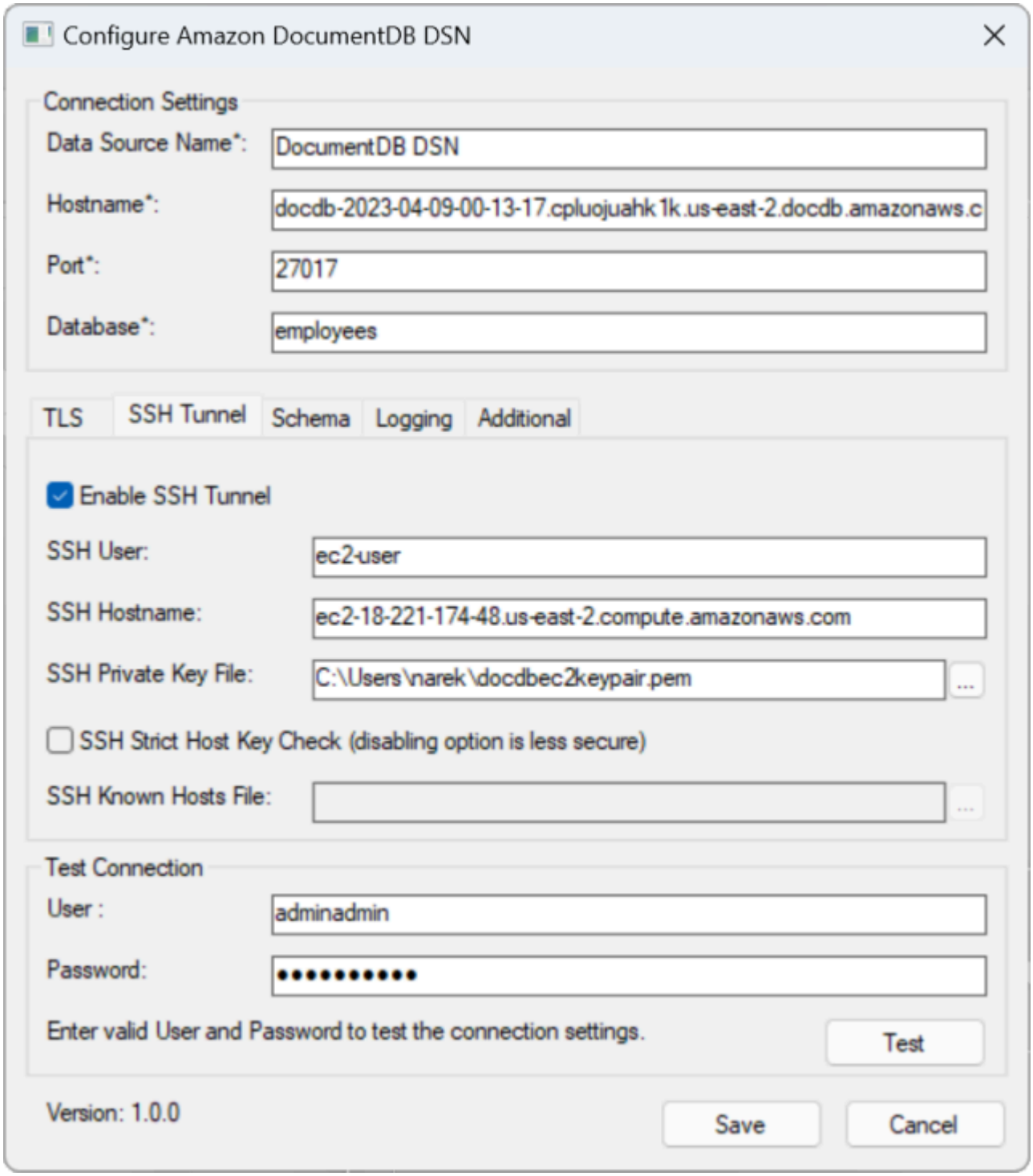
4. データソースドライバーリストから [Amazon DocumentDB] を選択します。



5. Amazon DocumentDB DSN の設定ダイアログで、接続設定、TLS タブ、および接続のテストフィールドに入力し、保存をクリックします。



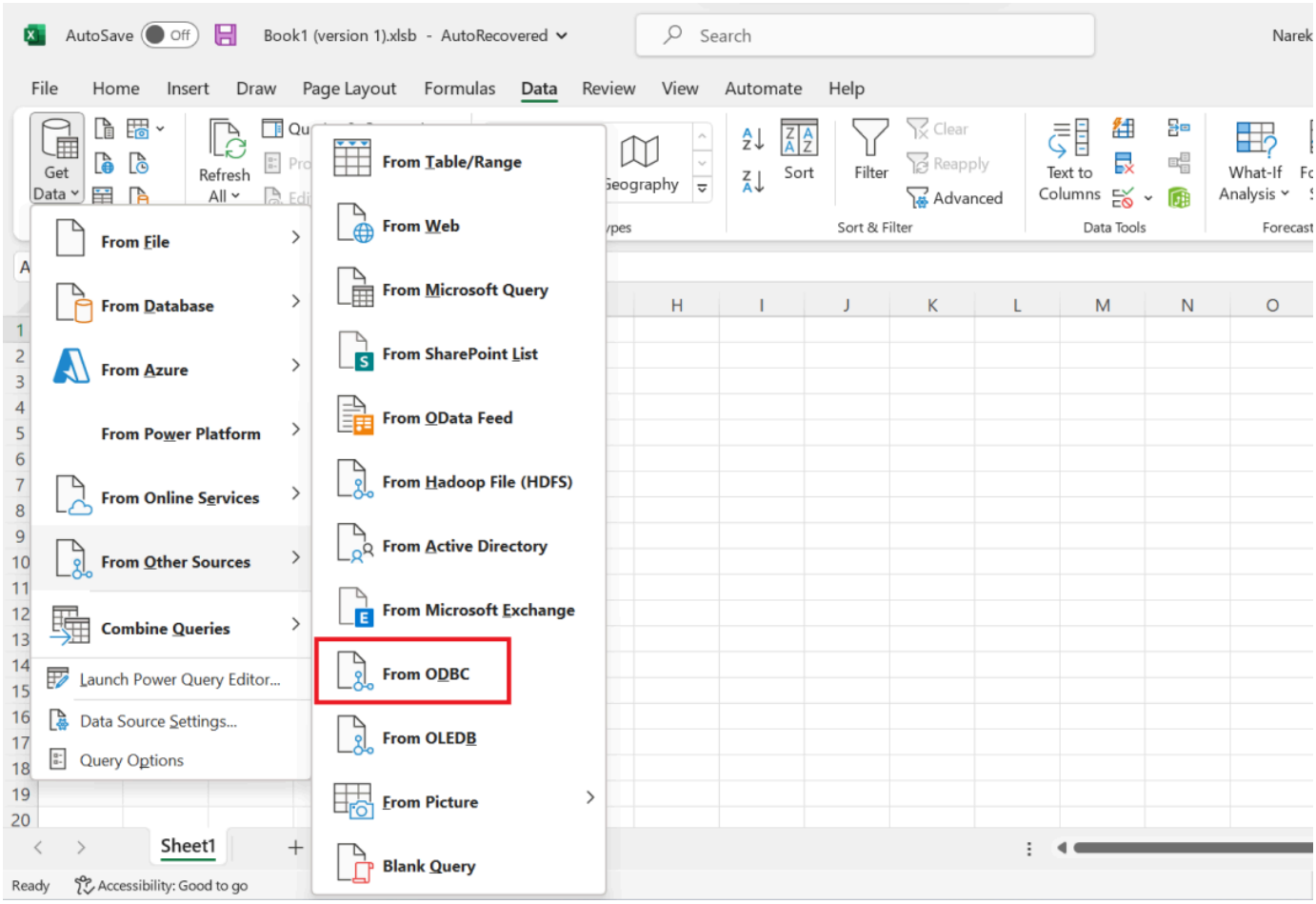
6. 選択した EC2 インスタンスへの SSH トンネリング方法によって接続の詳細が異なるため、Windows フォームには正確に記入してください。SSH トンネリング方法についてはこちらを参照してください。各プロパティの詳細については、「接続文字列の構文とオプション」を参照してください。



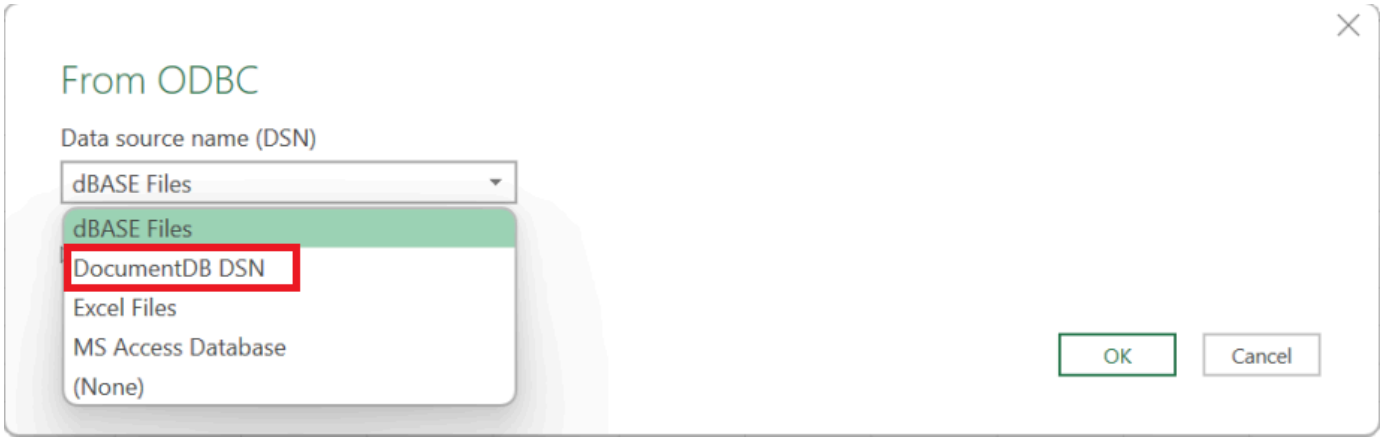
Windows での Amazon DocumentDB ODBC ドライバーの設定の詳細については、こちらを参照してください。

Microsoft Excel から Amazon DocumentDB に接続

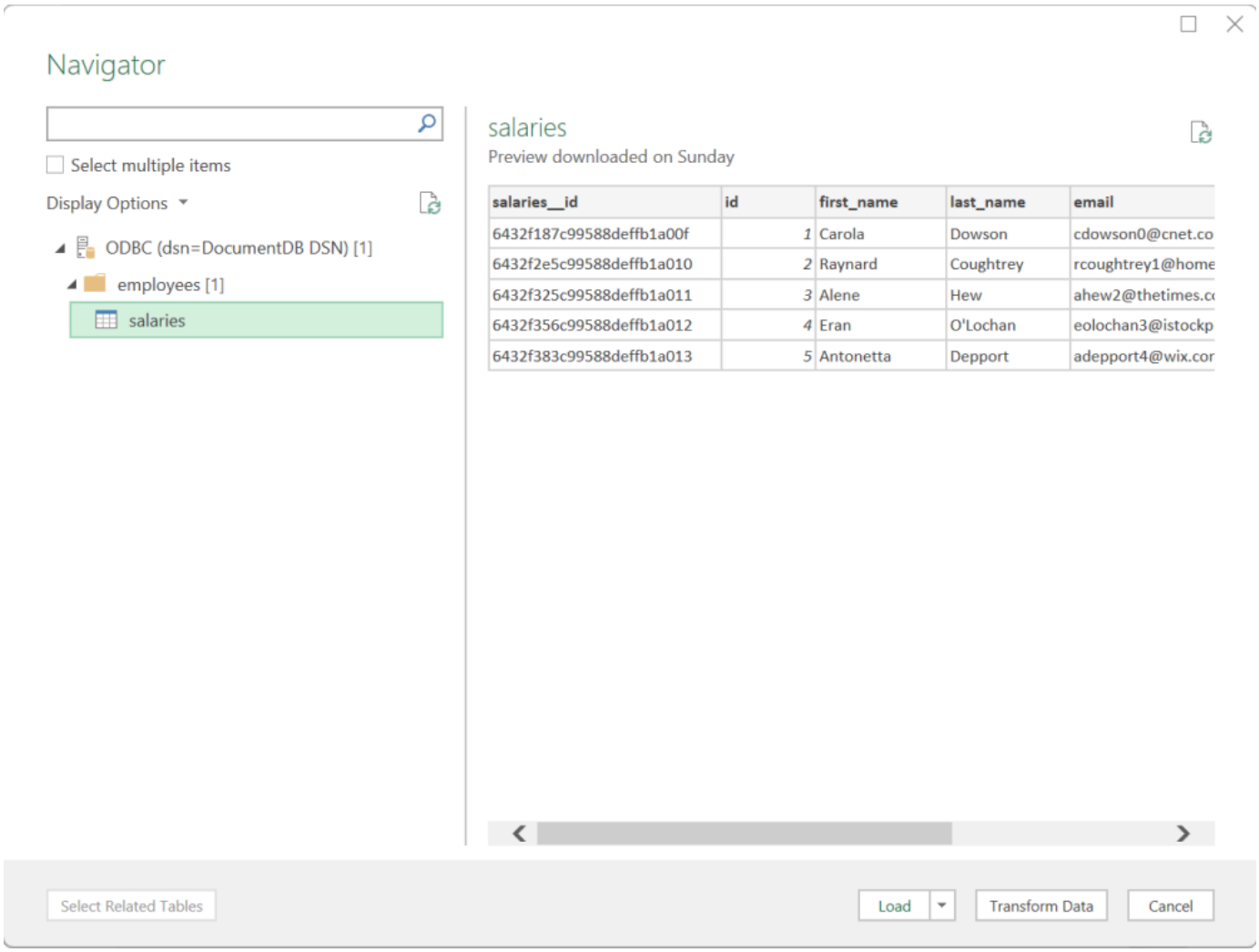
1. Amazon DocumentDB ドライバーが正しくインストールされ、設定されていることを確認します。詳細については、「Windows での ODBC ドライバーのセットアップ」を参照してください。
2. Microsoft Excel を起動します。
3. [データ] > [データの取得] > [他のソースから] に移動します。
4. [ODBC から] を選択します。



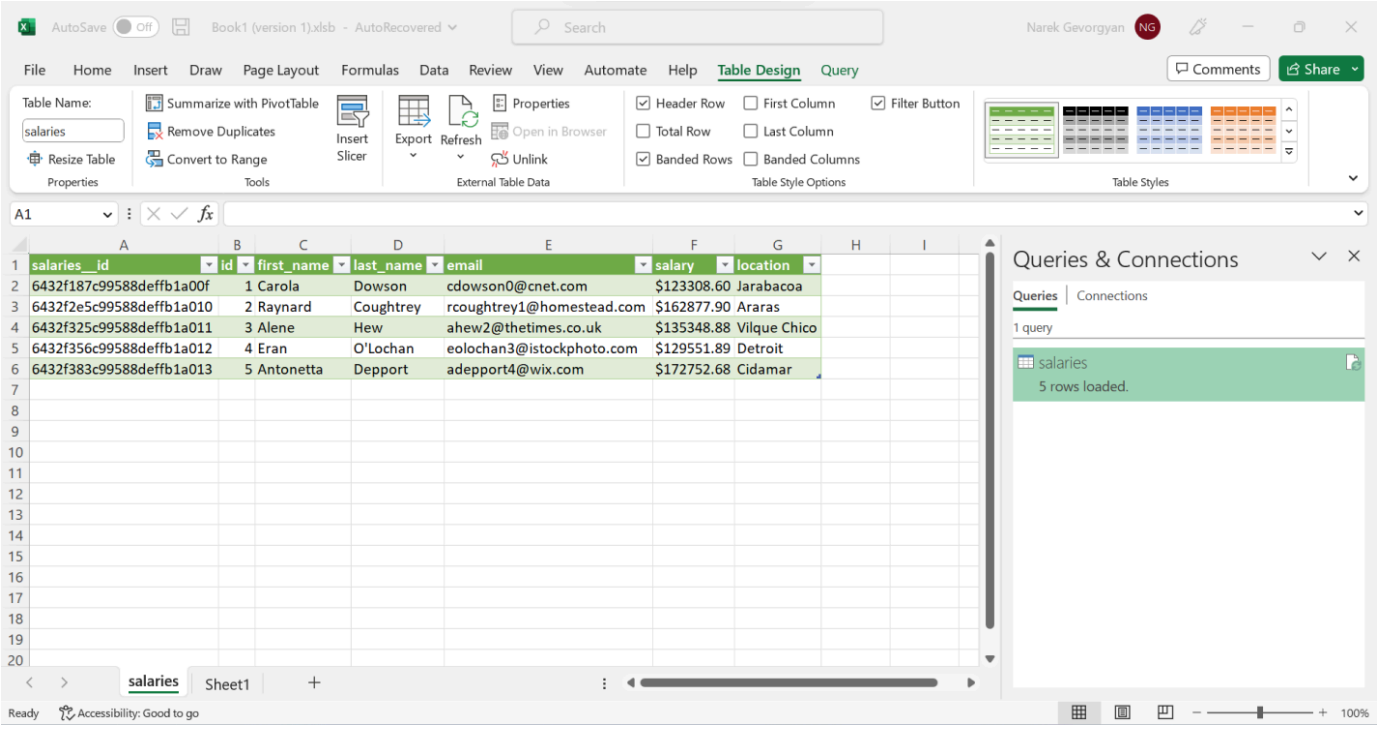
5. Amazon DocumentDB に関連付けられているデータソース名 (DSN) ドロップダウンメニューから、データソースを選択します。



6. Excel にデータをロードするコレクションを選択します。



7. Excel にデータをロードします。



Microsoft Power BI Desktop から Amazon DocumentDB に接続します

トピック

- 前提条件
- Microsoft Power BI Desktop カスタムコネクタの追加
- Amazon DocumentDB カスタムコネクタを使用して接続します
- Microsoft Power BI ゲートウェイの設定

前提条件

開始する前に、Amazon DocumentDB ODBC ドライバーが正しくインストールされていることを確認してください。

Microsoft Power BI Desktop カスタムコネクタの追加

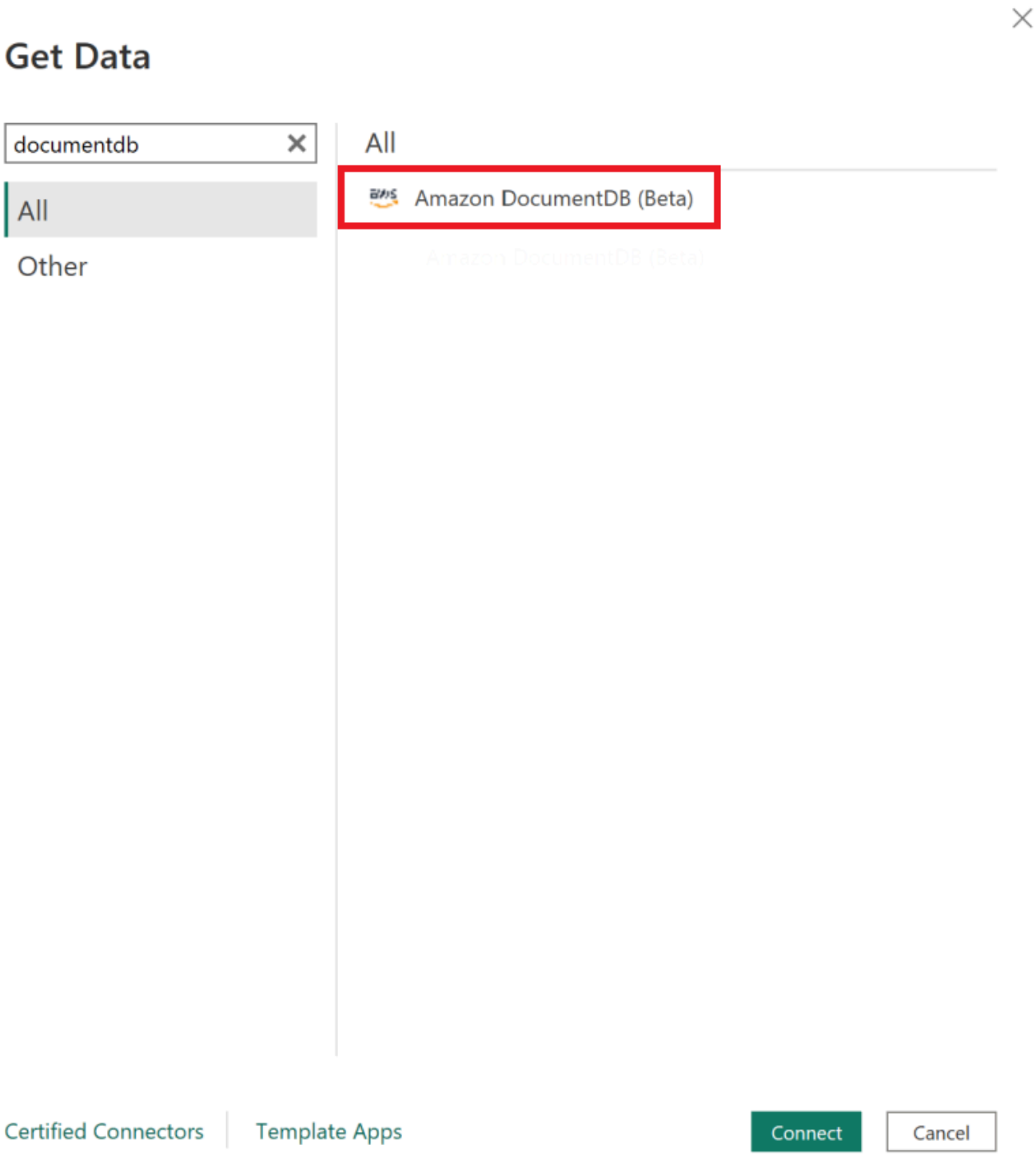
`AmazonDocumentDBConnector.mez` ファイルを `<User>\Documents\Power BI Desktop \Custom Connectors\` フォルダー (または OneDrive を使用している場合は `<User>\OneDrive \Documents\Power BI Desktop\Custom Connectors`) にコピーします。これにより、Power BI はカスタムコネクタにアクセスできるようになります。Power BI Desktop へのコネクタはこち

らから入手できます。Power BI Desktop を再起動して、コネクタがロードされていることを確認します。

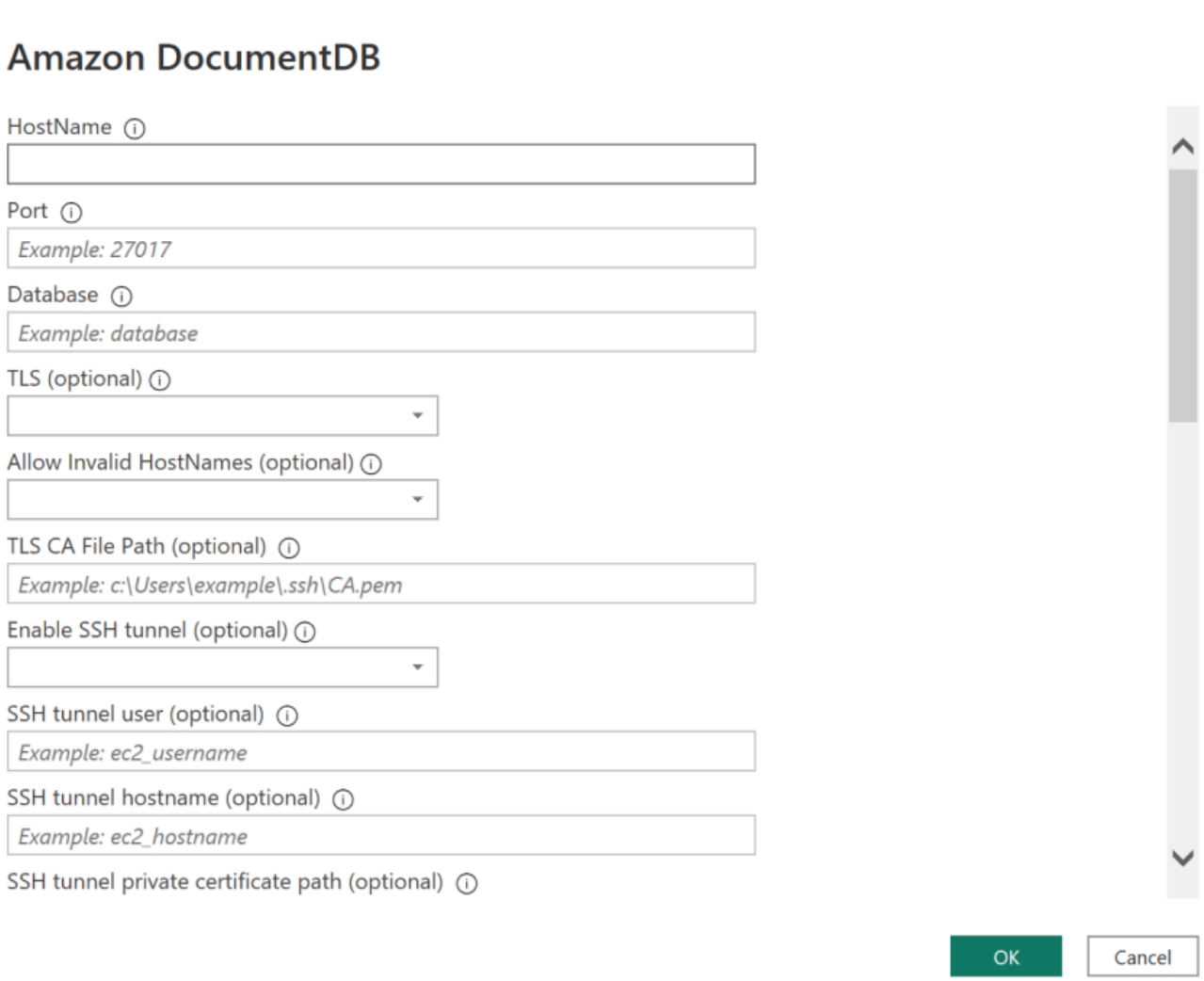
> **Note**
>
> カスタムコネクタは、認証用に Amazon DocumentDB のユーザー名とパスワードのみをサポートします。

Amazon DocumentDB カスタムコネクタを使用して接続します

1. [データを取得] から [Amazon DocumentDB (ベータ)] を選択し、[接続] をクリックします。サードパーティサービスの使用について警告が表示された場合は、[続行] をクリックします。



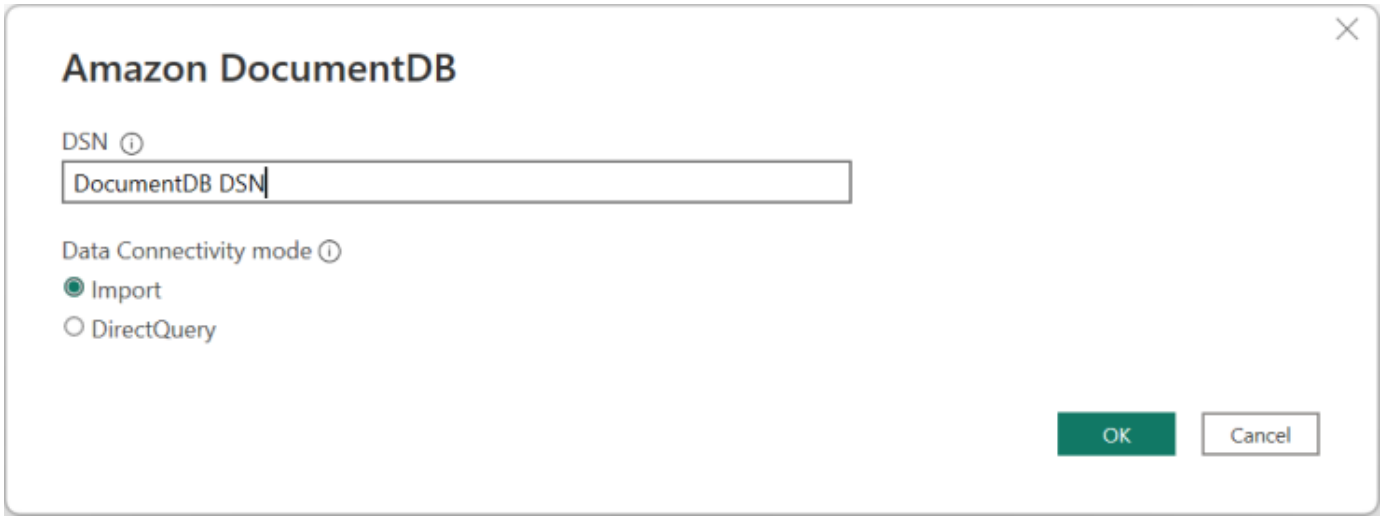
2. Amazon DocumentDB クラスターに接続するために必要な情報をすべて入力し、[OK] をクリックします。



> **Note**
>
> ODBC ドライバーのデータソース名 (DSN) の設定によっては、DSN 設定で必要な情報をすでに提供している場合、SSH 接続の詳細画面が表示されない場合があります。

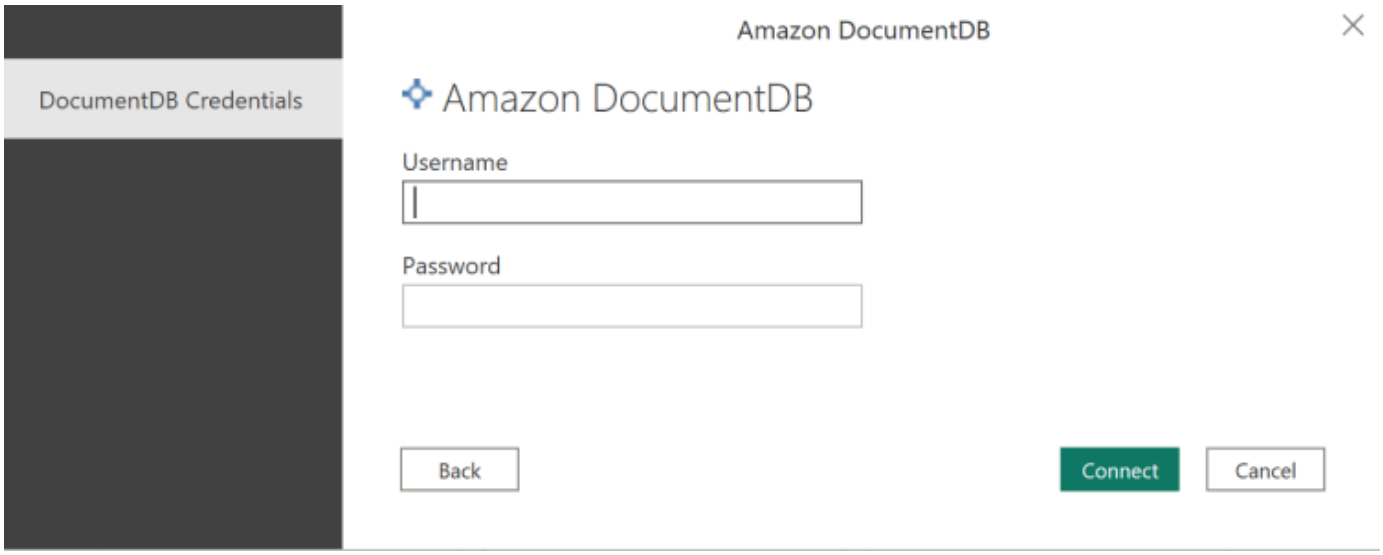
3. データ接続モードを選択します。

   - インポート: すべてのデータをロードし、情報をディスクに保存します。データの更新を表示するには、データを更新して再ロードする必要があります。
   - ダイレクトクエリ: データをロードしませんが、データに対してライブクエリを実行します。つまり、データの更新を表示するためにデータを更新したり再ロードしたりする必要はありません。

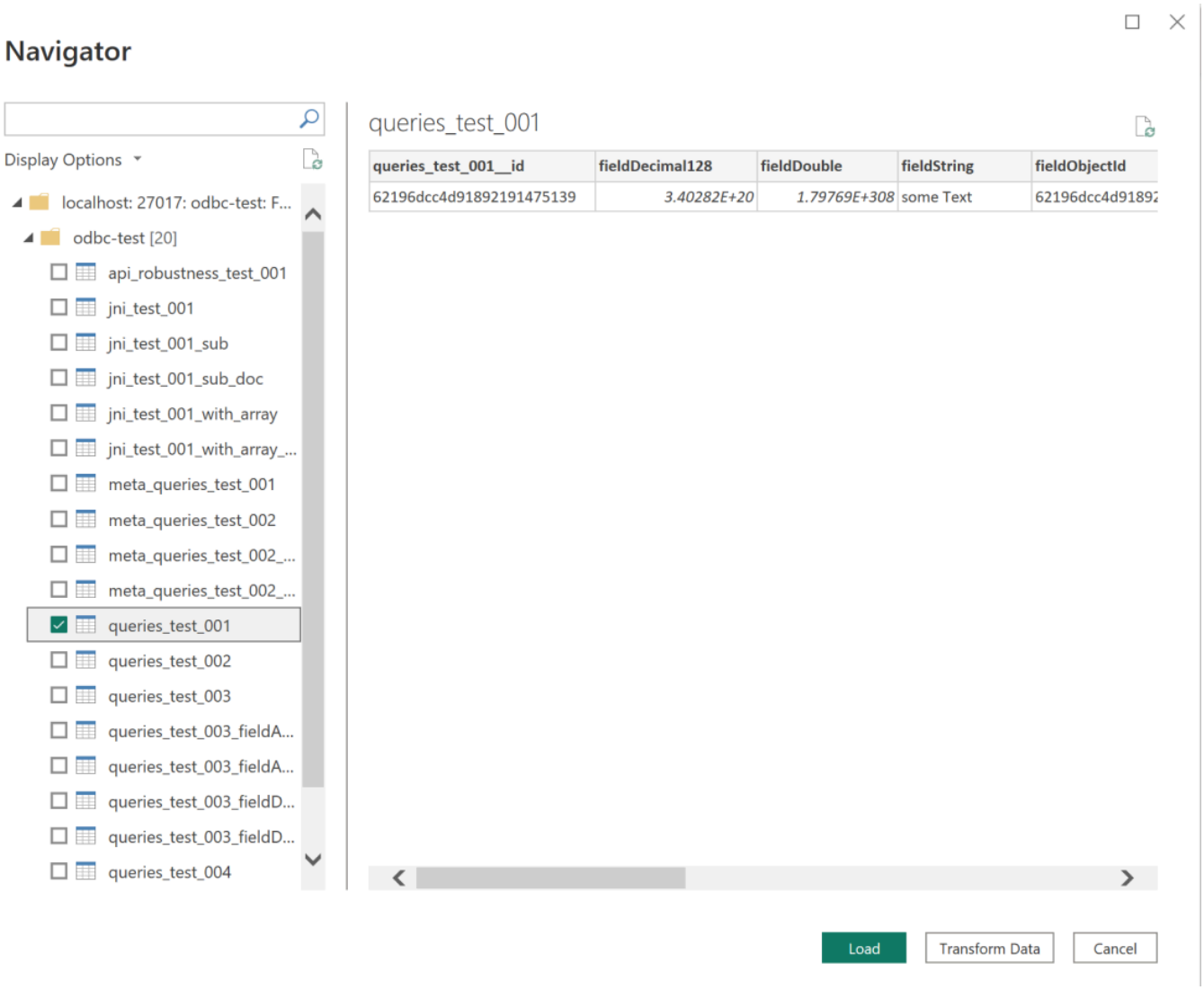


> **Note**
>
> 非常に大きなデータセットを使用している場合は、すべてのデータをインポートするのに時間がかかることがあります。

4. このデータソースに初めて接続する場合は、認証タイプを選択し、プロンプトが表示されたら認証情報を入力します。次に、[接続] をクリックします。



5. [ナビゲーター] ダイアログで、必要なデータベース表を選択し、[ロード] をクリックしてデータを読み込むか、[データ変換] をクリックしてデータの変換を続行します。



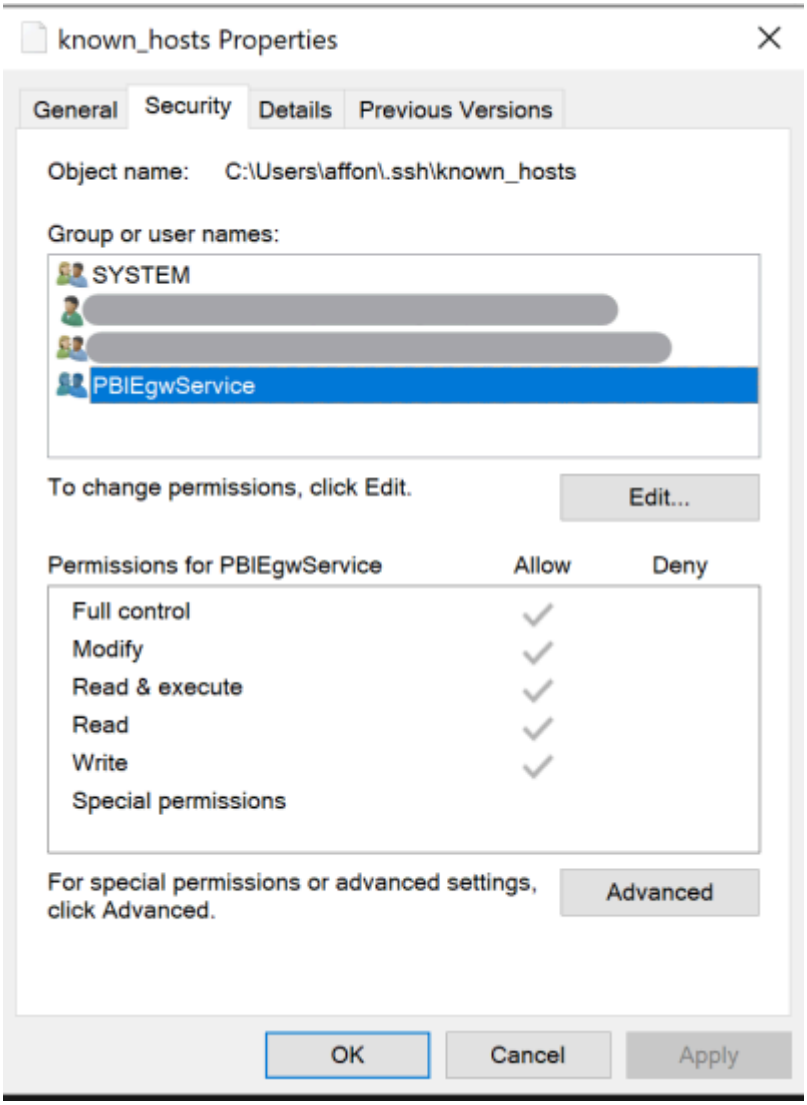
> **Note**
>
> 接続すると、データソースの設定が保存されます。設定を変更するには、[データ変換] > [データソース設定] を選択します。

Microsoft Power BI ゲートウェイの設定

前提条件:

- カスタムコネクタが Power BI ゲートウェイで動作することを確認します。
- Power BI ゲートウェイがインストールされているマシンの [システム] タブにある ODBC データソースに ODBC DSN が作成されていることを確認します。

内部 SSH トンネル機能を使用している場合は、Power BI サービスアカウントがアクセスできる場所に known_hosts ファイルを配置する必要があります。



> **Note**
>
> これは、認証機関 (CA) 証明書ファイル (pem ファイル) など、Amazon DocumentDB クラスターへの接続を確立するために必要なすべてのファイルにも当てはまります。

自動スキーマ生成

ODBC ドライバーは JNI (Java ネイティブインターフェイス) 経由で Amazon DocumentDB JDBC ドライバーを利用しているため、自動スキーマ生成機能は JDBC ドライバーでも同様に機能しま

す。スキーマの自動生成の詳細については、「JDBC 自動スキーマ生成」を参照してください。また、ODBC ドライバーのアーキテクチャーについて詳しくは、こちらをご覧ください。

SQL サポートと制限

Amazon DocumentDB ODBC ドライバーは、SQL-92 のサブセットといくつかの一般的な拡張を Support (サポート) する読み取り専用ドライバーです。詳細については、ODBC のサポートと制限に関するドキュメントを参照してください。

トラブルシューティング

Amazon DocumentDB JDBC ドライバーの使用に問題がある場合は、「トラブルシューティングガイド 」を参照してください。

Amazon DocumentDB を使用したプログラム

このサービスは JSON スキーマ検証をサポートしています。

トピック

- Amazon DocumentDB Java プログラミングガイド
- JSON スキーマ検証の使用

Amazon DocumentDB Java プログラミングガイド

この包括的なガイドでは、MongoDB の Java ドライバーを使用して Amazon DocumentDB を操作するための詳細なウォークスルーを提供し、データベースの運用と管理の重要な側面について説明します。

トピック

- 序章
- 前提条件
- このガイドのデータモデル
- MongoDB Java ドライバーを使用した Amazon DocumentDB への接続 MongoDB
- Java を使用した Amazon DocumentDB での CRUD オペレーションの実行
- Java を使用した Amazon DocumentDB でのインデックス管理
- Amazon DocumentDB と Java を使用したイベント駆動型プログラミング

序章

このガイドでは、MongoDB Java ドライバーを使用して DocumentDB クラスターへの安全な接続を確立する方法について説明します。接続文字列コンポーネント、SSL/TLS 実装、IAM 認証や接続プーリングなどのさまざまな接続オプション、堅牢なエラー処理戦略について詳しく説明します。

CRUD (作成、読み取り、更新、削除) オペレーションセクションで、このガイドではドキュメントの操作について詳しく説明し、単一オペレーションと一括オペレーションの両方を使用してドキュメントを作成、読み取り、更新、削除する方法を示します。ここでは、フィルター、クエリ、およびさまざまなオペレーションオプションの使用方法について説明しながら、エラー処理のベストプラクティスと信頼性を向上させるための再試行ロジックの実装に重点を置いています。このガイドでは、単一フィールド、複合インデックス、スパースインデックス、テキストインデックスなど、さまざまなインデックスタイプの作成とメンテナンスについて詳しく説明しています。`explain()` 関数の適切なインデックス選択と使用を通じてクエリパフォーマンスを最適化し、クエリ実行プランを分析する方法について説明します。

最後のセクションでは、Amazon DocumentDB の変更ストリームを使用したイベント駆動型プログラミングに焦点を当て、Java アプリケーションでリアルタイムのデータ変更モニタリングを実装する方法を示します。変更ストリームカーソルの実装、継続的オペレーションの再開トークンの処理、履歴データ処理の時間ベースのオペレーションについて説明します。このガイドでは、実用的なコード例とベストプラクティスが提供されるため、Amazon DocumentDB で堅牢な Java アプリケーションを構築する際に貴重なリソースとなります。

前提条件

開始する前に、以下があることを確認してください。

- DocumentDB クラスターが設定された AWS アカウント。DocumentDB クラスターのセットアップについては、この入門ブログ記事を参照してください。
- Java Development Kit (JDK) がインストールされました (このガイドでは Amazon Corretto 21 を使用します)。
- 依存関係管理のための Maven。

このガイドのデータモデル

このガイドのすべてのサンプルコードは、「Restaurants」コレクションを持つ「ProgGuideData」テストデータベースへの接続を前提としています。このガイドのすべてのサンプルコードは、レストラン出品システムで動作します。以下は、このシステムのドキュメントの例です。

```
{
    "_id": "ab6ad8f119b5bca3efa2c7ae",
    "restaurantId": "REST-CRT9BL",
    "name": "Thai Curry Palace",
    "description": "Amazing Restaurant, must visit",
    "cuisine": "Thai",
    "address": {
        "street": "914 Park Street",
        "city": "Bryan",
        "state": "AL",
        "zipCode": "96865",
        "location": {
            "type": "Point",
            "coordinates": [-25.4619, 8.389]
        }
    },
    "contact": {
        "phone": "(669) 915-9056 x6657"
    },
    "rating": {
        "average": 3.4,
        "totalReviews": 275
    },
    "priceRange": "$",
    "menu": [{
        "category": "Appetizers",
        "items": [{
            "name": "Buffalo Chicken Wings",
            "price": 13.42
        }]
    }],
    "features": [
        "Private Dining"
    ],
    "isActive": false“ michelin”: {“
        star”: 3,
        “ranking_years”: 4
    }
}
```

CRUD、インデックス管理、イベント駆動型プログラミングを示すすべてのコードサンプルは、MongoClientオブジェクト dbClient、MongoDatabaseオブジェクト

connectionDB、MongoCollectionおよびオブジェクト があることを前提としていますcollection。

> **Note**
>
> このガイドのすべてのコード例は、MongoDB Java ドライバーバージョン 5.3.0 でテストされています。

MongoDB Java ドライバーを使用した Amazon DocumentDB への接続 MongoDB

このセクションでは、Java ドライバーを使用して Amazon DocumentDB に接続するためのstep-by-stepガイドを提供します。これにより、DocumentDB を Java アプリケーションに統合できるようになります。

トピック

- ステップ 1: プロジェクトを設定する
- ステップ 2: 接続文字列を作成する
- ステップ 3: 接続コードを記述する
- ステップ 4: 接続例外を処理する
- ステップ 5: コードを実行する
- 接続のベストプラクティス

ステップ 1: プロジェクトを設定する

1. Maven を使用して、Java プロジェクトを作成します。

```
mvn archetype:generate -DgroupId=com.docdb.guide -DartifactId=my-docdb-project -
DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

2. MongoDB Java ドライバーをプロジェクトの依存関係として「pom.xml」ファイルに追加します。

```
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>5.3.0</version>
</dependency>
```

ステップ 2: 接続文字列を作成する

Amazon DocumentDB 接続文字列は、アプリケーションと DocumentDB クラスター間の接続を確立するために不可欠です。この文字列は、クラスターエンドポイント、ポート、認証の詳細、さまざまな接続オプションなどの重要な情報をカプセル化します。DocumentDB 接続文字列を構築するには、通常、基本的な形式から始めます。

```
"mongodb://username:password@cluster-endpoint:port/?[connection options]"
```

「username」と「password」を実際の認証情報に置き換える必要があります。クラスターのエンドポイントとポート番号は、 AWS Management Console だけでなく、 からも確認できます AWS CLI。クラスターのクラスターエンドポイントを見つける[クラスターのエンドポイントの検索]には、「」を参照してください。DocumentDB のデフォルトポートは 27017 です。

接続文字列の例

- 転送中の暗号化を使用して DocumentDB に接続し、読み取りリクエストがリードレプリカに送信され、プライマリに書き込まれることを確認します。

```
"mongodb://username:password@cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false"
```

- IAM 認証を使用して DocumentDB に接続する：

```
"mongodb://cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false&
   authSource=%24external&
   authMechanism=MONGODB-AWS"
```

接続文字列で使用できるさまざまなオプションは次のとおりです。

- TLS 証明書
- リードレプリカからの読み取り
- 懸念とジャーナリングを書き込む
- RetryWrites

- IAM 認証
- 接続プール
- 接続タイムアウトパラメータ

TLS 証明書

tls=true|false — このオプションは Transport Layer Security (TLS) を有効または無効にします。デフォルトでは、転送中の暗号化は Amazon DocumentDB クラスターで有効になっているため、クラスターレベルで TLS が無効になっていない限り、このオプションの値は である必要がありますtrue。

TLS を使用する場合、コードは DocumentDB クラスターへの接続を作成するときに SSL 証明書を提供する必要があります。クラスターへの安全な接続に必要な証明書をダウンロードします: global-bundle.pem。global-bundle.pem ファイルを使用するには 2 つの方法があります。

- オプション 1 — global-bundle.pem ファイルからすべての証明書を抽出し、Java の keytool を使用して、後でコードで使用できる.jksファイルに保存します。これを行う方法を示すスクリプトTLS が有効な場合の接続については、 の Java タブを参照してください。
- オプション 2 — global-bundle.pemコードを動的に追加し、インメモリキーストアを構築しSSLContext、 を使用して接続の一部として証明書を提供します。

リードレプリカからの読み取り

replicaSet=rs0&readPreference=secondaryPreferred — これら 2 つのオプションを指定すると、すべての読み取りリクエストがリードレプリカにルーティングされ、書き込みリクエストがプライマリインスタンスにルーティングされます。接続文字列replicaSet=rs0で を使用すると、MongoDB ドライバーはクラスタートポロジの自動更新ビューを維持できるため、インスタンスの追加または削除時にアプリケーションが現在のノード設定の可視性を維持できます。これらのオプションを指定しない場合、または を指定しない場合、すべての読み取りと書き込みがプライマリインスタンスreadPreference=primaryに送信されます。のその他のオプションについてはreadPreference、「」を参照してください読み込み設定のオプション。

懸念とジャーナリングを書き込む

書き込みに関する懸念は、書き込みオペレーションのためにデータベースからリクエストされた確認のレベルを決定します。MongoDB ドライバーは、書き込み懸念ファイルとジャーナルファイルを調整するオプションを提供します。Amazon DocumentDB では、書き込みの懸念とジャーナルを設定することは想定されておらず、 w と j (writeConcern と) に送信された値は無視されま

すjournal。DocumentDB は常に writeConcern: majorityおよび journal: でデータを書き込みます。trueそのため、クライアントに確認を送信する前に、書き込みはほとんどのノードに永続的に記録されます。

RetryWrites

retryWrites=false — DocumentDB は再試行可能な書き込みをサポートしていないため、この属性は常に に設定する必要がありますfalse。

IAM 認証

authSource=%24external および **authMechanism=MONGODB-AWS** — この 2 つのパラメータは、 を使用した認証に使用されます AWS Identity and Access Management。IAM 認証は現在、インスタンスベースのクラスターバージョン 5.0 でのみ使用できます。詳細については、「IAM ID を使用した認証」を参照してください。

接続プール

これらのオプションは、接続プーリングで使用できます。

- **maxPoolSize** — プールに作成できる接続の最大数を設定します。すべての接続が使用中で、新しいリクエストが入ると、接続が利用可能になるまで待機します。MongoDB Java ドライバーのデフォルトは 100 です。
- **minPoolSize** — プールで常に維持する必要がある接続の最小数を示します。MongoDB Java ドライバーのデフォルトは 0 です。
- **maxIdleTimeMS** — 接続をプールでアイドル状態のままにして閉じたり削除したりできる時間を決定します。MongoDB Java ドライバーのデフォルトは 100 ミリ秒です。
- **waitQueueTimeoutMS** — プールが最大サイズに達したときに、スレッドが接続が利用可能になるまで待機する時間を設定します。この時間内に接続が利用できない場合は、例外がスローされます。MongoDB Java ドライバーのデフォルト値は 120,000 ミリ秒 (2 分) です。

接続タイムアウトパラメータ

タイムアウトは、オペレーションまたは接続の試行が失敗したと見なされるまでにかかる時間を制限するメカニズムです。次のタイムアウトパラメータは、無期限の待機を防ぎ、リソース割り当てを管理するために使用できます。

- **connectTimeoutMS** — ドライバーがクラスターへの接続を確立するのを待機する時間を設定します。デフォルトは 10,000 ミリ秒 (10 秒) です。

- **socketTimeoutMS** — ドライバーがサーバーからの非書き込みオペレーションのレスポンスを待機する時間を指定します。デフォルトは 0 (タイムアウトや無限) です。
- **serverSelectionTimeoutMS** — ドライバーがクラスター内の使用可能なサーバーを見つけるまで待機する時間を指定します。この設定のデフォルト値は 30 秒で、フェイルオーバー中に新しいプライマリインスタンスを選択するために十分です。

ステップ 3: 接続コードを記述する

次のコード例は、Amazon DocumentDB への TLS 接続を行う方法を示しています。

- Java の KeyStoreおよび SSLContext > オブジェクトが作成されます。
- また、MongoClientSettingsオブジェクトに渡すことでConnectionStringオブジェクトを作成します。TLS 接続を行うには、 MongoClientSettings オブジェクトを使用して connectionstringと をバインドする必要がありますsslcontext。
- を使用すると、 MongoClient オブジェクトMongoClientsを取得します。

```
public static MongoClient makeDbConnection(String dbName, String DbUserName, String
 DbPassword,
    String DbClusterEndPoint, String keyStorePass) throws Exception {
    MongoClient connectedClient;
    String connectionOptions = "?
replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false";
    String connectionUrl = "mongodb://" + DbUserName + ":" + DbPassword + "@" +
 DbClusterEndPoint + ":27017/" +
        dbName + connectionOptions;

    try {
        KeyStore trustStore = KeyStore.getInstance(KeyStore.getDefaultType());
        try (FileInputStream fis = new FileInputStream("src/main/resources/certs/
truststore.jks")) {
            trustStore.load(fis, keyStorePass.toCharArray());
            TrustManagerFactory tmf =
 TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
            tmf.init(trustStore);

            SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
            sslContext.init(null, tmf.getTrustManagers(), new SecureRandom());
            ConnectionString connectionString = new ConnectionString(connectionUrl);
            MongoClientSettings settings = MongoClientSettings.builder()
```

```
                .applyConnectionString(connectionString)
                .applyToSslSettings(builder - > {
                    builder.enabled(true);
                    builder.context(sslContext);
                })
                .build();
            connectedClient = MongoClients.create(settings);
        }
        return connectedClient;
    } catch (MongoException e5) {
        throw new RuntimeException(e5);
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

ステップ 4: 接続例外を処理する

Java アプリケーションで DocumentDB を使用する場合、堅牢で信頼性の高いデータベースオペレーションを維持するには、接続例外の処理が不可欠です。これらの例外を適切に管理することで、問題を迅速に診断できるだけでなく、アプリケーションが一時的なネットワーク中断やサーバーの可用性を正常に処理できるため、安定性とユーザーエクスペリエンスが向上します。接続の確立に関連する重要な例外には、次のようなものがあります。

- **MongoException** — 一般的な例外であり、より具体的な例外の対象ではないさまざまなシナリオで発行される場合があります。これは一般的なすべての MongoDB 例外のキャッチであるため、この例外は他のすべての特定の例外の後に処理されていることを確認してください。
- **MongoTimeoutException** — オペレーションがタイムアウトしたときに発行されます。たとえば、存在しないクラスターエンドポイントをクエリします。
- **MongoSocketException** — ネットワーク関連の問題に対して発行されます。例えば、オペレーション中の突然のネットワーク切断などです。
- **MongoSecurityException** — 認証が失敗した場合に発行されます。たとえば、誤った認証情報で接続するなどです。
- **MongoConfigurationException** — クライアント設定にエラーがある場合に発行されます。たとえば、無効な接続文字列を使用しているとします。

ステップ 5: コードを実行する

次のコードサンプルは、Amazon DocumentDB 接続を作成し、すべてのデータベースを出力します。

```
public static void TestConnection() {
    try (MongoClient mongoClient = makeDbConnection(DATABASE_NAME, DB_USER_NAME,
 DB_PASSWORD, DB_CLUSTER_ENDPOINT, KEYSTORE_PASSWORD)) {
        List < String > databases = mongoClient.listDatabaseNames().into(new ArrayList
 < > ());
        System.out.println("Databases: " + databases);
    } catch (MongoException e) {
        System.err.println("MongoDB error: " + e.getMessage());
        throw new RuntimeException(e);
    }
}
```

接続のベストプラクティス

以下は、MongoDB Java ドライバーを使用して Amazon DocumentDB に接続するときに考慮すべきベストプラクティスです。 MongoDB

- クライアントがリソースをリリースする必要がなくなったMongoClientら、必ず を閉じてください。
- 例外を適切に処理し、適切なエラーログ記録を実装します。
- 環境変数または を使用して AWS Secrets Manager 、ユーザー名やパスワードなどの機密情報を保存します。

Java を使用した Amazon DocumentDB での CRUD オペレーションの実行

このセクションでは、MongoDB Java ドライバーを使用して Amazon DocumentDB で CRUD (作成、読み取り、更新、削除) オペレーションを実行する方法について説明します。

トピック

- DocumentDB コレクションでのドキュメントの作成と挿入
- DocumentDB コレクションからのデータの読み取りと取得
- DocumentDB コレクション内の既存のドキュメントの更新
- DocumentDB コレクションからのドキュメントの削除

- 再試行ロジックによるエラー処理

DocumentDB コレクションでのドキュメントの作成と挿入

Amazon DocumentDB にドキュメントを挿入すると、コレクションに新しいデータを追加できます。挿入を実行するには、ニーズと使用しているデータの量に応じて、いくつかの方法があります。コレクションに個々のドキュメントを挿入する最も基本的な方法は ですinsertOne()。一度に複数のドキュメントを挿入するには、 insertMany()メソッドを使用できます。これにより、1 回のオペレーションでドキュメントの配列を追加できます。DocumentDB コレクションに多くのドキュメントを挿入するもう 1 つの方法は、 ですbulkWrite()。このガイドでは、DocumentDB コレクションでドキュメントを作成するためのこれらのすべての方法について説明します。

insertOne()

まず、個々のドキュメントを Amazon DocumentDBB コレクションに挿入する方法を見てみましょう。単一のドキュメントを挿入するには、 insertOne()メソッドを使用します。このメソッドは、挿入のために BsonDocument を取得し、新しい挿入されたドキュメントのInsertOneResultオブジェクト ID を取得するために使用できるオブジェクトを返します。以下のコード例は、コレクションに 1 つのレストランドキュメントを挿入する方法を示しています。

```
Document article = new Document()
    .append("restaurantId", "REST-21G145")
    .append("name", "Future-proofed Intelligent Bronze Hat")
    .append("cuisine", "International")
    .append("rating", new Document()
        .append("average", 1.8)
        .append("totalReviews", 267))
    .append("features", Arrays.asList("Outdoor Seating", "Live Music"));

try {
    InsertOneResult result = collection.insertOne(article);
    System.out.println("Inserted document with the following id: " +
 result.getInsertedId());
} catch (MongoWriteException e) {
    // Handle duplicate key or other write errors
    System.err.println("Failed to insert document: " + e.getMessage());
    throw e;
} catch (MongoException e) {
    // Handle other MongoDB errors
    System.err.println("MongoDB error: " + e.getMessage());
    throw e;
```

```
}
```

を使用する場合はinsertOne()、適切なエラー処理を必ず含めてください。たとえば、上記のコードでは、restaurantId「」には一意のインデックスがあるため、このコードを再度実行すると、次の が生成されますMongoWriteException。

```
Failed to insert document: Write operation error on server
 docdbCluster.docdb.amazonaws.com:27017.
Write error: WriteError{code=11000, message='E11000 duplicate key error collection:
 Restaurants index: restaurantId_1', details={}}.
```

insertMany()

コレクションに多くのドキュメントを挿入する主な方法は insertMany() と ですbulkWrite()。

insertMany() メソッドは、1 回のオペレーションで複数のドキュメントを挿入する最も簡単な方法です。ドキュメントのリストを受け入れ、コレクションに挿入します。この方法は、互いに独立しており、特別な処理や混合操作を必要としない新しいドキュメントのバッチを挿入する場合に最適です。次のコードは、ファイルから JSON ドキュメントを読み取ってコレクションに挿入する方法を示しています。insertMany() 関数は、挿入されたすべてのドキュメントの IDsを取得するために使用できる InsertManyResultInsertManyResult オブジェクトを返します。

```
// Read JSON file content
String content = new String(Files.readAllBytes(Paths.get(jsonFileName)));
JSONArray jsonArray = new JSONArray(content);

// Convert JSON articles to Documents
List < Document > restaurants = new ArrayList < > ();
for (int i = 0; i < jsonArray.length(); i++) {
    JSONObject jsonObject = jsonArray.getJSONObject(i);
    Document doc = Document.parse(jsonObject.toString());
    restaurants.add(doc);
}
//insert documents in collection
InsertManyResult result = collection.insertMany(restaurants);

System.out.println("Count of inserted documents: " + result.getInsertedIds().size());
```

bulkWrite()

bulkWrite( ) メソッドを使用すると、複数の書き込みオペレーション (挿入、更新、削除) を 1 つのバッチで実行できます。は、一部のドキュメントを挿入して他のドキュメントを更新するなど、1 つのバッチでさまざまなタイプのオペレーションを実行bulkWrite( )する必要がある場合に使用できます。 は、順序付きと順序なしの 2 種類のバッチ書き込みbulkWrite( )をサポートしています。

- 順序付けられたオペレーション — (デフォルト) Amazon DocumentDB は書き込みオペレーションを順番に処理し、最初に発生したエラーで停止します。これは、後のオペレーションが以前のオペレーションに依存する場合など、オペレーションの順序が重要な場合に役立ちます。ただし、順序付けられたオペレーションは、通常、順序付けられていないオペレーションよりも遅くなります。順序付けられたオペレーションでは、バッチが最初のエラーで停止し、一部のオペレーションが未処理のままになる可能性があるケースに対処する必要があります。
- 順序付けされていないオペレーション — Amazon DocumentDB が挿入をデータベース内の 1 回の実行として処理できるようにします。1 つのドキュメントでエラーが発生した場合、オペレーションは残りのドキュメントで続行されます。これは、大量のデータを挿入していて、キーの重複が原因で一部のドキュメントが失敗する可能性のあるデータ移行や一括インポートなど、一部の障害を許容できる場合に特に便利です。順序付けされていないオペレーションでは、一部のオペレーションが成功し、他のオペレーションが失敗する部分的な成功シナリオに対処する必要があります。

bulkWrite( ) メソッドを使用する場合、必須のクラスがいくつかあります。まず、WriteModel クラスはすべての書き込みオペレーションのベースクラスとして機能し、InsertOneModel、、UpdateManyModel、UpdateOneModelDeleteOneModel、 などの特定の実装でさまざまなタイプのオペレーションDeleteManyModelを処理します。

BulkWriteOptions クラスは、順序付き/順序なしの実行の設定やドキュメントの検証のバイパスなど、一括オペレーションの動作を設定するために必要です。BulkWriteResult クラスは、挿入、更新、削除されたドキュメントの数など、実行結果に関する詳細情報を提供します。

エラー処理では、 MongoBulkWriteException クラスには一括オペレーション中の障害に関する情報が含まれているため、 クラスは重要です。一方、 BulkWriteError クラスは個々のオペレーションの障害に関する特定の詳細を提供します。次のコードは、単一のbulkWrite( )メソッド呼び出しの実行中に、ドキュメントのリストを挿入し、単一のドキュメントを更新および削除する例を示しています。このコードは、 BulkWriteOptionsおよび の操作方法とBulkWriteResult、bulkWrite( )オペレーションの適切なエラー処理も示します。

```
List < WriteModel < Document >> bulkOperations = new ArrayList < > ();
// get list of 10 documents representing 10 restaurants
List < Document > restaurantsToInsert = getSampleData();
```

```
for (Document doc: restaurantsToInsert) {
    bulkOperations.add(new InsertOneModel < > (doc));
}
// Update operation
bulkOperations.add(new UpdateOneModel < > (
    new Document("restaurantId", "REST-Y2E9H5"),
    new Document("", new Document("stats.likes", 20))
    .append("", new Document("rating.average", 4.5))));
// Delete operation
bulkOperations.add(new DeleteOneModel < > (new Document("restaurantId", "REST-
D2L431")));

// Perform bulkWrite operation
try {
    BulkWriteOptions options = new BulkWriteOptions()
        .ordered(false); // Allow unordered inserts

    BulkWriteResult result = collection.bulkWrite(bulkOperations, options);

    System.out.println("Inserted: " + result.getInsertedCount());
    System.out.println("Updated: " + result.getModifiedCount());
    System.out.println("Deleted: " + result.getDeletedCount());
} catch (MongoBulkWriteException e) {
    System.err.println("Bulk write error occurred: " + e.getMessage());
    // Log individual write errors
    for (BulkWriteError error: e.getWriteErrors()) {
        System.err.printf("Error at index %d: %s (Code: %d)%n", error.getIndex(),
 error.getMessage(),
            error.getCode());

        // Log the problematic document
        Document errorDoc = new Document(error.getDetails());
        if (errorDoc != null) {
            System.err.println("Problematic document: " + errorDoc);
        }
    }
} catch (Exception e) {
    System.err.println("Error during bulkWrite: " + e.getMessage());
}
```

再試行可能な書き込み

MongoDB とは異なり、Amazon DocumentDB は再試行可能な書き込みをサポートしていません。そのため、特にネットワークの問題や一時的なサービス利用不可を処理するために、アプリケーションにカスタム再試行ロジックを実装する必要があります。適切に実装された再試行戦略では、通常、再試行間の遅延を増やし、再試行の合計数を制限します。エラー処理を使用して再試行ロジックを構築するコードサンプルについては、再試行ロジックによるエラー処理以下を参照してください。

DocumentDB コレクションからのデータの読み取りと取得

Amazon DocumentDB でのドキュメントのクエリは、データを正確に取得して操作できるいくつかの主要なコンポーネントを中心に展開されます。`find()` メソッドは、MongoDB Java ドライバーの基本的なクエリ APIs。これにより、フィルタリング、ソート、結果の射影のための多数のオプションを使用して、複雑なデータ取得が可能になります。`find()` メソッドに加えて、`Filters`と`FindIterable`は、MongoDB Java ドライバーでのクエリオペレーションの構成要素を提供する他の 2 つの基本的なコンポーネントです。

`Filters` クラスは、クエリフィルターを構築するための流暢な API を提供する MongoDB Java ドライバーのユーティリティクラスです。このクラスは、さまざまなクエリ条件を表すBsonオブジェクトのインスタンスを作成する静的ファクトリメソッドを提供します。最も一般的に使用される方法には、等価比較`eq()`には `gt()`、数値比較`gte()``lte()`には `lt()`、、、`and()`および 、複数の条件を組み合わせる`or()`には 、配列メンバーシップテスト`in()``nin()`には 、パターンマッチング`regex()`には が含まれます。クラスは、タイプセーフであるように設計されており、raw ドキュメントベースのクエリと比較してコンパイル時のチェックに優れているため、Java アプリケーションで DocumentDB クエリを構築するための推奨アプローチです。エラー処理は堅牢で、無効なフィルター構造には明確な例外がスローされます。

`FindIterable` は、 `find()`メソッドの結果を処理するように設計された特殊なインターフェイスです。クエリ実行を改良および制御するための豊富なメソッドのセットを提供し、メソッド連鎖のための流暢な API を提供します。インターフェイスには、返されるドキュメントの数`limit()`の制限、`skip()`ページ分割、結果の`sort()`順序付け、特定のフィールドの選択、インデックスの選択など`projection()`、必須`hint()`のクエリ変更方法が含まれています。のバッチ、スキップ、制限オペレーションFindIterableは、データベースからドキュメントを取得して処理する方法を制御するのに役立つ重要なページ分割およびデータ管理ツールです。

バッチ処理 (`batchSize`) は、1 回のネットワークラウンドトリップで DocumentDB がクライアントに返すドキュメントの数を制御します。バッチサイズを設定すると、DocumentDB は一致するすべてのドキュメントを一度に返すのではなく、指定されたバッチサイズのグループで返します。

Skip を使用すると、結果の開始ポイントをオフセットできます。基本的には、一致を返す前に、指定した数のドキュメントをスキップするように DocumentDB に指示します。たとえば、

skip(20)は一致する最初の 20 のドキュメントをバイパスします。これは、後続の結果ページを取得するページ分割シナリオで一般的に使用されます。

制限は、クエリから返されるドキュメントの総数を制限します。を指定するとlimit(n)、DocumentDB はデータベースにより多くの一致がある場合でも、「n」ドキュメントを返した後でドキュメントの返しを停止します。

FindIterable は、Amazon DocumentDB からドキュメントを取得するときに、イテレーターパターンとカーソルパターンの両方をサポートします。イテレーターFindIterableとして を使用する利点は、ドキュメントの遅延ロードを許可し、アプリケーションから要求された場合にのみドキュメントを取得することです。イテレーターを使用するもう 1 つの利点は、クラスターへの接続を維持する責任がないため、接続を明示的に閉じる必要がないことです。

FindIterable は、Amazon DocumentDB クエリを操作するときにカーソルパターンを使用MongoCursorできるようにする のサポートも提供します。 MongoCursorは、データベースオペレーションとリソース管理を制御する MongoDB Java ドライバー固有の実装です。AutoCloseable インターフェイスを実装しているため、try-with-resources ブロックによる明示的なリソース管理が可能になります。これは、データベース接続を適切に閉じ、サーバーリソースを解放するために不可欠です。デフォルトでは、カーソルは 10 分でタイムアウトし、DocumentDB はこのタイムアウト動作を変更するオプションを提供しません。バッチ処理されたデータを使用する場合は、カーソルがタイムアウトする前に、必ず次のデータのバッチを取得してください。を使用する際の 1 つの重要な考慮事項MongoCursorは、リソースリークを防ぐために明示的な閉鎖が必要であることです。

このセクションでは、、find()、Filtersおよび の例をいくつか示しますFindIterable。

次のコード例は、 find()を使用して、restaurantId」フィールドを使用して単一のドキュメントを取得する方法を示しています。

```
Document filter = new Document("restaurantId", "REST-21G145");
Document result = collection.find(filter).first();
```

を使用するとコンパイル時のエラーチェックFiltersが向上しますが、Java ドライバーではfind()メソッドで直接Bsonフィルターを指定することもできます。次のコード例では、Bsonドキュメントを に渡しますfind()。

```
result = collection.find(new Document("$and", Arrays.asList(
    new Document("rating.totalReviews", new Document("$gt", 1000)),
    new Document("priceRange", "$$"))))
```

次のコード例は、 で Filters クラスを使用するいくつかの例を示していますfind()。

```
FindIterable < Document > results;

// Exact match
results = collection.find(Filters.eq("name", "Thai Curry Palace"));

// Not equal
results = collection.find(Filters.ne("cuisine", "Thai"));

// find an element in an array
results = collection.find(Filters.in("features", Arrays.asList("Private Dining")));

// Greater than
results = collection.find(Filters.gt("rating.average", 3.5));

// Between (inclusive)
results = collection.find(Filters.and(
    Filters.gte("rating.totalReviews", 100),
    Filters.lte("rating.totalReviews", 200)));
// AND
results = collection.find(Filters.and(
    Filters.eq("cuisine", "Thai"),
    Filters.gt("rating.average", 4.5)));

// OR
results = collection.find(Filters.or(
    Filters.eq("cuisine", "Thai"),
    Filters.eq("cuisine", "American")));


// All document where the Field exists
results = collection.find(Filters.exists("michelin"));

// Regex
results = collection.find(Filters.regex("name", Pattern.compile("Curry",
 Pattern.CASE_INSENSITIVE)));

// Find all document where the array contain the list of value regardless of its order
results = collection.find(Filters.all("features", Arrays.asList("Private Dining",
 "Parking")));

// Array size
```

```
results = collection.find(Filters.size("features", 4));
```

次の例は、 FindIterable オブジェクトbatchSize()に対する sort()、、limit()、および skip()のオペレーションを連鎖する方法を示しています。これらのオペレーションがどのように提供されるかの順序は、クエリのパフォーマンスに影響します。ベストプラクティスとして、これらのオペレーションの順序は sort()、projection()、skip()、limit()および である必要がありますbatchSize()。

```
FindIterable < Document > results = collection.find(Filters.gt("rating.totalReviews",
 1000))
    // Sorting
    .sort(Sorts.orderBy(
        Sorts.descending("address.city"),
        Sorts.ascending("cuisine")))
    // Field selection
    .projection(Projections.fields(
        Projections.include("name", "cuisine", "priceRange"),
        Projections.excludeId()))

    // Pagination
    .skip(20)
    .limit(10)
    .batchSize(2);
```

次のコード例は、 でのイテレーターの作成を示していますFindIterable。Java の forEachコンストラクトを使用して、結果セットをトラバースします。

```
collection.find(Filters.eq("cuisine", "American")).forEach(doc ->
 System.out.println(doc.toJson()));
```

最後のfind()コード例では、 を使用してcursor()ドキュメントを取得する方法を示しています。試行ブロックにカーソルが作成され、コードが試行ブロックを終了するとカーソルが閉じられます。

```
try (MongoCursor < Document > cursor = collection.find(Filters.eq("cuisine",
 "American"))
    .batchSize(25)
    .cursor()) {
    while (cursor.hasNext()) {
        Document doc = cursor.next();
        System.out.println(doc.toJson());
```

```
    }
} // Cursor automatically closed
```

DocumentDB コレクション内の既存のドキュメントの更新

Amazon DocumentDB は、既存のドキュメントを変更し、存在しないときに新しいドキュメントを挿入するための柔軟で強力なメカニズムを提供します。MongoDB Java ドライバーには、1 つのドキュメントの更新、updateMany()複数のドキュメントの更新、replaceOne()完全なドキュメント置換updateOne()の複数の更新方法が用意されています。これらの 3 つのメソッドに加えて、、UpdateOptions、 UpdateResultはUpdates、MongoDB Java ドライバーでの更新オペレーションの構成要素を提供するその他の基本的なコンポーネントです。

MongoDB Java ドライバーの Updates クラスは、更新演算子を作成するための静的ファクトリメソッドを提供するユーティリティクラスです。これは、タイプセーフで読み取り可能な方法で更新オペレーションを構築するためのプライマリビルダーとして機能します。set()、、 などの基本的なメソッドinc()ではunset()、ドキュメントを直接変更できます。このクラスの能力は、複数の更新オペレーションをアトミックに実行できる Updates.combine()メソッドを使用して複数のオペレーションを組み合わせると明らかになり、データの整合性が確保されます。

UpdateOptions は、ドキュメント更新オペレーションに不可欠なカスタマイズ機能を提供する MongoDB の Java ドライバーの強力な設定クラスです。このクラスの 2 つの重要な点は、更新オペレーションのアップサートと配列フィルターのサポートを提供することです。を介して有効になっているアップサート機能を使用するとupsert(true)、更新オペレーション中に一致するドキュメントが見つからなかった場合に、新しいドキュメントを作成できます。を通じてarrayFilters()、更新オペレーションは特定の基準を満たす配列要素を正確に更新できます。

UpdateResult MongoDB の Java ドライバーでは、更新オペレーションの結果を詳述するフィードバックメカニズムが提供されます。このクラスは、更新基準に一致するドキュメントの数 (matchedCount) 、実際に変更されたドキュメントの数 ()、およびアップサートされたドキュメントに関する情報 (modifiedCount) の 3 つの主要なメトリクスをカプセル化しますupsertedId。これらのメトリクスを理解することは、適切なエラー処理、更新オペレーションの検証、アプリケーションのデータ整合性の維持に不可欠です。

1 つのドキュメントを更新して置き換える

DocumentDB では、updateOne() メソッドを使用して 1 つのドキュメントを更新できます。このメソッドは、通常は Filters クラスによって提供されるフィルターパラメータを使用して、更新するドキュメントを識別し、どのフィールド (複数可) を更新するかを決定する Update パラメータと、更新のさまざまなオプションを設定するオプションのUpdateOptionsパラメータを使用しま

す。updateOne( ) メソッドを使用すると、選択基準に一致する最初のドキュメントのみが更新されます。次のコード例では、1 つのドキュメントの 1 つのフィールドを更新します。

```
collection.updateOne(Filters.eq("restaurantId", "REST-Y2E9H5"),
    Updates.set("name", "Amazing Japanese sushi"));
```

1 つのドキュメント内の複数のフィールドを更新するには、次の例Update.combine()に示すように、 updateOne()で を使用します。この例は、ドキュメント内の配列に項目を追加する方法も示しています。

```
List<Bson> updates = new ArrayList<>();
// Basic field updates
updates.add(Updates.set("name", "Shanghai Best"));
// Array operations
updates.add(Updates.addEachToSet("features", Arrays.asList("Live Music")));
// Counter updates
updates.add(Updates.inc("rating.totalReviews", 10));
// Combine all updates
Bson combinedUpdates = Updates.combine(updates);
// Execute automic update with one call
collection.updateOne(Filters.eq("restaurantId","REST-1J83NH"), combinedUpdates);
```

次のコード例は、データベース内のドキュメントを更新する方法を示しています。指定されたドキュメントが存在しない場合、オペレーションはそれを新しいドキュメントとして自動的に挿入します。このコードは、 UpdateResult オブジェクトを介して利用可能なメトリクスの使用方法も示します。

```
Bson filter = Filters.eq("restaurantId", "REST-0Y9GL0");
Bson update = Updates.set("cuisine", "Indian");
// Upsert operation
UpdateOptions options = new UpdateOptions().upsert(true);
UpdateResult result = collection.updateOne(filter, update, options);

if (result.getUpsertedId() != null) {
    System.out.println("Inserted document with _id: " + result.getUpsertedId());
} else {
     System.out.println("Updated " + result.getModifiedCount() + " document(s)");
}
```

次のコード例は、個々のフィールドを更新するのではなく、 replaceOne()メソッドを使用して既存のドキュメントを新しいドキュメントと完全に置き換える方法を示しています。replaceOne()

メソッドはドキュメント全体を上書きし、元の _idフィールドのみを保持します。複数のドキュメントがフィルター条件に一致する場合、最初に検出されたドキュメントのみが置き換えられます。

```
Document newDocument = new Document()
                .append("restaurantId", "REST-0Y9GL0")
                .append("name", "Bhiryani Adda")
                .append("cuisine", "Indian")
                .append("rating", new Document()
                        .append("average", 4.8)
                        .append("totalReviews", 267))
                .append("features", Arrays.asList("Outdoor Seating", "Live Music"));

UpdateResult result = collection.replaceOne(
                    Filters.eq("restaurantId", "REST-0Y9GL0"),
                    newDocument);
System.out.printf("Modified %d document%n", result.getModifiedCount());
```

複数のドキュメントを更新する

コレクション内の複数のドキュメントを同時に更新するには、2 つの方法があります。updateMany() メソッドを使用するか、 bulkWrite()メソッドUpdateManyModelで を使用できます。updateMany() メソッドは、フィルターパラメータを使用して更新するドキュメントを選択し、 Updateパラメータを使用して更新するフィールドを識別し、オプションの UpdateOptions パラメータを使用して更新オプションを指定します。

次のサンプルコードは、 updateMany()メソッドの使用方法を示しています。

```
Bson filter = Filters.and(
    Filters.in("features", Arrays.asList("Private Dining")),
    Filters.eq("cuisine", "Thai"));
UpdateResult result1 = collection.updateMany(filter, Updates.set("priceRange", "$$$"));
```

次のコード例は、同じ更新を使用する bulkWrite()メソッドを示しています。

```
BulkWriteOptions options = new BulkWriteOptions().ordered(false);
List < WriteModel < Document >> updates = new ArrayList < > ();
Bson filter = Filters.and(
    Filters.in("features", Arrays.asList("Private Dining")),
    Filters.eq("cuisine", "Indian"));
Bson updateField = Updates.set("priceRange", "$$$");
updates.add(new UpdateManyModel < > (filter, updateField));
BulkWriteResult result = collection.bulkWrite(updates, options);
```

```
System.out.printf("Modified %d document%n", result.getModifiedCount());
```

DocumentDB コレクションからのドキュメントの削除

MongoDB Java ドライバーはdeleteOne()、1 つのドキュメントを削除したり、特定の条件に一致する複数のドキュメントを削除deleteMany()したりできます。更新と同様に、削除オペレーションは bulkWrite()メソッドでも使用できます。deleteOne() と はどちらも、削除されたドキュメントの数など、オペレーションの結果に関する情報を提供するDeleteResultオブジェクトdeleteMany()を返します。deleteMany() を使用して複数のドキュメントを削除する例を次に示します。

```
Bson filter = Filters.and(
    Filters.eq("cuisine", "Thai"),
    Filters.lt("rating.totalReviews", 50));
DeleteResult result = collection.deleteMany(filter);
System.out.printf("Deleted %d document%n", result.getDeletedCount());
```

再試行ロジックによるエラー処理

Amazon DocumentDB の堅牢なエラー処理戦略では、エラーを再試行可能 (ネットワークタイムアウト、接続の問題など) と再試行不可能 (認証の失敗、無効なクエリなど) に分類する必要があります。再試行する必要があるエラーによるオペレーションの失敗の場合、各再試行と最大再試行回数の間に時間遅延を実装する必要があります。CRUD オペレーションは、 MongoExceptionとそのサブクラスをキャッチする try-catch ブロックにある必要があります。さらに、運用の可視性のために エラーのモニタリングと ログ記録を含める必要があります。以下は、再試行エラー処理の実装方法を示すサンプルコードです。

```
int MAX_RETRIES = 3;
int INITIAL_DELAY_MS = 1000;
int retryCount = 0;

while (true) {
    try {
        crud_operation(); //perform crud that will throw MongoException or one of its
 subclass
        break;
    } catch (MongoException e) {
        if (retryCount < MAX_RETRIES) {
            retryCount++;
            long delayMs = INITIAL_DELAY_MS * (long) Math.pow(2, retryCount - 1);
            try {
```

```
                TimeUnit.MILLISECONDS.sleep(delayMs);
            } catch (InterruptedException t) {
                Thread.currentThread().interrupt();
                throw new RuntimeException("Retry interrupted", t);
            }
            continue;
        } else
            throw new RuntimeException("Crud operation failed", e);
    }
}
```

Java を使用した Amazon DocumentDB でのインデックス管理

インデックスを使用すると、Amazon DocumentDB コレクションからデータを効率的に取得できます。インデックスがない場合、DocumentDB はコレクション内のすべてのドキュメントをスキャンして、特定のクエリを満たす結果を返す必要があります。このトピックでは、MongoDB Java ドライバーを使用してインデックスを作成、削除、および一覧表示する方法について説明します。また、クエリで特定のインデックスが使用されているかどうかを判断する方法と、特定のインデックスを使用するように Amazon DocumentDB にヒントを与える方法について説明します。

トピック

- Java を使用したインデックスの作成
- インデックスの削除
- インデックス選択の決定とインデックスヒントの提供

Amazon DocumentDB は、多くのタイプのインデックスをサポートしています。サポートされているすべてのインデックスの包括的な概要については、このブログ記事を参照してください。

Java を使用したインデックスの作成

MongoDB Java ドライバーを使用して Amazon DocumentDB でインデックスを作成するメカニズムには、 とrunCommand()、単一のインデックスの createIndex()メソッドまたは複数のインデックスの createIndexes()メソッドの 2 つがあります。 MongoDB createIndex() および createIndexes()メソッドを使用する理由の 1 つは、インデックス作成に関連する特定のエラーをキャッチすることで、より良いエラー処理を構築できることです。これらのメソッドを で使用するもう 1 つの理由はrunCommand()、MongDB Java ドライバーがインデックスの作成と操作のための豊富なサポートクラスを提供することです。これらのサポートクラスは、 createIndex() または createIndexes()メソッドを使用している場合にのみ使用できることに注意してください。3 つのサポートクラスがあります。

- **Indexes** — このクラスは、さまざまなタイプのインデックスを作成するための静的ファクトリメソッドを提供するユーティリティクラスとして機能します。これにより、複雑なインデックス定義を作成するプロセスが簡素化され、一般的に他のインデックス関連クラスと組み合わせて使用されます。
- **IndexModel** — これは、インデックスキー定義とそのオプションの両方をカプセル化する基本的なクラスです。これは、インデックスを作成する内容 (キー) とインデックスを作成する方法 (オプション) を組み合わせた完全なインデックス仕様を表します。このクラスは、createIndexes()メソッドに渡すことができるインデックス仕様のコレクションを定義できるため、複数のインデックスを同時に作成する場合に特に便利です。
- **IndexOptions** — これは、インデックスの動作をカスタマイズするための豊富なメソッドのセットを提供する包括的な設定クラスです。これには、一意のインデックス、スパースインデックス、有効期限 (TTL)、部分フィルター式の設定が含まれます。メソッドチェーンを使用すると、バックグラウンドインデックスの構築や一意の制約など、複数のオプションを設定できます。

単一のインデックスを作成する

この例では、バックグラウンドで createIndex() メソッドを使用して単一のインデックスを作成する方法を示します。背景インデックスとフォアグラウンドインデックスの作成については、「」を参照してくださいインデックスビルドのタイプ。次のコード例ではIndexOptions、 を使用して、バックグラウンドで「unique_restaurantId_idx」という名前の一意のインデックスを作成します。その後、このIndexOptionsオブジェクトは createIndex()メソッドに渡されます。

```
collection.createIndex(
    Indexes.ascending("restaurantId"),
    new IndexOptions()
        .unique(true)
        .name("unique_restaurantId_idx")
        .background(true));
```

複数のインデックスを作成する

この例では、 createIndexes()メソッドを使用して複数のインデックスを作成します。まず、IndexModel オブジェクトを使用して各インデックスのオプションを構築し、次にIndexModelオブジェクトのリストを createIndexes()メソッドに渡します。次のコード例は、 Indexesユーティリティクラスを使用して複合インデックスを作成する方法を示しています。このクラスは、昇順または降順のソート順序を使用してインデックスを作成するかどうかを指定するためにも使用されます。複数のインデックスを作成した後、 listIndexes()メソッドを呼び出してインデックスの作成を検証します。

```
// Single Field Index on cuisine
IndexModel singleIndex = new IndexModel(
    Indexes.ascending("cuisine"),
    new IndexOptions().name("cuisine_idx"));

// Compound Index
IndexModel compoundIndex = new IndexModel(
    Indexes.compoundIndex(
        Indexes.ascending("address.state"),
        Indexes.ascending("priceRange")),
    new IndexOptions().name("location_price_idx"));

// Build a list of IndexModel for the indexes
List < IndexModel > indexes = Arrays.asList(
    singleIndex,
    compoundIndex
);

collection.createIndexes(indexes);

// Verify created indexes
collection.listIndexes().forEach(index - > System.out.println("Created index: " +
 index.toJson()));
```

スパースインデックスと部分インデックスを作成する

この例では、インデックスのタイプ[IndexModel]ごとに を作成して、スパースインデックスと部分インデックスを作成します。

```
// Sparse Index Model, this will identify only those documents that have a
// michelin star rating
IndexModel sparseIndex = new IndexModel(
    Indexes.ascending("michelin.star"),
    new IndexOptions()
    .name("michelin_sparse_idx")
    .sparse(true));

// Partial Index Model where the restaurant is active and has a rating of 4 and above
IndexModel partialIndex = new IndexModel(
    Indexes.ascending("rating.average"),
    new IndexOptions()
    .name("high_rated_active_idx")
    .partialFilterExpression(
```

```
        Filters.and(
            Filters.eq("isActive", true),
            Filters.gte("rating.average", 4.0))));
```

テキストインデックスを作成する

この例では、テキストインデックスを作成する方法を示します。コレクションでは 1 つのテキストインデックスのみが許可されますが、1 つのテキストインデックスを複数のフィールドをカバーする複合インデックスにすることができます。テキストインデックスで複数のフィールドを使用する場合、インデックス内の各フィールドに重みを割り当てることもできます。配列フィールドのテキストインデックスは Amazon DocumentDB ではサポートされていません。複合テキストインデックスでは最大 30 個のフィールドを使用できますが、重みを割り当てることができるフィールドは 3 つだけです。

```
IndexModel textIndex = new IndexModel(
    new Document()
    .append("name", "text")
    .append("description", "text")
    .append("cuisine", "text"),
    new IndexOptions()
    .name("restaurant_text_idx")
    .weights(new Document()
        .append("name", 10) // Restaurant name gets highest weight
        .append("description", 5) // Description get medium weight
        .append("cuisine", 2) // Cuisine type gets low weight
    ));

collection.createIndex(textIndex.getKeys(), textIndex.getOptions());
```

を使用してインデックスを作成する **runCommand()**

Amazon DocumentDB は、インデックスの作成にかかる時間を短縮するために、並列インデックスの作成をサポートしています。並列インデックス作成は、複数の同時ワーカーを使用します。インデックスの作成に使用されるデフォルトのワーカーは 2 です。このブログ記事では、並列インデックス作成について詳しく説明します。現在、MongDB Java ドライバーは、`createIndex()`または を使用しているときにワーカーオプションの指定をサポートしていないため`createIndexes()`、ワーカーを指定する唯一の方法は を使用することです`runCommand`。次のコード例は、 `runCommand`を使用して、ワーカーを 4 つに増やすインデックスを作成する方法を示しています。

```
Document command = new Document("createIndexes", "Restaurants")
```

```
    .append("indexes", Arrays.asList(
        new Document("key", new Document("name", 1))
        .append("name", "restaurant_name_idx")
        .append("workers", 4) // Specify number of workers
    ));

Document commendResult = connectedDB.runCommand(command);
```

インデックスの削除

MongoDB Java ドライバーは、さまざまなシナリオや好みに応じて、インデックスを削除する複数の方法を提供します。インデックスは、名前、キー仕様、またはすべてのインデックスを一度に削除できます。メソッドdropIndex()と は、コレクションオブジェクトで呼び出してインデックスを削除dropIndexes()できます。インデックスを名前で削除するときは、特に複合インデックスまたは自動生成されたインデックスの場合、常に直感的とは限らない正しいインデックス名を使用してください。存在しないインデックスを削除しようとすると、 になりますMongoCommandException。コレクション内のドキュメントの一意性が確保されるため、default _idインデックスを削除することはできません。

次のコード例は、インデックスが作成されたフィールド名を指定するか、すべてのインデックスを削除することで、インデックスを削除する方法を示しています。

```
String indexName = "unique_restaurantId_idx";
Document keys = new Document("cuisine", 1);
// Drop index by name
collection.dropIndex(indexName);

// Drop index by keys
collection.dropIndex(keys);

// Drop all indexes
collection.dropIndexes();
```

複数のキーを使用してインデックスを削除する場合は、指定されたすべてのキーを含む複合インデックスがあり、キーの順序が正しいことを確認してください。上記のインデックス作成サンプルコードは、「cuisine」および 機能の複合キーを示しています。その複合キーを削除しようとしても、その順序が作成に使用された順序ではない場合、MongoCommnadException エラーは次のように発生します。

```
Document keys = new Document("features", 1)
```

```
    .append("cuisine", 1);
try {
    // Drop index by keys
    collection.dropIndex(keys);
    System.out.println("Successfully dropped index with keys: " + keys.toJson());

} catch (MongoCommandException commErr) {
    System.out.println("Error dropping index: " + commErr.getErrorMessage());
    throw new RuntimeException("MongoCommandException was thrown while dropping index",
 commErr);
}
```

次のエラーが表示されます。

```
Error dropping index: Cannot drop index: index not found.
Tests run: 3, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.819 sec <<< FAILURE!
com.amazon.docdb.guide.DocDBGuideTest.testindexGuide()  Time elapsed: 0.817 sec  <<<
 FAILURE!
org.opentest4j.AssertionFailedError: Unexpected exception thrown:
 java.lang.RuntimeException: MongoCommandException was thrown while dropping index
```

インデックス選択の決定とインデックスヒントの提供

Amazon DocumentDB の説明機能を操作することは、クエリのパフォーマンスとインデックスの使用法を理解するために不可欠です。クエリを実行するときに、 explain()メソッドを追加して、どのインデックスが使用されているかなど、クエリプランに関する詳細情報を取得できます。explain() 出力は、クエリ実行ステージ、調査されたドキュメントの数、各ステージにかかる時間に関するインサイトを提供します。この情報は、特定のインデックスが効果的に使用されているかどうか、またはクエリが別のインデックス構造から恩恵を受けることができるかどうかを識別するのに非常に役立ちます。

explain() メソッドは、 find()メソッドと連鎖できます。explain() メソッドは、ExplainVerbosity によって返される詳細レベルを決定するオプションの列挙型を取ることができますexplain()。現時点では、 EXECUTION_STATSおよび QUERY_PLANNER 列挙子のみが DocumentDB でサポートされています。次のコード例は、特定のクエリのクエリプランナーを取得する方法を示しています。

```
// Query we want to analyze
Document query = new Document()
    .append("cuisine", "Thai")
    .append("rating.average", new Document("$gte", 4.0));
```

```
Document allPlansExplain =
 collection.find(query).explain(ExplainVerbosity.QUERY_PLANNER);
System.out.println("All Plans Explain:\n" + allPlansExplain.toJson());
```

クエリプランナーの詳細レベルには、次の JSON ドキュメントが返されます。

```
{
  "queryPlanner": {
    "plannerVersion": 1,
    "namespace": "ProgGuideData.Restaurants",
    "winningPlan": {
      "stage": "IXSCAN",
      "indexName": "cuisine_idx",
      "direction": "forward"
    }
  },
  "serverInfo": {
    "host": "guidecluster3",
    "port": 27017,
    "version": "5.0.0"
  },
  "ok": 1,
  "operationTime": {
    "$timestamp": {
      "t": 1739221668,
      "i": 1
    }
  }
}
```

Amazon DocumentDB が特定のインデックスを使用するように影響または強制するには、いくつかのオプションがあります。hint() および hintString()メソッドを使用すると、クエリに使用するインデックスを明示的に指定することで、クエリオプティマイザのデフォルトのインデックス選択動作を上書きできます。DocumentDB のクエリオプティマイザは通常、インデックスの選択に適切な選択を行いますが、歪んだデータを扱うときやインデックスのパフォーマンスをテストするときなど、特定のインデックスを hint()または で強制することがhintString()有益なシナリオがあります。

次のコード例では、上記のコードで実行されたのと同じクエリに対して複合インデックス「cuisine_features_idx」を強制的に使用します。

```
// Query we want to analyze
Document query = new Document()
    .append("cuisine", "Thai")
    .append("rating.average", new Document("$gte", 4.0));

List < Document > queryDocs = new ArrayList < > ();
collection.find(query).hintString("cuisine_features_idx").forEach(doc - >
 queryDocs.add(doc));
```

Amazon DocumentDB と Java を使用したイベント駆動型プログラミング

Amazon DocumentDB のコンテキストにおけるイベント駆動型プログラミングは、データベースの変更が後続のビジネスロジックとプロセスをトリガーする主要なイベントジェネレーターとして機能する強力なアーキテクチャパターンを表します。DocumentDB コレクションでレコードが挿入、更新、または削除されると、これらの変更はさまざまなダウンストリームプロセス、通知、またはデータ同期タスクを自動的に開始するイベントとして機能します。このパターンは、複数のアプリケーションまたはサービスがデータ変更にリアルタイムで対応する必要がある最新の分散システムでは特に重要です。DocumentDB でイベント駆動型プログラミングを実装する主なメカニズムは、変更ストリームです。

> **Note**
>
> このガイドでは、作業中のコレクションで変更ストリームを有効にしていることを前提としています。コレクションで変更ストリームを有効にするAmazon DocumentDB を用いた変更ストリームの使用方法については、「」を参照してください。

Java アプリケーションからの変更ストリームの使用

MongoDB の Java ドライバーの `watch()`メソッドは、Amazon DocumentDB でリアルタイムのデータ変更をモニタリングするための主要なメカニズムです。`watch()` メソッドは、`MongoClient`、、`MongoDatabase`および `MongoCollection` オブジェクトによって で呼び出すことができます。

この`watch()`メソッドは、更新の完全なドキュメント検索、信頼性のための再開トークンとタイムスタンプの提供、変更をフィルタリングするためのパイプライン集約ステージなど、さまざまな設定オプション`ChangeStreamIterable`をサポートする のインスタンスを返します。

`ChangeStreamIterable` はコア Java インターフェイスを実装`Iterable`しており、 で使用できます`forEach()`。を使用してイベントをキャプチャするには`forEach()`、変更されたイベン

トforEach()を処理するコールバック関数を に渡します。次のコードスニペットは、コレクションで変更ストリームを開いて変更イベントのモニタリングを開始する方法を示しています。

```
ChangeStreamIterable < Document > iterator = collection.watch();
iterator.forEach(event - > {
    System.out.println("Received a change: " + event);
});
```

すべての変更イベントを通過するもう 1 つの方法は、クラスターへの接続を維持し、発生時に新しい変更イベントを継続的に受信するカーソルを開くことです。変更ストリームカーソルを取得するには、 ChangeStreamIterable オブジェクトの cursor()メソッドを使用します。次のコード例は、カーソルを使用して変更イベントをモニタリングする方法を示しています。

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().cursor()) {
    System.out.println(cursor.tryNext());
}
```

ベストプラクティスとして、 try-with-resource ステートメントMongoChangeStreamCursorでを作成するか、カーソルを手動で閉じます。で cursor()メソッドを呼び出すと、ChangeStreamDocument オブジェクト上に作成された MongoChangeStreamCursor ChangeStreamIterableが返されます。

ChangeStreamDocument クラスは、ストリーム内の個々の変更イベントを表す重要なコンポーネントです。これには、オペレーションタイプ (挿入、更新、削除、置換) 、ドキュメントキー、名前空間情報、および利用可能な場合の完全なドキュメントコンテンツなど、各変更に関する詳細情報が含まれます。クラスは、変更のタイプgetOperationType()の決定、完全なドキュメント状態getFullDocument()へのアクセス、変更されたドキュメントのgetDocumentKey()識別など、変更イベントのさまざまな側面にアクセスする方法を提供します。

ChangeStreamDocument オブジェクトは、再開トークンと変更イベントの時刻の 2 つの重要な情報を提供します。

DocumentDB 変更ストリームでのトークンの再開と時間ベースのオペレーションは、継続性を維持し、履歴データアクセスを管理するための重要なメカニズムを提供します。再開トークンは、変更イベントごとに生成される一意の識別子であり、切断または障害後にアプリケーションが特定のポイントから変更ストリーム処理を再開できるようにするブックマークとして機能します。変更ストリームカーソルを作成すると、 resumeAfter()オプションを使用して以前に保存した再開トークンを使

用できます。これにより、ストリームは最初から開始したり、イベントを失うのではなく、中断した場所から続行できます。

変更ストリームの時間ベースのオペレーションは、変更イベントモニタリングの開始点を管理するためのさまざまなアプローチを提供します。startAtOperationTime( ) オプションを使用すると、特定のタイムスタンプ以降に発生した変更の監視を開始できます。これらの時間ベースの機能は、履歴データ処理、point-in-timeリカバリ、またはシステム間の同期を必要とするシナリオで特に役立ちます。

次のコード例は、挿入ドキュメントに関連付けられたイベントを取得し、再開トークンをキャプチャしてから、挿入イベントの後にイベントのモニタリングを開始するトークンを提供します。イベントは更新イベントに関連付けられ、更新が発生したクラスター時間を取得し、そのタイムスタンプをさらなる処理の開始点として使用します。

```
BsonDocument resumeToken;
BsonTimestamp resumeTime;

try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().cursor()) {
    System.out.println("****************** Insert Document ******************");
    ChangeStreamDocument < Document > insertChange = cursor.tryNext();
    resumeToken = insertChange.getResumeToken();
    printJson(cursor.tryNext());
}
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch()
    .resumeAfter(resumeToken)
    .cursor()) {
    System.out.println("****************** Update Document ******************");
    ChangeStreamDocument < Document > insertChange = cursor.tryNext();
    resumeTime = insertChange.getClusterTime();
    printJson(cursor.tryNext());
}
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch()
    .startAtOperationTime(resumeTime)
    .cursor()) {
    System.out.println("****************** Delete Document ******************");
    printJson(cursor.tryNext());
  }
```

デフォルトでは、更新変更イベントには完全なドキュメントが含まれず、加えられた変更のみが含まれます。更新した完全なドキュメントにアクセスする必要がある場合は、ChangeStreamIterable オブジェクトで fullDocument()メソッドを呼び出すことができます。更新イベントに対して完全なドキュメントが返されるように要求すると、ストリームを変更する呼び出しが行われた時点で存在するドキュメントが返されることに注意してください。

このメソッドは、FullDocument列挙型をパラメータとして受け取ります。現在、Amazon DocumentDB は DEFAULT および UPDATE_LOOKUP値のみをサポートしています。次のコードスニペットは、変更の監視を開始するときに、更新イベントの完全なドキュメントを要求する方法を示しています。

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().fullDocument(FullDocument.UPDATE_LOOKUP).cursor())
```

JSON スキーマ検証の使用

$jsonSchema 評価クエリ演算子を使用すると、コレクションに挿入されているドキュメントを検証できます。

トピック

- JSON スキーマ検証の作成と使用
- サポートされるキーワード
- bypassDocumentValidation
- 制限

JSON スキーマ検証の作成と使用

スキーマ検証によるコレクションの作成

createCollection 操作ルールと検証ルールを含むコレクションを作成できます。これらの検証ルールは、Amazon DocumentDB ドキュメントの挿入または更新時に適用されます。以下は、従業員のコレクションの検証ルールを説明するコード例です。

```
db.createCollection("employees", {
   "validator": {
      "$jsonSchema": {
         "bsonType": "object",
         "title": "employee validation",
         "required": [ "name", "employeeId"],
```

```
         "properties": {
            "name": {
                  "bsonType": "object",
                  "properties": {
                     "firstName": {
                        "bsonType": ["string"]
                     },
                     "lastName": {
                        "bsonType": ["string"]
                     }
                  },
                  "additionalProperties" : false
            },
            "employeeId": {
               "bsonType": "string",
               "description": "Unique Identifier for employee"
            },
             "salary": {
               "bsonType": "double"
            },
            "age": {
               "bsonType": "number"
            }
         },
         "additionalProperties" : true
      }
   },
   "validationLevel": "strict", "validationAction": "error"
} )
```

有効なドキュメントを挿入します

次の例では、上記のスキーマ検証ルールに準拠するドキュメントを挿入します。

```
db.employees.insert({"name" : { "firstName" : "Carol" , "lastName" : "Smith"},
 "employeeId": "c720a" , "salary": 1000.0 })
db.employees.insert({ "name" : { "firstName" : "William", "lastName" : "Taylor" },
 "employeeId" : "c721a", "age" : 24})
```

無効なドキュメントを挿入します

次の例では、上記のスキーマ検証ルールに準拠していないドキュメントを挿入します。この例では、employeeId の値は文字列ではありません。

```
db.employees.insert({
    "name" : { "firstName" : "Carol" , "lastName" : "Smith"},
    "employeeId": 720 ,
    "salary": 1000.0
})
```

この例は、文書内の構文が正しくないことを示しています。

コレクションを変更する

collMod コマンドは、既存のコレクションの検証ルールを追加または変更するために使用されます。次の例では、給与フィールドを必須フィールドリストに追加しています。

```
db.runCommand({"collMod" : "employees",
   "validator": {
      "$jsonSchema": {
         "bsonType": "object",
         "title": "employee validation",
         "required": [ "name", "employeeId", "salary"],
         "properties": {
            "name": {
                  "bsonType": "object",
                  "properties": {
                     "firstName": {
                        "bsonType": ["string"]
                     },
                     "lastName": {
                        "bsonType": ["string"]
                     }
                  },
                  "additionalProperties" : false
            },
            "employeeId": {
               "bsonType": "string",
               "description": "Unique Identifier for employee"
            },
             "salary": {
               "bsonType": "double"
            },
            "age": {
               "bsonType": "number"
            }
         },
```

```
         "additionalProperties" : true
      }
   }
} )
```

検証ルールが変更される前に追加されたドキュメントの対処

検証ルールが変更される前にコレクションに追加されたドキュメントの対処には、以下の `validationLevel` 修飾子を使用します。

- strict: すべての挿入と更新に検証ルールを適用します。
- moderate: 既存の有効なドキュメントに検証ルールを適用します。更新中、既存の無効なドキュメントはチェックされません。

次の例では、「employees」という名前のコレクションの検証ルールを更新した後に、給与フィールドが必須になります。次のドキュメントの更新は失敗します。

```
db.runCommand({
    update: "employees",
    updates: [{
        q: { "employeeId": "c721a" },
        u: { age: 25 , salary : 1000},
        upsert: true }]
})
```

Amazon DocumentDB は、次の出力を返します。

```
{
"n" : 0,
    "nModified" : 0,
    "writeErrors" : [
        {
"index" : 0,
            "code" : 121,
            "errmsg" : "Document failed validation"
        }
    ],
    "ok" : 1,
    "operationTime" : Timestamp(1234567890, 1)
}
```

検証レベルを moderate に更新すると、上記のドキュメントを正常に更新できるようになります。

```
db.runCommand({
    "collMod" : "employees",
    validationLevel : "moderate"
})

db.runCommand({
    update: "employees",
    updates: [{
        q: { "employeeId": "c721a" },
        u: { age: 25 , salary : 1000},
        upsert: true }]
})
```

Amazon DocumentDB は、次の出力を返します。

```
{
"n" : 1,
    "nModified" : 1,
    "ok" : 1,
    "operationTime" : Timestamp(1234567890, 1)
}
```

$jsonSchema を使用してドキュメントの取得

$jsonSchema 演算子は、JSON スキーマに一致するドキュメントをクエリするフィルタとして使用できます。これは最上位の演算子で、フィルタードキュメントに最上位フィールドとして含めることも $and、$or、$nor などのクエリ演算子と一緒に使用することもできます。以下の例は、$jsonSchema を個別のフィルターとして、また他のフィルター演算子と併用する方法を示しています。

「employee」コレクションに挿入されたドキュメント:

```
{ "name" : { "firstName" : "Carol", "lastName" : "Smith" }, "employeeId" : "c720a",
 "salary" : 1000 }
{ "name" : { "firstName" : "Emily", "lastName" : "Brown" }, "employeeId" : "c720b",
 "age" : 25, "salary" : 1050.2 }
{ "name" : { "firstName" : "William", "lastName" : "Taylor" }, "employeeId" : "c721a",
 "age" : 24, "salary" : 1400.5 }
{ "name" : { "firstName" : "Jane", "lastName" : "Doe" }, "employeeId" : "c721a",
 "salary" : 1300 }
```

$jsonSchema 演算子のみでフィルタリングされたコレクション:

```
db.employees.find({
       $jsonSchema: { required: ["age"] } })
```

Amazon DocumentDB は、次の出力を返します。

```
{ "_id" : ObjectId("64e5f91c6218c620cf0e8f8b"), "name" : { "firstName" : "Emily",
 "lastName" : "Brown" }, "employeeId" : "c720b", "age" : 25, "salary" : 1050.2 }
{ "_id" : ObjectId("64e5f94e6218c620cf0e8f8c"), "name" : { "firstName" : "William",
 "lastName" : "Taylor" }, "employeeId" : "c721a", "age" : 24, "salary" : 1400.5 }
```

$jsonSchema 演算子と別の演算子でフィルタリングされたコレクション:

```
db.employees.find({
       $or: [{ $jsonSchema: { required: ["age", "name"]}},
            { salary: { $lte:1000}}]});
```

Amazon DocumentDB は、次の出力を返します。

```
{ "_id" : ObjectId("64e5f8886218c620cf0e8f8a"), "name" : { "firstName" : "Carol",
 "lastName" : "Smith" }, "employeeId" : "c720a", "salary" : 1000 }
{ "_id" : ObjectId("64e5f91c6218c620cf0e8f8b"), "name" : { "firstName" : "Emily",
 "lastName" : "Brown" }, "employeeId" : "c720b", "age" : 25, "salary" : 1050.2 }
{ "_id" : ObjectId("64e5f94e6218c620cf0e8f8c"), "name" : { "firstName" : "William",
 "lastName" : "Taylor" }, "employeeId" : "c721a", "age" : 24, "salary" : 1400.5 }
```

$jsonSchema 演算子でフィルタリングされ、集計フィルタでは $match でフィルタリングされたコレクション:

```
db.employees.aggregate(
    [{ $match: {
        $jsonSchema: {
            required: ["name", "employeeId"],
            properties: {"salary" :{"bsonType": "double"}}
        }
       }
    }]
)
```

Amazon DocumentDB は、次の出力を返します。

```
{
"_id" : ObjectId("64e5f8886218c620cf0e8f8a"),
 "name" : { "firstName" : "Carol", "lastName" : "Smith" },
"employeeId" : "c720a",
"salary" : 1000
}
{
"_id" : ObjectId("64e5f91c6218c620cf0e8f8b"),
"name" : { "firstName" : "Emily", "lastName" : "Brown" },
"employeeId" : "c720b",
"age" : 25,
"salary" : 1050.2
}
{
"_id" : ObjectId("64e5f94e6218c620cf0e8f8c"),
"name" : { "firstName" : "William", "lastName" : "Taylor" },
"employeeId" : "c721a",
"age" : 24,
"salary" : 1400.5
}
{
"_id" : ObjectId("64e5f9786218c620cf0e8f8d"),
"name" : { "firstName" : "Jane", "lastName" : "Doe" },
"employeeId" : "c721a",
"salary" : 1300
}
```

既存の検証ルールの表示

コレクションの既存の検証ルールを確認するには、以下を使用してください。

```
db.runCommand({
    listCollections: 1,
    filter: { name: 'employees' }
})
```

Amazon DocumentDB は、次の出力を返します。

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
```

```
                "name" : "employees",
                "type" : "collection",
                "options" : {
                    "autoIndexId" : true,
                    "capped" : false,
                    "validator" : {
                        "$jsonSchema" : {
                            "bsonType" : "object",
                            "title" : "employee validation",
                            "required" : [
                                "name",
                                "employeeId",
                                "salary"
                            ],
                            "properties" : {
                                "name" : {
                                    "bsonType" : "object",
                                    "properties" : {
                                        "firstName" : {
                                            "bsonType" : [
                                                "string"
                                            ]
                                        },
                                        "lastName" : {
                                            "bsonType" : [
                                                "string"
                                            ]
                                        }
                                    },
                                    "additionalProperties" : false
                                },
                                "employeeId" : {
                                    "bsonType" : "string",
                                    "description" : "Unique Identifier for employee"
                                },
                                "salary" : {
                                    "bsonType" : "double"
                                },
                                "age" : {
                                    "bsonType" : "number"
                                }
                            },
                            "additionalProperties" : true
                        }
```

```
                    },
                    "validationLevel" : "moderate",
                    "validationAction" : "error"
                },
                "info" : {
                    "readOnly" : false
                },
                "idIndex" : {
                    "v" : 2,
                    "key" : {
                        "_id" : 1
                    },
                    "name" : "_id_",
                    "ns" : "test.employees"
                }
            }
        ],
        "id" : NumberLong(0),
        "ns" : "test.$cmd.listCollections"
    },
    "ok" : 1,
    "operationTime" : Timestamp(1692788937, 1)
}
```

Amazon DocumentDB は、$out 集約ステージでも検証ルールを保持します。

サポートされるキーワード

create および collMod コマンドでは以下のフィールドがサポートされています。

- **Validator**: $jsonSchem オペレーションをサポートしています。
- **ValidationLevel**: off、strict、moderate の値をサポートします。
- **ValidationAction**: error の値をサポートします。

$jsonSchema 演算子は以下のキーワードをサポートします。

- additionalItems
- additionalProperties
- allOf
- anyOf

- bsonType
- dependencies
- description
- enum
- exclusiveMaximum
- exclusiveMinimum
- items
- maximum
- minimum
- maxItems
- minItems
- maxLength
- minLength
- maxProperties
- minProperties
- multipleOf
- not
- oneOf
- pattern
- patternProperties
- properties
- required
- title
- type
- uniqueItems

bypassDocumentValidation

Amazon DocumentDB は、次のコマンドとメソッドについて bypassDocumentValidation をサポートしています。

- insert

- update
- findAndModify
- aggregate コマンドおよび db.collection.aggregate() メソッドにおける $out ステージ

Amazon DocumentDB は、次のコマンドについては bypassDocumentValidation をサポートしていません。

- aggregate コマンドおよび db.collection.aggregate() メソッドにおける $merge
- mapReduce コマンドおよび db.collection.mapReduce() メソッド
- applyOps コマンド

制限

$jsonSchema の検証には以下の制限が適用されます。

- Amazon DocumentDB は、オペレーションが検証ルールに失敗すると、「ドキュメントの検証に失敗しました」というエラーを返します。
- Amazon DocumentDB Elastic クラスターは、$jsonSchema をサポートしていません。

Amazon DocumentDB のクォータと制限事項

このトピックでは、 Amazon DocumentDB (MongoDB 互換で) のためのリソースのクォータ、制限、命名に関する制約について説明します。

特定の管理機能では、Amazon DocumentDB は Amazon Relational Database Service (Amazon RDS) と Amazon Neptune を用いて共有される運用テクノロジーを使用します。

トピック

- サポートされるインスタンスタイプ
- サポートされるリージョン
- リージョン別クォータ
- 集約限度
- クラスター限度
- インスタンス制限
- 命名に関する制約
- TTL の制約
- エラスティッククラスターの制限
- エラスティッククラスターのシャード制限
- 1 シャードあたりの Elastic クラスターの CPU、メモリ、接続、カーソル限度

サポートされるインスタンスタイプ

Amazon DocumentDB は、オンデマンドインスタンスと以下のインスタンスタイプをサポートしています。

- NVMe-backed:
  - R6GD インスタンスタイプ : `db.r6gd.xlarge`、`db.r6gd.2xlarge`、`db.r6gd.4xlarge`、`db.r6gd.8xlarge`、`db.r6gd.12x`
- メモリ最適化:
  - R6G インスタンスタイプ: `db.r6g.large`、`db.r6g.2xlarge`、`db.r6g.4xlarge`、`db.r6g.8xlarge`、`db.r6g.12xlarge`、

  - R5 インスタンスタイプ: db.r5.large、db.r5.2xlarge、db.r5.4xlarge、db.r5.8xlarge、db.r5.12xlarge、db.r5.24xlarge。
  - R4 インスタンスタイプ: db.r4.large、db.r4.2xlarge、db.r4.4xlarge、db.r4.8xlarge、db.r4.16xlarge。
- バーストパフォーマンス :
  - T4G インスタンスタイプ: db.t4g.medium。
  - T3 インスタンスタイプ db.t3.medium。

サポートされているインスタンスのタイプと仕様の詳細については、「インスタンスクラスの仕様」を参照してください。

サポートされるリージョン

Amazon DocumentDB は、次の AWS リージョンで使用できます。

リージョン名	リージョン	アベイラビリティーゾーン (コンピューティング)
米国東部 (オハイオ)	us-east-2	3
米国東部 (バージニア北部)	us-east-1	6
米国西部 (オレゴン)	us-west-2	4
アフリカ (ケープタウン)	af-south-1	3
南米 (サンパウロ)	sa-east-1	3
アジアパシフィック (香港)	ap-east-1	3
アジアパシフィック (ハイデラバード)	ap-south-2	3
アジアパシフィック (ムンバイ)	ap-south-1	3
アジアパシフィック (ソウル)	ap-northeast-2	4
アジアパシフィック (シンガポール)	ap-southeast-1	3
アジアパシフィック (シドニー)	ap-southeast-2	3
アジアパシフィック (東京)	ap-northeast-1	3

リージョン名	リージョン	アベイラビリティーゾーン (コンピューティング)
カナダ (中部)	ca-central-1	3
中国 (北京) リージョン	cn-north-1	3
中国 (寧夏)	cn-northwest-1	3
欧州 (フランクフルト)	eu-central-1	3
欧州 (アイルランド)	eu-west-1	3
欧州 (ロンドン)	eu-west-2	3
欧州 (ミラノ)	eu-south-1	3
欧州 (パリ)	eu-west-3	3
欧州 (スペイン)	eu-south-2	3
欧州 (ストックホルム)	eu-north-1	3
中東 (アラブ首長国連邦)	me-central-1	3
イスラエル (テルアビブ)	il-central-1	3
AWS GovCloud (米国西部)	us-gov-west-1	3
AWS GovCloud (米国東部)	us-gov-east-1	3

リージョン別クォータ

特定の管理機能では、Amazon DocumentDB は Amazon Relational Database Service (Amazon RDS) と共有の運用テクノロジーを使用します。以下の表では、Amazon DocumentDB と Amazon RDS の間で共有されているリージョンの制限について説明します。

> **Note**
>
> 上記の Amazon RDS 共有テクノロジーは、Amazon DocumentDB インスタンスベースのクラスターにのみ適用されます。Amazon DocumentDB Elastic クラスターは、Amazon RDS とテクノロジーを共有しません。

以下の制限は Amazon DocumentDB インスタンスベースのクラスターに適用され、リージョンごとに AWS アカウントごとに適用されます。

リソース	AWS デフォルトの制限
クラスター	40
クラスターパラメータグループ	50
イベントサブスクリプション	20
インスタンス	40
手動クラススナップショット	100
クラスターあたりのリードレプリカ	15
サブネットグループ	50
サブネットグループあたりのサブネット	20
リソースあたりのタグ	50
インスタンスあたりの VPC セキュリティグループ	5

以下の制限は Amazon DocumentDB エラスティッククラスターに適用され、リージョンごとに AWS アカウントごとに適用されます。

リソース	AWS デフォルトの制限
Elastic クラスター	20
Elastic クラスター vCPU	1024
手動 Elastic クラススナップショット	20

クォータが調整可能な場合は、Service Quotas を使用してクォータの引き上げを要求できます。一部のリクエストは自動的に解決され、他のリクエストは送信されます サポート。送信されたクォータ引き上げリクエストのステータスを追跡できます サポート。サービスクォータを引き上げるリクエストは、優先サポートを受けません。緊急のリクエストがある場合は、サポート にお問い合わせください。サービスクォータの詳細については、「Service Quotas とは」を参照してください。

Amazon DocumentDB のクォータの引き上げをリクエストするには

1. https://console.aws.amazon.com/servicequotas にある Service Quotas コンソールを開き、必要に応じてサインインします。
2. ナビゲーションペインで、[AWS のサービス] を選択します。
3. [Amazon DocumentDB (MongoDB 互換性)] もしくは [Amazon DocumentDB Elastic クラスター] を、リストから選択するかもしくは検索フィールドに入力します。
4. クォータが調整可能な場合は、そのラジオボタンまたは名前を選択し、ページの右上にある [クォータ引き上げリクエスト] を選択します。
5. [クォータ値を変更] に、新しい値を入力します。新しい値は現在値よりも大きい値である必要があります。
6. [リクエスト] を選択します。リクエストが解決されると、クォータの [適用されたクォータ値] が新しい値に設定されます。
7. 保留中または最近解決されたリクエストを表示するには、ナビゲーションペインから [ダッシュボード] を選択します。保留中のリクエストの場合は、リクエストのステータスを選択してリクエストの受信をオープンします。リクエストの初期ステータスは `Pending` です。ステータスがに変わると`Quota requested`、ケース番号が表示されます サポート。リクエストのチケットを開くには、ケース番号を選択します。

集約限度

以下の表では、Amazon DocumentDB 中の集約の制限について説明します。

リソース	[制限]
サポートされるステージの最大数	500

クラスター限度

以下の表は、Amazon DocumentDB インスタンスごとのクラスターの制限について示しています。

リソース	[制限]
クラスターサイズ (すべてのコレクションとインデックスの合計)	128 TiB
コレクションサイズ (すべてのコレクションの合計がクラスターの制限を超えることはできません) - インデックスのサイズは含まれません	32 TiB
クラスターあたりのコレクション数	100,000
クラスターあたりのデータベース数	100,000
データベースサイズ (すべてのデータベースの合計がクラスターの制限を超えることはできません)	128 TiB
ドキュメントのネストの深さ	200 レベル
ドキュメントサイズ	16 MiB
インデックスキーサイズ	2,048 バイト
コレクションごとのインデックス	64
複合インデックスのキー	32
1 回のバッチコマンドでの書き込みの最大数	100,000

リソース	[制限]
クラスターあたりのユーザー数	1,000

インスタンス制限

以下の表は、インスタンスごとの Amazon DocumentDB の制限について示しています。

インスタンスタイプ	インスタンスメモリ（GiB）	接続 (すべて)	カーソル制限	オープントランザクション	接続 (アクティブ)
T3.medium	4	1,000	30	50	102
T4G.medium	4	1,000	30	50	102
R4.large	15.25	1700	450	該当なし	1100
R4.xlarge	30.5	3400	450	該当なし	2700
R4.2xlarge	61	6800	450	該当なし	4500
R4.4xlarge	122	13600	725	該当なし	4500
R4.8xlarge	288	27200	1450	該当なし	4500
R4.16xlarge	488	30000	2900	該当なし	4500
R5.large	16	3400	450	200	1100
R5.xlarge	32	7000	450	400	2700
R5.2xlarge	64	14200	450	800	4500
R5.4xlarge	128	28400	760	1600	4500
R5.8xlarge	256	60000	1520	3200	4500
R5.12xlarge	384	60000	2280	4800	4500
R5.16xlarge	512	60000	3040	6400	4500
R5.24xlarge	768	60000	4560	9600	4500
R6G.large*	16	3400	450	200	1100

インスタンスタイプ	インスタンスメモリ (GiB)	接続 (すべて)	カーソル制限	オープントランザクション	接続 (アクティブ)
R6G.xlarge*	32	7000	450	400	2700
R6G.2xlarge*	64	14200	450	800	4500
R6G.4xlarge*	128	28400	760	1600	4500
R6G.8xlarge*	256	60000	1520	3200	4500
R6G.12xlarge*	384	60000	2280	4800	4500
R6G.16xlarge*	512	60000	3040	6400	4500

* R6GD を含む

次の CloudWatch メトリクスを使用して、インスタンスごとの制限を監視およびアラームできます。Amazon DocumentDB CloudWatch のメトリクスのさらなる詳細については、Amazon DocumentDB と CloudWatch のモニタリング をご参照ください。

リソース	CloudWatch 制限メトリクス	CloudWatch 使用状況メトリクス (最大 1 分)	CloudWatch 使用状況メトリクス
インスタンスメモリ	-	-	FreeableMemory
接続 (すべて)	DatabaseConnectionsLimit	DatabaseConnectionsMax	DatabaseConnections
カーソル	DatabaseCursorsLimit	DatabaseCursorsMax	DatabaseCursors
トランザクション	TransactionsOpenLimit	TransactionsOpenMax	TransactionsOpen

命名に関する制約

次の表に、Amazon DocumentDB 内の命名に関する制約を示します。

リソース	デフォルトの制限
クラスター識別子	• 長さは、[1 ~ 63] 個の英字、数字、またはハイフンです。 • 1 字目は英字である必要があります。 • ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。 • AWS アカウントごと、リージョンごとに、すべてのクラスター (Amazon RDS、Amazon Neptune、Amazon DocumentDB 全体) で一意である必要があります。
コレクション名: <col>	長さは [1 ~ 57] 文字です。
データベース名: <db>	長さは [1 ~ 63] 文字です。
完全修飾コレクション名: <db>.<col>	長さは [3 ~ 120] 文字です。
完全修飾インデックス名: <db>.<col>$<index>	長さは [6 ~ 377] 文字です。
インデックス名: <col>$<index>	長さは [3 ~ 255] 文字です。
インスタンス識別子	• 長さは、[1 ~ 63] 個の英字、数字、またはハイフンです。 • 1 字目は英字である必要があります。 • ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません

リソース	デフォルトの制限
	• AWS アカウント、リージョンごとに、すべてのインスタンス (Amazon RDS、Amazon Neptune、Amazon DocumentDB 全体) で一意である必要があります。

プライマリパスワード	• 長さは、[8 ~ 100] 個の印刷可能な ASCII 文字です。
	• 以下のものを除くすべての印刷可能な ASCII 文字を使用できます。
	  • **/** (スラッシュ)
	  • **"** (二重引用符)
	  • **@** (アットマーク)

プライマリユーザー名	• 長さは、[1 ~ 63] 個の英数字です。 • 1 字目は英字である必要があります。 • データベースエンジンの予約語は使用できません。
パラメータグループ名	• 長さは、[1 ~ 255] 個の英数字です。 • 1 字目は英字である必要があります。 • ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

TTL の制約

TTL インデックスからの削除は、特定の期間内に保証されるものではなく、ベストエフォートです。インスタンスリソースの使用率、ドキュメントサイズ、全体的なスループットなどの要因は、TTL 削除のタイミングに影響を与える可能性があります。

エラスティッククラスターの制限

以下の表は、Amazon DocumentDB のエラスティッククラスターの最大限度について示しています。

リソース	[制限]
リージョンあたりのエラスティッククラスターの数	20
リージョンあたりのすべてのエラスティッククラスターの vCPU の合計数	1024
リージョンあたりの手動クラスタースナップショット数	20
クラスターあたりのシャード数	32
クラスターあたりのストレージ (データがシャードキーによって均等に分配されている場合)	4 PiB
クラスターへの接続	300,000 またはシャード数のうちいずれか低い方の値 x 1 シャードあたりの vCPU に関連付けられる接続制限
シャーディングされていないコレクションのサイズ	32 TiB
シャーディングされたコレクションのサイズ (データがシャードキーによって均等に分配されている場合)	1 PB

リソース	[制限]
クラスターあたりのデータベース数	10,000
クラスターあたりのシャーディングされていないコレクション数	100,000
クラスターあたりのシャーディングされたコレクション数	1,000
クラスターあたりのユーザー数	100
1 回のバッチコマンドでの書き込み数	100,000
コレクションごとのインデックス	64
ドキュメントのネストの深さ	100 レベル
ドキュメントサイズ	16 MB
インデックスキーサイズ	2048 バイト
複合インデックスのキー	32

エラスティッククラスターのシャード制限

以下の表は、Amazon DocumentDB エラスティッククラスター内のシャードの最大限度について示しています。

リソース	[制限]
シャードインスタンスあたりの vCPU 数	64
シャードあたりのインスタンス数	16
1 シャードあたりのストレージ	128 TiB
コレクションあたり、シャードあたりのストレージ	32 TiB

1 シャードあたりの Elastic クラスターの CPU、メモリ、接続、カーソル限度

次の表は、Amazon DocumentDB Elastic クラスターシャードにおける最大の CPU、メモリ、接続、およびカーソル限度を示します。

シャードあたりの vCPU 数	インスタンスメモリ（GiB）	接続制限	カーソル制限
2	16	1700	450
4	32	3500	450
8	64	7100	450
16	128	14200	760
32	256	28400	1520
48	384	30000	2280
64	512	30000	3040

Amazon DocumentDB でのクエリの概要

このセクションでは、Amazon DocumentDB でのクエリのすべての側面について説明します。

トピック

- ドキュメントのクエリ
- クエリプラン
- 結果の説明
- Amazon DocumentDB を使用した地理空間データのクエリ
- 部分インデックス
- Amazon DocumentDB でのテキスト検索の実行

ドキュメントのクエリ

ときどきオンラインストアのインベントリを検索し、販売商品を顧客が表示して購入できることを確認する必要があります。コレクションのクエリは、コレクションのすべてのドキュメントを対象とするか、特定の条件を満たすドキュメントのみを対象とするかにかかわらず、比較的簡単です。

ドキュメントのクエリを実行するには、`find()` オペレーションを使用します。この `find()` コマンドには、返すドキュメントを選択するために使用する条件を定義する単一のドキュメントパラメータがあります。`find()` からの出力は、改行のない 1 行のテキストとしてフォーマットされたドキュメントです。読みやすくするために出力ドキュメントをフォーマットするには、`find().pretty()` を使用します。このトピックのすべての例では、`.pretty()` を使用して出力を書式設定しています。

以下のコードサンプルでは「Working with Documents」の「Adding Documents」セクションにある、前の 2 つの練習問題 - `example` と `insertOne()` で `insertMany()` コレクションに挿入した 4 つのドキュメントを使用します。

トピック

- コレクション内のすべてのドキュメントの取得
- フィールドの値に一致するドキュメントの取得
- 埋め込みドキュメントに一致するドキュメントの取得
- 埋め込まれたドキュメントのフィールド値に一致するドキュメントの取得

- 配列に一致するドキュメントの取得
- 配列値に一致するドキュメントの取得
- 演算子を使用したドキュメントの取得

コレクション内のすべてのドキュメントの取得

コレクション内のすべてのドキュメントを取得するには、find() オペレーションで、空のクエリドキュメントを指定します。

以下のクエリは、example コレクション内のすべてのドキュメントを返します。

```
db.example.find( {} ).pretty()
```

フィールドの値に一致するドキュメントの取得

フィールドおよび値に一致するすべてのドキュメントを取得するには、find() オペレーションで、一致するフィールドおよび値を識別するクエリドキュメントを指定します。

上記のドキュメントを使用して、このクエリは、「Item」フィールドが「Pen」に等しいすべてのドキュメントを返します。

```
db.example.find( { "Item": "Pen" } ).pretty()
```

埋め込みドキュメントに一致するドキュメントの取得

埋め込みドキュメントに一致するすべてのドキュメントを検索するには、find() オペレーションで、埋め込みドキュメントの名前と、その埋め込みドキュメントのすべてのフィールドおよび値を識別するクエリドキュメントを指定します。

埋め込みドキュメントに一致するには、そのドキュメントの埋め込みドキュメントの名前が、クエリで指定した名前と同じであることが必要です。さらに、埋め込みドキュメントのフィールドおよび値も、クエリで指定したものと同じであることが必要です。

次のクエリでは、「Poster Paint」ドキュメントのみ返ります。これは、「Pen」の値は「OnHand」や「MinOnHand」とは異なり、「Spray Paint」には、クエリドキュメントよりもフィールドが 1 つ (OrderQnty) 多いためです。

```
db.example.find({"Inventory": {
```

```
    "OnHand": 47,
    "MinOnHand": 50 } } ).pretty()
```

埋め込まれたドキュメントのフィールド値に一致するドキュメントの取得

埋め込みドキュメントに一致するすべてのドキュメントを検索するには、find() オペレーションで、埋め込みドキュメントの名前と、その埋め込みドキュメントのすべてのフィールドおよび値を識別するクエリドキュメントを指定します。

上記のドキュメントでは、以下のようにクエリで「ドット表記」を使用して埋め込みドキュメントと目的のフィールドを指定しています。埋め込みドキュメント内に他のフィールドがある場合でも、これらに一致するすべてのドキュメントが返されます。クエリは「Poster Paint」と「Spray Paint」を返します。それらはいずれも、指定したフィールドおよび値と一致するためです。

```
db.example.find({"Inventory.OnHand": 47, "Inventory.MinOnHand": 50 }).pretty()
```

配列に一致するドキュメントの取得

配列に一致するすべてのドキュメントを検索するには、find() オペレーションで、目的の配列名とその配列内のすべての値を指定します。クエリは、配列値の名前および順序がクエリで指定したものと一致するすべてのドキュメントを返します。

以下のクエリは「Pen」のみを返します。「Poster Paint」では追加の色 (White) があり、「Spray Paint」では色の順序が異なるためです。

```
db.example.find( { "Colors": ["Red","Green","Blue","Black"] } ).pretty()
```

配列値に一致するドキュメントの取得

特定の属性配列を含むすべてのドキュメントを見つけるには、find() オペレーションで、目的の配列名と値を指定します。

```
db.example.find( { "Colors": "Red" } ).pretty()
```

上記のオペレーションは、すべてのドキュメントを返します。3 つのドキュメントのそれぞれで、Colors という配列があり、その配列のどこかに「Red」という値があるためです。値として「White」を指定すると、クエリからは「Poster Paint」のみが返されます。

演算子を使用したドキュメントの取得

以下のクエリは、「Inventory.OnHand」値が 50 未満のすべてのドキュメントを返します。

```
db.example.find(
        { "Inventory.OnHand": { $lt: 50 } } )
```

サポートされているクエリ演算子のリストについては、「クエリおよびプロジェクション演算子」を参照してください。

クエリプラン

クエリプランの executionStats を表示する方法を教えてください。

クエリの実行速度が予想よりも遅い理由を判断する場合は、クエリプランの executionStats の目的を理解しておくと役立ちます。executionStats は、特定のステージから返されたドキュメントの数 (nReturned)、各ステージで費やされた実行時間 (executionTimeMillisEstimate)、およびクエリプランの生成に要する時間 (planningTimeMillis) を提供します。次のクエリ例に示すように、executionStats の出力から、最も時間がかかるクエリ内のステージを判断して最適化作業を集中させることができます。executionStats パラメータは、現在 update コマンドと delete コマンドをサポートしていません。

> **Note**
>
> Amazon DocumentDB は、分散型で、耐障害性が高く、自己復旧ストレージシステムを利用する専用データベースエンジンで MongoDB 3.6 API をエミュレートしています。その結果、Amazon DocumentDB と MongoDB では、クエリプランと explain() の出力が異なる場合があります。クエリプランを制御する場合は、$hint 演算子を使用して優先インデックスの選択を強制できます。

次のように、explain() コマンドで改善対象のクエリを実行します。

```
db.runCommand({explain: {query document}}).
explain("executionStats").executionStats;
```

以下に示しているのは、オペレーションの例です。

```
db.fish.find({}).limit(2).explain("executionStats");
```

このオペレーションによる出力は、次のようになります。

```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "test.fish",
        "winningPlan" : {
            "stage" : "SUBSCAN",
            "inputStage" : {
                "stage" : "LIMIT_SKIP",
                "inputStage" : {
                    "stage" : "COLLSCAN"
                }
            }
        }
    },
    "executionStats" : {
        "executionSuccess" : true,
        "executionTimeMillis" : "0.063",
        "planningTimeMillis" : "0.040",
        "executionStages" : {
            "stage" : "SUBSCAN",
            "nReturned" : "2",
            "executionTimeMillisEstimate" : "0.012",
            "inputStage" : {
                "stage" : "LIMIT_SKIP",
                "nReturned" : "2",
                "executionTimeMillisEstimate" : "0.005",
                "inputStage" : {
                    "stage" : "COLLSCAN",
                    "nReturned" : "2",
                    "executionTimeMillisEstimate" : "0.005"
                }
            }
        }
    },
    "serverInfo" : {
        "host" : "enginedemo",
        "port" : 27017,
        "version" : "3.6.0"
    },
```

```
    "ok" : 1
}
```

上のクエリの executionStats のみを表示する場合は、次のコマンドを使用できます。小規模なコレクションで、パフォーマンスの向上がごくわずかである場合、Amazon DocumentDB クエリプロセッサはインデックスを使用しないことがあります。

```
db.fish.find({}).limit(2).explain("executionStats").executionStats;
```

クエリプランキャッシュ

パフォーマンスを最適化し、計画期間を短縮するために、Amazon DocumentDB はクエリプランを内部的にキャッシュしています。これにより、同じ形状のクエリは、キャッシュされたプランを使用して直接実行することができます。

ただし、このキャッシュにより、同じクエリでもランダムに遅延が発生することがあります。例えば、通常 1 秒で実行されるクエリが、時には 10 秒かかることがあります。これは、時間の経過とともに、リーダーインスタンスがクエリのさまざまな形状をキャッシュし、メモリを消費するためです。このランダムな速度低下が発生した場合、メモリを解放するために必要なアクションはありません。システムがメモリ使用量を管理し、メモリが一定のしきい値に達すると、自動的に解放されます。

結果の説明

Amazon DocumentDB は、クエリプランに関する情報を返すために、冗長モード queryPlanner をサポートしています。explain の結果は、オプティマイザによって選択されたクエリプランを、以下のような形式で返します。

```
{
   "queryPlanner" : {
      "plannerVersion" : <int>,
      "namespace" : <string>,
      "winningPlan" : {
         "stage" : <STAGE1>,
         ...
         "inputStage" : {
            "stage" : <STAGE2>,
            ...
            "inputStage" : {
```

```
                ...
            }
         }
      }
   }
}
```

以下のセクションでは、一般的な explain の結果を定義します。

トピック

- スキャンとフィルタのステージ
- インデックス公差
- インデックス結合
- 複数のインデックス交差/結合
- 複合インデックス
- ソートステージ
- グループステージ

スキャンとフィルタのステージ

オプティマイザは、以下のいずれかのスキャンのいずれかを選択します。

COLSCAN

このステージは、順次収集スキャンです。

```
{
    "stage" : "COLLSCAN"
}
```

IXSCAN

このステージでは、インデックスキーをスキャンします。オプティマイザはこのステージ内でドキュメントを取得することがあり、その結果、後で FETCH ステージが追加されることがあります。

```
db.foo.find({"a": 1})
{
```

```
    "stage" : "IXSCAN",
    "direction" : "forward",
    "indexName" : <idx_name>
}
```

FETCH

オプティマイザが IXSCAN 以外のステージでドキュメントを取得した場合、その結果には FETCH ステージが含まれます。例えば、上記の IXSCAN クエリは、FETCH と IXSCAN ステージを組み合わせた結果になる可能性があります。

```
db.foo.find({"a": 1})
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXSCAN",
        "indexName" : <idx_name>
    }
}
```

IXONLYSCAN はインデックスキーのみをスキャンします。複合インデックスを作成しても、FETCH は回避されません。

インデックス公差

IXAND

Amazon DocumentDB は、インデックス交差を利用できる場合、IXSCAN の InputStages 配列を持つ IXAND ステージを含めることができます。例えば、次のような出力が表示される場合があります。

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXAND",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
```

```
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

インデックス結合

IXOR

インデックス交差と同様に、Amazon DocumentDBは、IXOR 演算子の inputStages 配列で $or ステージを含むことができます。

```
db.foo.find({"$or": [{"a": {"$gt": 2}}, {"b": {"$lt": 2}}]})
```

上記のクエリの場合、説明出力は以下のようになります。

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

複数のインデックス交差/結合

Amazon DocumentDB は、複数のインデックス交差または結合ステージを組み合わせて、結果を取得できます。例:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                ...
            },
            {
                "stage" : "IXAND",
                "inputStages" : [
                    {
                        "stage" : "IXSCAN",
                        ...
                    },
                    {
                        "stage" : "IXSCAN",
                        ...
                    }
                ]
            }
        ]
    }
}
```

インデックスの交差や結合ステージの使用は、インデックスの種類 (スパース、複合など) に影響されません。

複合インデックス

Amazon DocumentDB の複合インデックスの使用は、インデックスされたフィールドの最初のサブセットに限定されるものではありません。それはサフィックス部分を持つインデックスを使用できますが、あまり効率的ではないかもしれません。

例えば、`{ a: 1, b: -1 }` の複合インデックスは、以下の 3 つのクエリをサポートすることができます。

```
db.orders.find( { a: 1 } )
```

```
db.orders.find( { b: 1 } )
```

```
db.orders.find( { a: 1, b: 1 } )
```

ソートステージ

要求されたソートキーにインデックスがある場合、Amazon DocumentDB はインデックスを使用して順序を取得することができます。その場合、結果には SORT ステージではなく IXSCAN ステージが含まれます。オプティマイザがプレーンソートを好む場合は、次のようなステージが含まれます。

```
{
    "stage" : "SORT",
    "sortPattern" : {
        "a" : 1,
        "b" : -1
    }
}
```

グループステージ

Amazon DocumentDBは、2 つの異なるグループ戦略をサポートしています。

- SORT_AGGREGATE: オンディスクソート集計。
- HASH_AGGREGATE: メモリハッシュ集計。

Amazon DocumentDB を使用した地理空間データのクエリ

このセクションでは、Amazon DocumentDB を使用して地理空間データをクエリする方法について説明します。このセクションを読むと、Amazon DocumentDB に地理空間データを格納、クエリ、インデックス付けする方法について理解することができます。

トピック

- 概要
- 地理空間データのインデックス作成と格納
- 地理空間データのクエリ
- 制限

概要

地理空間の一般的なユースケースには、データからの近接解析が含まれます。たとえば、「シアトルから50マイル以内のすべての空港を検索する」、または「特定の場所から最も近いレストランを探す」などです。Amazon DocumentDB は GeoJSON 仕様 を使用して、地理空間データを表します。GeoJSON は、座標空間内の図形の JSON 形式のためのオープンソース仕様です。GeoJSON 座標は、地球のような球上の位置を表す経度と緯度の両方をキャプチャします。

地理空間データのインデックス作成と格納

Amazon DocumentDB は、地理空間データを格納するために「ポイント」GeoJSON タイプを使用します。各 GeoJSON ドキュメント (またはサブドキュメント) は、通常 2 つのフィールドで構成されます。

- タイプ - 表される図形。Amazon DocumentDB に「座標」フィールドの解釈方法を知らせます。現時点では、Amazon DocumentDB はポイントのみをサポートしています
- 座標 — 配列内のオブジェクトとして表される緯度と経度のペア — [経度、緯度]

Amazon DocumentDB では、2dsphere インデックスを使用して地理空間データのインデックスも使用します。Amazon DocumentDB はインデックスポイントをサポートしています。Amazon DocumentDB は、2dsphere インデックスを使用した近接クエリをサポートしています。

フードデリバリーサービスのアプリケーションを構築しているシナリオを考えてみましょう。さまざまなレストランの緯度と経度のペアを Amazon DocumentDB に保存したい。これを行うには、まず緯度と経度のペアを保持する [地理空間] フィールドにインデックスを作成することをお勧めします。

```
use restaurantsdb
db.usarestaurants.createIndex({location:"2dsphere"})
```

このコマンドの出力は次のようになります。

```
{
 "createdCollectionAutomatically" : true,
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "ok" : 1
}
```

インデックスを作成したら、Amazon DocumentDB コレクションへのデータの挿入をスタートできます。

```
db.usarestaurants.insert({
   "state":"Washington",
   "city":"Seattle",
   "name":"Thai Palace",
   "rating": 4.8,
   "location":{
      "type":"Point",
      "coordinates":[
         -122.3264,
         47.6009
      ]
   }
});

db.usarestaurants.insert({
   "state":"Washington",
   "city":"Seattle",
   "name":"Noodle House",
   "rating": 4.8,
   "location":{
      "type":"Point",
      "coordinates":[
        -122.3517,
         47.6159
      ]
   }
});

db.usarestaurants.insert({
   "state":"Washington",
   "city":"Seattle",
   "name":"Curry House",
   "rating": 4.8,
   "location":{
      "type":"Point",
      "coordinates":[
         -121.4517,
         47.6229
      ]
   }
```

```
});
```

地理空間データのクエリ

Amazon DocumentDB は、地理空間データの近接クエリをサポートしています。近接クエリの良い例は、特定の距離より小さく、別のポイントからの距離を超えるすべてのポイントを検索することです。包含クエリの良い例は、指定したエリア/ポリゴン (ニューヨーク州) にあるすべてのポイント (すべての空港) を検索することです。交差クエリの良い例として、ポイント (都市) と交差するポリゴン (州/地域) の検索があります。次の地理空間演算子を使用して、データからインサイトを得ることができます。

- **$nearSphere** - $nearSphere は、GeoJSON ポイントから最も近いポイントから最も遠いポイントの検索をサポートする検索演算子です。
- **$geoNear** - $geoNear は、GeoJSON ポイントからの距離をメートル単位で計算する集計演算子です。
- **$minDistance** - $minDistance は、$nearSphere または $geoNear と併用される検索演算子で、中心ポイントから指定した最小距離のドキュメントをファイリングします。
- **$maxDistance** - $maxDistance は、$nearSphere または $geoNear と併用される検索演算子で、中心ポイントから指定した最大距離のドキュメントをファイリングします。
- **$geoWithin**: $geoWithin は、ポリゴンなどの特定の形状に完全に存在する地理空間データを含むドキュメントの検索をサポートする検索演算子です。
- **$geoIntersects**: $geoIntersects は、地理空間データが指定された GeoJSON オブジェクトと交差するドキュメントの検索をサポートする検索演算子です。

> **Note**
>
> $geoNear そして $nearSphere は、近接クエリで使用する GeoJSON フィールドに 2dsphere インデックスを必要とします。

例 1

この例では、住所 (ポイント) から最も近い距離でソートされたすべてのレストラン (ポイント) を検索する方法を学習します。

このようなクエリを実行するには、$geoNear を使用して、別のポイントからのポイントのセットの距離を計算します。また、distanceMultiplier を追加して、距離をキロメートル単位で測定します。

```
db.usarestaurants.aggregate([
   {
      "$geoNear":{
         "near":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "spherical":true,
         "distanceField":"DistanceKilometers",
         "distanceMultiplier":0.001
      }
   }
])
```

上記のコマンドでは、指定したポイントからの距離 (最も遠い) でソートされたレストランが返されます。このコマンドの出力は次のようになります。

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "state" : "Washington", "city" :
 "Seattle", "name" : "Noodle House", "rating" : 4.8, "location" : { "type" : "Point",
 "coordinates" : [ -122.3517, 47.6159 ] }, "DistanceKilometers" : 0.03422834547294996 }
{ "_id" : ObjectId("611f3da185009a81ad38e74a"), "state" : "Washington", "city" :
 "Seattle", "name" : "Thai Palace", "rating" : 4.8, "location" : { "type" : "Point",
 "coordinates" : [ -122.3264, 47.6009 ] }, "DistanceKilometers" : 2.5009390081704277 }
{ "_id" : ObjectId("611f3dae85009a81ad38e74c"), "state" : "Washington", "city" :
 "Seattle", "name" : "Curry House", "rating" : 4.8, "location" : { "type" : "Point",
 "coordinates" : [ -121.4517, 47.6229 ] }, "DistanceKilometers" : 67.52845344856914 }
```

クエリの結果の数を制限するには、limit または num オプションを使用します。

limit:

```
db.usarestaurants.aggregate([
   {
      "$geoNear":{
```

```
         "near":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "spherical":true,
         "distanceField":"DistanceKilometers",
         "distanceMultiplier":0.001,
         "limit": 10
      }
   }
])
```

num:

```
db.usarestaurants.aggregate([
   {
      "$geoNear":{
         "near":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "spherical":true,
         "distanceField":"DistanceKilometers",
         "distanceMultiplier":0.001,
         "num": 10
      }
   }
])
```

> **Note**
>
> $geoNear ステージは、返されるドキュメントの最大数を指定する limit および num オプションをサポートします。$geoNear は、limit または num オプションが指定されていない場合、デフォルトで最大 100 個のドキュメントを返します。これは、$limit ステージの値が存在する場合はその値により上書きされ、常に 100 未満の値となります。

例 2

この例では、特定の住所 (ポイント) から 2 キロメートル以内のすべてのレストラン (ポイント) を検索する方法を学習します。このようなクエリを実行するには、GeoJSON ポイントから $nearSphere 最小値以内および $minDistance 最大値 $maxDistance を使用します。

```
db.usarestaurants.find({
   "location":{
      "$nearSphere":{
         "$geometry":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "$minDistance":1,
         "$maxDistance":2000
      }
   }
},
{
   "name":1
})
```

上記のコマンドは、指定されたポイントから最大 2 km のレストランを返します。このコマンドの出力は次のようになります。

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
```

制限

Amazon DocumentDB は、ポリゴン、ラインストリング、マルチポイント、マルチポリゴン、マルチラインストリング、GeometryCollection のクエリやインデックス作成をサポートしていません。

部分インデックス

部分インデックスは、特定のフィルター基準を満たすコレクション内のドキュメントをインデックス化します。部分インデックス機能は、Amazon DocumentDB 5.0 インスタンスベースのクラスターでサポートされています。

トピック

- 部分インデックスを作成する
- サポートされている演算子
- 部分インデックスを使用したクエリ
- 部分インデックス機能
- 部分インデックスに関する制限事項

部分インデックスを作成する

部分インデックスを作成するには、partialFilterExpression オプションで createIndex() メソッドを使用します。例えば、次のオペレーションでは、OrderID を持ち、かつ isDelivered フィールドが true であるドキュメントをインデックス化する一意の複合インデックスを注文コレクションに作成します。

```
db.orders.createIndex(
  {"category": 1, "CustomerId": 1, "OrderId": 1},
  {"unique": true, "partialFilterExpression":
    {"$and": [
      {"OrderId": {"$exists": true}},
      {"isDelivered": {"$eq": false}}
    ]}
  }
)
```

サポートされている演算子

- $eq
- $exists
- $and (最上位のみ)
- $gt/$gte/$lt/$lte (インデックススキャンは、クエリに述されているフィルターが部分フィルター式と完全に一致する場合にのみ使用されます) (制限事項を参照)

部分インデックスを使用したクエリ

部分インデックスの使用により、次のクエリパターンを使用できます。

- クエリ述語は、部分インデックスフィルター式と完全に一致します。

```
db.orders.find({"$and": [
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}}
  ]}).explain()
```

- クエリフィルターの想定される結果は、部分フィルターの論理サブセットです。

```
db.orders.find({"$and": [
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}},
    {"OrderAmount": {"$eq": "5"}}
  ]}).explain()
```

- クエリのサブ述語は、他のインデックスと組み合わせて使用できます。

```
db.orders.createIndex({"anotherIndex":1})
db.orders.find({ "$or": [
      {"$and": [
        {"OrderId": {"$exists": true}},
        {"isDelivered": {"$eq": false}}
      ]},
      {"anotherIndex": {"$eq": 5}}
    ]
  }).explain()
```

> **Note**
>
> クエリプランナーは、効率的に望ましい場合には、インデックススキャンではなくコレクションスキャンを使用することがあります。これは通常、コレクションの大部分を返す、非常に小さなコレクションまたはクエリに使用されます。

部分インデックス機能

部分インデックスを一覧表示する

getIndex オペレーションを使用して、partialFilterExpression を含む部分インデックスを一覧表示します。例えば、getIndex オペレーションは、キー、名前、および partialfilterExpressions フィールドを含む部分インデックスを一覧表示します。

```
db.orders.getIndexes()
```

この例は、以下のサンプル出力を返します。

```
[
  {
    "v" : 4,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "ecommerceApp.orders"
  },
  {
    "v" : 4,
    "unique" : true,
    "key" : {
      "category" : 1,
      "" : 1,
      "CustomerId" : 1,
      "OrderId" : 1
    },
    "name" : "category_1_CustID_1_OrderId_1",
    "ns" : "ecommerceApp.orders",
    "partialFilterExpression" : {
      "$and" : [
        {"OrderId": {"$exists": true}},
        {"isDelivered": {"$eq": false}}
      ]
    }
  }
]
```

同じキー: 順序の複数の部分フィルター式

同じフィールドの組み合わせ (キー:順序) に対して異なる部分インデックスを作成できます。これらのインデックスには別の名前が必要です。

```
db.orders.createIndex(
  {"OrderId":1},
  {
    name:"firstPartialIndex",
    partialFilterExpression:{"OrderId":{"$exists": true}}
  }
)
```

```
db.orders.createIndex(
  {"OrderId":1},
  {
    name:"secondPartialIndex",
    partialFilterExpression:{"OrderId":{"$gt": 1000}}
  }
)
```

getIndexes オペレーションを実行して、コレクション内のすべてのインデックスを一覧表示します。

```
db.orders.getIndexes()
```

コマンドは、次の出力例を返します。

```
[
  {
    "v" : 4,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "ecommerceApp.orders"
  },
  {
    "v" : 4,
    "key" : {
      "OrderId" : 1
    },
    "name" : "firstPartialIndex",
    "ns" : "ecommerceApp.orders",
    "partialFilterExpression" : {"OrderId":{"$exists": true}}
  },
```

```
  {
    "v" : 4,
    "key" : {
      "OrderId" : 1
    },
    "name" : "secondPartialIndex",
    "ns" : "ecommerceApp.orders",
    "partialFilterExpression" : {"OrderId":{"$gt": 1000}}
  }
]
```

> **Important**
>
> インデックス名は異なる必要があり、名前のみで削除する必要があります。

部分プロパティと TTL プロパティを持つインデックス

インデックスの作成時に partialFilterExpression と expireAfterSeconds の両方のオプションを指定することで、部分プロパティと TTL プロパティを持つインデックスを作成することもできます。これにより、コレクションからどのようなドキュメントを除外するのかをよりきめ細かにコントロールできます。

例えば、特定の期間後に削除するドキュメントを特定する TTL インデックスがあるとします。ここで部分インデックスオプションを使用することで、ドキュメントを削除するタイミングに関する追加の条件を指定することができます。

```
db.orders.createIndex(
    { "OrderTimestamp": 1 },
    {
        expireAfterSeconds: 3600 ,
        partialFilterExpression: { "isDelivered": { $eq: true } }
    }
)
```

この例は、以下のサンプル出力を返します。

```
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
```

```
        "numIndexesAfter" : 2,
        "ok" : 1,
        "operationTime" : Timestamp(1234567890, 1)
}
```

getIndexes オペレーションを実行して、コレクションに存在するインデックスを一覧表示します。

```
db.orders.getIndexes()
[
    {
        "v" : 4,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.orders"
    }
```

この例は、以下のサンプル出力を返します。

```
[
    {
        "v": 4,
        "key": {
            "_id": 1
        },
        "name": "_id_",
        "ns": "ecommerceApp.orders"
    },
    {
        "v": 4,
        "key": {
            "OrderTimestamp": 1
        },
        "name": "OrderTimestamp_1",
        "ns": "ecommerceApp.orders",
        "partialFilterExpression": {
            "isDelivered": {
                "$eq": true
            }
        },
        "expireAfterSeconds": 3600
```

```
    }
]
```

部分インデックスに関する制限事項

部分インデックス機能には、次の制限事項が適用されます。

- Amazon DocumentDB の不等式クエリは、クエリフィルター述語が `partialFilterExpression` と完全に一致し、かつ同じデータ型である場合にのみ、部分インデックスを使用します。

  > **Note**
  >
  > 上記のケースでは、`$hint` は IXSCAN の執行にも使用できません。

  次の例では、`partialFilterExpression` は `field1` にのみ適用され、`field2` には適用されません。

  ```
  db.orders.createIndex(
    {"OrderAmount": 1},
    {"partialFilterExpression": { OrderAmount : {"$gt" : 5}}}
  )

  db.orders.find({OrderAmount : {"$gt" : 5}}) // Will use partial index
  db.orders.find({OrderAmount : {"$gt" : 6}}) // Will not use partial index
  db.orders.find({OrderAmount : {"$gt" : Decimal128(5.00)}}) // Will not use partial
   index
  ```

- 配列演算子を持つ `partialFilterExpression` はサポートされていません。次の演算はエラーを生じます。

  ```
  db.orders.createIndex(
    {"CustomerId":1},
    {'partialFilterExpression': {'OrderId': {'$eq': [1000, 1001, 1002]}}}
  )
  ```

- partialFilterExpression フィールドでは、次の演算子はサポートされていません。
  - `$all` (配列演算子)
  - `$mod` (配列演算子)

- $or
- $xor
- $not
- $nor

- フィルター式とフィルターのデータ型は同じである必要があります。

Amazon DocumentDB でのテキスト検索の実行

Amazon DocumentDB のネイティブフルテキスト検索機能を使用することで、特定の目的向けのテキストインデックスを使用して、サイズの大きなテキストデータセットに対してテキスト検索を実行できます。このセクションでは、テキストインデックス機能の機能について説明し、Amazon DocumentDB でテキストインデックスを作成および使用する手順について説明します。テキスト検索における制限事項もリストアップします。

トピック

- サポートされている機能
- Amazon DocumentDB テキストインデックスの使用
- MongoDB との違い
- ベストプラクティスとガイドライン
- 制限

サポートされている機能

Amazon DocumentDB テキスト検索は、以下の MongoDB API 互換機能をサポートしています。

- 1 つのフィールドにテキストインデックスを作成します。
- 複数のテキストフィールドを含む複合テキストインデックスを作成します。
- 単一単語または複数単語の検索を実行します。
- 重み付けにより検索結果を制御します。
- 検索結果をスコア順にソートします。
- 集約パイプラインでテキストインデックスを使用します。
- 完全一致フレーズを検索できます。

Amazon DocumentDB テキストインデックスの使用

文字列データを含むフィールドにテキストインデックスを作成するには、次のように文字列「text」を指定します。

単一フィールドインデックス

```
db.test.createIndex({"comments": "text"})
```

このインデックスは、指定されたコレクションの「comments」文字列フィールドのテキスト検索クエリをサポートします。

複数の文字列フィールドに複合テキストインデックスを作成します。

```
db.test.createIndex({"comments": "text", "title":"text"})
```

このインデックスは、指定されたコレクションの「comments」および「title」文字列フィールドでのテキスト検索クエリをサポートします。複合テキストインデックスを作成する際は、最大 30 個のフィールドを指定できます。作成したテキスト検索クエリは、すべてのインデックス付きフィールドを対象にクエリを実行します。

> **Note**
>
> 各コレクションで許可されるテキストインデックスは 1 つだけです。

Amazon DocumentDB コレクションでのテキストインデックスの一覧表示

以下の例に示すように、コレクションで `getIndexes()` を使用して、テキストインデックスを含むインデックスを特定して記述できます。

```
rs0:PRIMARY> db.test.getIndexes()
[
   {
      "v" : 4,
      "key" : {
         "_id" : 1
      },
      "name" : "_id_",
      "ns" : "test.test"
```

```
   },
   {
      "v" : 1,
      "key" : {
         "_fts" : "text",
         "_ftsx" : 1
      },
      "name" : "contents_text",
      "ns" : "test.test",
      "default_language" : "english",
      "weights" : {
         "comments" : 1
      },
      "textIndexVersion" : 1
   }
]
```

インデックスを作成したら、Amazon DocumentDB コレクションへのデータの挿入をスタートできます。

```
db.test.insertMany([{"_id": 1, "star_rating": 4, "comments": "apple is red"},
                    {"_id": 2, "star_rating": 5, "comments": "pie is delicious"},
                    {"_id": 3, "star_rating": 3, "comments": "apples, oranges - healthy
 fruit"},
                    {"_id": 4, "star_rating": 2, "comments": "bake the apple pie in the
 oven"},
                    {"_id": 5, "star_rating": 5, "comments": "interesting couch"},
                    {"_id": 6, "star_rating": 5, "comments": "interested in couch for
 sale, year 2022"}])
```

テキスト検索クエリの実行

単一語テキスト検索クエリを実行する

テキスト検索を実行するには、$text と $search 演算子を使用する必要があります。次の例では、テキストインデックス付きフィールドに文字列「apple」または、「apple」を他の形式で含む(「apples」、等)文字列を含むすべてのドキュメントを返します。

```
db.test.find({$text: {$search: "apple"}})
```

出力:

このコマンドの出力は次のようになります。

```
{ "_id" : 1, "star_rating" : 4, "comments" : "apple is red" }
{ "_id" : 3, "star_rating" : 3, "comments" : "apples, oranges - healthy fruit" }
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
```

複数単語のテキスト検索を実行する

Amazon DocumentDB データに対して複数単語のテキスト検索を実行することもできます。次のコマンドは、「apple」または「pie」を含むテキストインデックス付きフィールドを持つドキュメントを返します。

```
db.test.find({$text: {$search: "apple pie"}})
```

出力:

このコマンドの出力は次のようになります。

```
{ "_id" : 1, "star_rating" : 4, "comments" : "apple is red" }
{ "_id" : 2, "star_rating" : 5, "comments" : "pie is delicious" }
{ "_id" : 3, "star_rating" : 3, "comments" : "apples, oranges - healthy fruit" }
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
```

複数単語のフレーズテキスト検索を実行する

複数単語のフレーズ検索には、次の例を使用します。

```
db.test.find({$text: {$search: "\"apple pie\""}})
```

出力:

上記のコマンドは、「apple pie」というフレーズに完全一致する文字列を含むテキストインデックス付きフィールドを持つドキュメントを返します。このコマンドの出力は次のようになります。

```
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
```

フィルターを使用してテキスト検索を実行する

テキスト検索を他のクエリ演算子と組み合わせて、追加の基準に基づいて結果をフィルタリングすることもできます。

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "interest"}}]})
```

出力:

上記のコマンドは、「interest」のすべての変化形と、「星評価」が 5 であるテキストインデックス付きフィールドを持つドキュメントを返します。このコマンドの出力は次のようになります。

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
{ "_id" : 6, "star_rating" : 5, "comments" : "interested in couch for sale, year
 2022" }
```

テキスト検索で返されるドキュメントの数を制限する

`limit` を使用して、返されるドキュメントの数を制限できます。

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "couch"}}]}).limit(1)
```

出力:

上記のコマンドは、フィルターを満たす 1 つの結果を返します。

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
```

テキストスコアで結果をソートする

次の例では、テキスト検索結果をテキストスコアでソートします。

```
db.test.find({$text: {$search: "apple"}}, {score: {$meta: "textScore"}}).sort({score:
 {$meta: "textScore"}})
```

出力:

上記のコマンドは、文字列「apple」または、「apple」を他の形式で含む(「apples」、等)文字列含むテキストインデックス付きフィールドを持つドキュメントを返し、それぞれのドキュメントの検索用語との関連性の高さに基づいて結果をソートします。このコマンドの出力は次のようになります。

```
{ "_id" : 1, "star_rating" : 4, "comments" : "apple is red", "score" :
 0.6079270860936958 }
{ "_id" : 3, "star_rating" : 3, "comments" : "apples, oranges - healthy fruit",
 "score" : 0.6079270860936958 }
```

```
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven",
 "score" : 0.6079270860936958 }
```

$text および $search は、aggregate、count、findAndModify、update、delete の各コマンドでもサポートされています。

集約演算子

$match を使用した集約パイプライン

```
db.test.aggregate(
   [{ $match: { $text: { $search: "apple pie" } } }]
)
```

出力:

このコマンドでは次のようなレスポンスが返されます。

```
{ "_id" : 1, "star_rating" : 4, "comments" : "apple is red" }
{ "_id" : 3, "star_rating" : 3, "comments" : "apple - a healthy fruit" }
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
{ "_id" : 2, "star_rating" : 5, "comments" : "pie is delicious" }
```

他の集約演算子の組み合わせ

```
db.test.aggregate(
   [
      { $match: { $text: { $search: "apple pie" } } },
      { $sort: { score: { $meta: "textScore" } } },
      { $project: { score: { $meta: "textScore" } } }
   ]
)
```

出力:

このコマンドでは次のようなレスポンスが返されます。

```
{ "_id" : 4, "score" : 0.6079270860936958 }
{ "_id" : 1, "score" : 0.3039635430468479 }
{ "_id" : 2, "score" : 0.3039635430468479 }
{ "_id" : 3, "score" : 0.3039635430468479 }
```

テキストインデックスを作成するときに複数のフィールドを指定する

重み付けは、複合テキストインデックスの最大 3 つのフィールドに付与することができます。テキストインデックスの各フィールドに割り当てられたデフォルトの重み値は 1 です。重み付けはオプションのパラメータであり、1～100,000 の範囲である必要があります。

```
db.test.createIndex(
   {
     "firstname": "text",
     "lastname": "text",
     ...
   },
   {
     weights: {
       "firstname": 5,
       "lastname":10,
       ...
     },
     name: "name_text_index"
   }
 )
```

MongoDB との違い

Amazon DocumentDB のテキストインデックス機能は、用語-頻出度アルゴリズムによる転置インデックスを使用します。デフォルトでは、テキストインデックスはスパースです。解析ロジック、トークン化区切り文字などが異なるため、同じデータセットまたはクエリ形状に対して MongoDB と同じ結果セットが返されない場合があります。

Amazon DocumentDB テキストインデックスと MongoDB には、他にも以下のような違いがあります。

- テキスト以外のインデックスを用いた複合インデックスはサポートされていません。
- Amazon DocumentDB テキストインデックスでは、大文字と小文字は区別されません。
- テキストインデックスでは、英語のみがサポートされています。
- 配列 (またはマルチキー) フィールドのテキストインデックス構成はサポートされていません。例えば、文字 a のテキストインデックスとして、{“a”:[“apple”, “pie”]} などといったドキュメントを使用したインデックスを作成しようとすると失敗します。
- ワイルドカードのテキストインデックス構成はサポートされていません。

- 一意のテキストインデックスはサポートされていません。
- 用語の除外はサポートされていません。

ベストプラクティスとガイドライン

- テキストスコアによるソートを含むテキスト検索クエリのパフォーマンスを最適化するには、データをロードする前に実際にテキストインデックスを作成してみることをお勧めします。
- テキストインデックスの運用には、インデックス化されたデータの最適化された内部コピーを保持するために追加のストレージが必要となります。これには、追加のコストが伴います。

制限

テキスト検索は Amazon DocumentDB で次の制限があります。

- テキスト検索は、Amazon DocumentDB 5.0 インスタンスベースのクラスターでのみサポートされています。
- テキストインデックスには、レグゼームとその位置情報が保存されます。1 つのドキュメント内のすべてのレグゼームとその位置情報の合計サイズは 1MB に制限されています。

Amazon DocumentDB のトラブルシューティング

以下のセクションでは、Amazon DocumentDB (MongoDB 互換) の使用時に発生する可能性がある問題をトラブルシューティングする方法について説明します。

トピック

- 接続の問題のトラブルシューティング
- インデックス作成のトラブルシューティング
- パフォーマンスとリソース使用率のトラブルシューティング

接続の問題のトラブルシューティング

接続に問題がありますか。一般的なシナリオと、それらを解決する方法は、次の通りです。

トピック

- Amazon DocumentDB エンドポイントに接続できない
- Amazon DocumentDB インスタンスへの接続のテスト
- 無効なエンドポイントへの接続
- 接続数に影響するドライバー設定

Amazon DocumentDB エンドポイントに接続できない

Amazon DocumentDB に接続しようとすると、以下のような最も一般的なエラーメッセージが表示されることがあります。

```
connecting to: mongodb://docdb-2018-11-08-21-47-27.cluster-ccuszbx3pn5e.us-east-
1.docdb.amazonaws.com:27017/
2018-11-14T14:33:46.451-0800 W NETWORK [thread1] Failed to connect to
172.31.91.193:27017 after 5000ms milliseconds, giving up.
2018-11-14T14:33:46.452-0800 E QUERY [thread1] Error: couldn't connect to server
docdb-2018-11-08-21-47-27.cluster-ccuszbx3pn5e.us-east-1.docdb.amazonaws.com:27017,
connection attempt failed :
connect@src/mongo/shell/mongo.js:237:13
@(connect):1:6
```

```
exception: connect failed
```

このエラーメッセージは、クライアント (この例では mongo シェル) が Amazon DocumentDB エンドポイントにアクセスできないことを意味します。これには以下のようにいくつかの原因が考えられます。

トピック

- パブリックエンドポイントからの接続
- リージョン間接続
- 別の Amazon VPC からの接続
- セキュリティグループがインバウンド接続をブロックする
- Java Mongo ドライバーの読み取り設定に関する問題

パブリックエンドポイントからの接続

ノートパソコンやローカル開発マシンから直接 Amazon DocumentDB クラスターに接続しようとしています。

ノートパソコンやローカル開発マシンなどのパブリックエンドポイントから Amazon DocumentDB クラスターに直接接続しようとしても失敗します。Amazon DocumentDB は仮想プライベートクラウド (VPC) 専用で、現在パブリックエンドポイントをサポートしていません。したがって、ノートパソコンまたは VPC の外部のローカル開発環境から直接 Amazon DocumentDB クラスターに接続することはできません。

Amazon VPC の外部から Amazon DocumentDB クラスターに接続するには、SSH トンネルを使用します。詳細については、「Amazon VPC の外部から Amazon DocumentDB クラスターへの接続」を参照してください。さらに、開発環境が別の Amazon VPC にある場合は、VPC ピアリングを使用して、同じリージョンまたは別のリージョンにある別の Amazon VPC から Amazon DocumentDB クラスターに接続することもできます。

リージョン間接続

他のリージョンにある Amazon DocumentDB クラスターに接続しようとしています。

クラスターのリージョン以外のリージョンにある Amazon EC2 インスタンスから Amazon DocumentDB クラスターに接続しようとするとします。例えば、米国西部 (オレゴン) リージョン (us-west-2) から米国東部 (バージニア北部) リージョン (us-east-1) からクラスターに接続しようとしても失敗します。

Amazon DocumentDB クラスターのリージョンを確認するには、以下のコマンドを実行します。そのリージョンはエンドポイントにあります。

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].Endpoint'
```

このオペレーションによる出力は、次のようになります。

```
[
    "sample-cluster.node.us-east-1.docdb.amazonaws.com"
]
```

EC2 インスタンスのリージョンを確認するには、以下のコマンドを実行します。

```
 aws ec2 describe-instances \
     --query 'Reservations[*].Instances[*].Placement.AvailabilityZone'
```

このオペレーションによる出力は、次のようになります。

```
[
    [
        "us-east-1a"
    ]
]
```

別の Amazon VPC からの接続

クラスターのデプロイ先の Amazon VPC とは異なる VPC から Amazon DocumentDB クラスターに接続しようとしています。

Amazon DocumentDB クラスターと Amazon EC2 インスタンスの両方が同じ にあり AWS リージョン、同じ Amazon VPC にない場合、2 つの Amazon VPCs 間で VPC ピアリングが有効になっていない限り、Amazon DocumentDB クラスターに直接接続することはできません。

Amazon DocumentDB インスタンスの Amazon VPC を確認するには、以下のコマンドを実行します。

```
aws docdb describe-db-instances \
```

```
   --db-instance-identifier sample-instance \
   --query 'DBInstances[*].DBSubnetGroup.VpcId'
```

Amazon ÉC2 インスタンスの Amazon VPC を確認するには、以下のコマンドを実行します。

```
aws ec2 describe-instances \
   --query 'Reservations[*].Instances[*].VpcId'
```

セキュリティグループがインバウンド接続をブロックする

Amazon DocumentDB クラスターに接続しようとしているが、クラスターのセキュリティグループはクラスターのポート (デフォルトポート: 27017) でインバウンド接続を許可していない。

Amazon DocumentDB クラスターと Amazon EC2 インスタンスが両方とも同じリージョン内および同じ Amazon VPC 内にあり、同じ Amazon VPC セキュリティグループを使用しているとします。Amazon DocumentDB クラスターに接続できない場合は、クラスターのセキュリティグループ (ファイアウォール) で、Amazon DocumentDB クラスター用に選択したポート (デフォルトポートは 27017) でインバウンド接続を許可していないことが原因と考えられます。

Amazon DocumentDB クラスターのポートを確認するには、以下のコマンドを実行します。

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[DBClusterIdentifier,Port]'
```

Amazon DocumentDB クラスターのセキュリティグループを取得するには、以下のコマンドを実行します。

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[VpcSecurityGroups[*],VpcSecurityGroupId]'
```

セキュリティグループのインバウンドルールを確認するには、Amazon EC2 ドキュメントの以下のトピックを参照してください。

- Linux インスタンスのインバウンドトラフィックの承認
- Windows インスタンスのインバウンドトラフィックの承認

Java Mongo ドライバーの読み取り設定に関する問題

クライアントの読み取り設定は適用されず、一部のクライアントは再起動しない限りフェイルオーバー後に Amazon DocumentDB に書き込むことができません。

この問題は、Java Mongo ドライバー 3.7.x で初めて発見されましたが、クライアントが MongoClientSettings を使用して Amazon DocumentDB への接続を確立したとき、具体的には applyToClusterSettings メソッドをチェーニングしたときに発生します。MongoClient クラスター設定は、hosts()、requiredReplicaSetName()、mode() などのいくつかの異なる方法を使用して定義できます。

クライアントが hosts() メソッドでホストを 1 つだけ指定した場合、モードは ClusterConnectionMode.MULTIPLE の代わりに ClusterConnectionMode.SINGLE に設定されます。これにより、クライアントは読み取り設定を無視し、hosts() で設定されているサーバーにのみ接続します。そのため、クライアントの設定が以下のように初期化されても、すべての読み取りはセカンダリではなくプライマリに送られます。

```
final ServerAddress serverAddress0 = new ServerAddress("cluster-endpoint", 27317));
    final MongoCredential credential = MongoCredential.createCredential("xxx",
            "admin", "xxxx".toCharArray());
    final MongoClientSettings settings = MongoClientSettings.builder()
            .credential(credential)
            .readPreference(ReadPreference.secondaryPreferred())
            .retryWrites(false)
            .applyToSslSettings(builder -> builder
                    .enabled(false))
            .applyToClusterSettings(builder -> builder.hosts(
                            Arrays.asList(serverAddress0
                            ))
                    .requiredReplicaSetName("rs0"))
            .build();
    MongoClient mongoClient = MongoClients.create(settings);
```

フェイルオーバーケース

上記のクライアント接続設定を使用すると、クラスターライターエンドポイントのフェイルオーバーが発生し、DNS レコードの更新が遅れた場合でも、クライアントは古いライター (フェイルオーバー後にリーダーになる) への書き込みを試みます。その結果、(プライマリではない) サーバー側のエラーが発生し、Java ドライバーでは適切に処理することができません (この問題についてはまだ調査中です)。そのため、たとえばアプリケーションサーバーが再起動されるまで、クライアントは不正な状態のままになる可能性があります。

これには次の 2 つの回避策があります。

- 接続文字列を使用して Amazon DocumentDB に接続するクライアントは、読み込み設定を設定するときに ClusterConnectionMode が MULTIPLE に設定されるため、この問題は発生しません。

```
MongoClientURI mongoClientURI = new MongoClientURI("mongodb://usr:pass:cluster-
endpoint:27317/test?ssl=false&replicaSet=rs0&readpreference=secondaryPreferred");
MongoClient mongoClient = MongoClients.create(mongoClientURI.getURI());
```

  または、applyConnectionString メソッドで MongoClientSettings ビルダーを使用することもできます。

```
final MongoClientSettings settings = MongoClientSettings.builder()
        .credential(credential)
        .applyConnectionString(new ConnectionString("usr:pass:cluster-endpoint:27317/
test?ssl=false&replicaSet=rs0&readpreference=secondaryPreferred"))
        .retryWrites(false)
        .applyToSslSettings(builder # builder
                .enabled(false))
        .build();
MongoClient mongoClient = MongoClients.create(settings);
```

- 明示的に ClusterConnectionMode を MULTIPLE に設定します。これは applyToClusterSettings と hosts().size() == 1 を使用する場合にのみ必要です。

```
final ServerAddress serverAddress0 = new ServerAddress("cluster-endpoint", 27317));
final MongoCredential credential = MongoCredential.createCredential("xxx","admin",
 "xxxx".toCharArray());
final MongoClientSettings settings = MongoClientSettings.builder()
    .credential(credential)
    .readPreference(ReadPreference.secondaryPreferred())
    .retryWrites(false)
    .applyToSslSettings(builder # builder
    .enabled(false))
    .applyToClusterSettings(builder # builder
                .hosts(Arrays.asList(serverAddress0))
                .requiredReplicaSetName("rs0"))
                .mode(ClusterConnectionMode.MULTIPLE))
    .build();
MongoClient mongoClient = MongoClients.create(settings);
```

Amazon DocumentDB インスタンスへの接続のテスト

一般的な Linux または Windows ツールを使用して、クラスターへの接続をテストできます。

Linux または UNIX のターミナルからは、次のように入力することで接続をテストします (`cluster-endpoint` と `port` をインスタンスのエンドポイントとポートに置き換えてください)。

```
nc -zv cluster-endpoint port
```

サンプルのオペレーションと戻り値の例を次に示します。

```
nc -zv docdbTest.d4c7nm7stsfc0.us-west-2.docdb.amazonaws.com 27017

Connection to docdbTest.d4c7nm7stsfc0.us-west-2.docdb.amazonaws.com 27017 port [tcp/*]
 succeeded!
```

無効なエンドポイントへの接続

Amazon DocumentDB クラスターに接続し、有効でないクラスターエンドポイントを使用すると、以下のようなエラーが表示されます。

```
mongo --ssl \
   --host sample-cluster.node.us-east-1.docdb.amazonaws.com:27017 \
   --sslCAFile global-bundle.pem \
   --username <user-name> \
   --password <password>
```

出力は次のようになります。

```
MongoDB shell version v3.6
connecting to: mongodb://sample-cluster.node.us-east-1.docdb.amazonaws.com:27017/
2018-11-14T17:21:18.516-0800 I NETWORK [thread1] getaddrinfo("sample-cluster.node.us-
east-1.docdb.amazonaws.com") failed:
nodename nor servname provided, or not known 2018-11-14T17:21:18.537-0800 E QUERY
 [thread1] Error: couldn't initialize
connection to host sample-cluster.node.us-east-1.docdb.amazonaws.com, address is
 invalid :
connect@src/mongo/shell/mongo.js:237:13@(connect):1:6
exception: connect failed
```

クラスターの有効なエンドポイントを取得するには、以下のコマンドを実行します。

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[Endpoint,Port]'
```

インスタンスの有効なエンドポイントを取得するには、以下のコマンドを実行します。

```
aws docdb describe-db-instances \
   --db-instance-identifier sample-instance \
   --query 'DBInstances[*].[Endpoint.Address,Endpoint.Port]'
```

詳細については、「Amazon DocumentDB エンドポイントについて」を参照してください。

接続数に影響するドライバー設定

クライアントドライバーを使用して Amazon DocumentDB クラスターに接続する場合は、maxPoolSize 設定パラメータを考慮することが重要です。maxPoolSize 設定は、クライアントドライバーが接続プールに維持できる接続の最大数を決定します。

インデックス作成のトラブルシューティング

次のトピックでは、インデックスまたはバックグラウンドインデックスの構築が失敗したときにどうすべきかを説明しています。

トピック

- インデックスの構築が失敗する
- バックグラウンドインデックスビルドのレイテンシーの問題と失敗

インデックスの構築が失敗する

Amazon DocumentDB は、インデックス作成プロセスの一部としてローカルストレージをインスタンス上で使用します。[FreeLocalStorage] CloudWatch メトリクス (CloudWatch -> Metrics -> DocDB -> Instance Metrics) を使用して、このディスクの使用状況をモニタリングできます。インデックスの構築がすべてのローカルディスクを消費して失敗すると、エラーが発生します。Amazon DocumentDB にデータを移行する場合、まずインデックスを作成してからデータを挿入することをお勧めします。移行戦略やインデックスの作成については、Amazon DocumentDB の

ドキュメントや「オフラインメソッドを使用して、MongoDB から Amazon DocumentDB に移行する」のブログで Amazon DocumentDB への移行 を参照してください。

既存のクラスターでインデックスを作成するときに、インデックスの構築に予想以上の時間がかかる場合、または失敗する場合、まずインスタンスをスケールアップし、インデックスが作成されたらスケールバックして戻すことが推奨されます。Amazon DocumentDB では、 AWS Management Console または を使用してインスタンスサイズを数分ですばやくスケーリングできます AWS CLI。詳細については、「インスタンスクラスの管理」を参照してください。秒単位のインスタンス料金によって、使用したリソースの秒数分のみを支払います。

バックグラウンドインデックスビルドのレイテンシーの問題と失敗

Amazon DocumentDB のバックグラウンドインデックスビルドは、インデックスビルドが開始される前に開始されたプライマリインスタンス上のすべてのクエリの実行が完了するまで開始されません。クエリが長時間実行されている場合、バックグラウンドインデックスビルドはクエリが終了するまでブロックされるため、完了までに予想以上に時間がかかることがあります。コレクションが空であるとしても同様です。

フォアグラウンドインデックスビルドは、同じブロッキング動作を示しません。代わりに、フォアグラウンドインデックスビルドは、インデックスのビルドが完了するまで、コレクションを排他的にロックします。したがって、空のコレクションにインデックスを作成し、長時間実行されるクエリでブロックされないようにするには、フォアグラウンドインデックスビルドを使用することをお勧めします。

> **Note**
>
> Amazon DocumentDB では、任意の時点でコレクションにおいてバックグラウンドインデックスの作成が 1 つのみ許可されます。バックグラウンドインデックスの作成中に `createIndex()` や `dropIndex()` などの DDL (データ定義言語) オペレーションが発生すると、バックグラウンドインデックスの作成は失敗します。

パフォーマンスとリソース使用率のトラブルシューティング

このセクションでは、Amazon DocumentDB デプロイにおける一般的な診断の問題に関する質問と解決策を示します。提供されている例では mongo シェルを使用し、個々のインスタンスに範囲を設定しています。インスタンスエンドポイントを見つけるには、「Amazon DocumentDB エンドポイントについて」を参照してください 。

トピック

- Mongo API を使用してコレクションに対して実行された挿入、更新、削除操作の数を確認する方法
- キャッシュのパフォーマンスを分析する方法
- 長時間実行されているクエリやブロックされているクエリを見つけて終了する方法
- クエリプランを参照してクエリを最適化する方法
- Elastic クラスターのクエリプランを確認する方法
- インスタンスで実行中のすべての操作をリストする方法
- クエリの進捗状況を知る方法
- システムの実行速度が突然遅くなった理由を判断する方法
- 1 つ以上のクラスターインスタンスで CPU 使用率が高くなる原因を特定する方法
- インスタンスで開いているカーソルを確認する方法
- 現在の Amazon DocumentDB エンジンのバージョンを確認する方法
- インデックスの使用状況を分析し、未使用のインデックスを特定する方法
- 欠落しているインデックスを特定する方法
- 便利なクエリの概要

Mongo API を使用してコレクションに対して実行された挿入、更新、削除操作の数を確認する方法

特定のコレクションに対して実行された挿入、更新、削除操作の数を表示するには、そのコレクションで以下のコマンドを実行します。

```
db.collection.stats()
```

このコマンドの出力は、opCounters フィールドの下に以下のように記述されています。

- numDocsIns: このコレクションに挿入されたドキュメントの数。これには、insert および insertMany コマンドを使用して挿入されたドキュメントと、アップサートによって挿入されたドキュメントが含まれます。
- numDocsUpd: このコレクションで更新されたドキュメントの数。これには、update および findAndModify コマンドを使用して更新されたドキュメントが含まれます。

- NumDocsDel: このコレクションから削除されたドキュメントの数。これには deleteOne、deleteMany、remove、および findAndModify コマンドを使用して削除されたドキュメントも含まれます。
- LastReset: これらのカウンタが最後にリセットされた時刻。このコマンドによって提供される統計は、クラスターを起動/停止したり、インスタンスをスケールアップ/スケールダウンしたりするとリセットされます。

以下に示しているのは、db.collection.stats() 実行時の出力例です。

```
{
    "ns" : "db.test",
    "count" : ...,
    "size" : ...,
    "avgObjSize" : ...,
    "storageSize" : ...,
    "capped" : false,
    "nindexes" : ...,
    "totalIndexSize" : ...,
    "indexSizes" : {
        "_id_" : ...,
        "x_1" : ...
    },
    "collScans" : ...,
    "idxScans" : ...,
    "opCounter" : {
        "numDocsIns" : ...,
        "numDocsUpd" : ...,
        "numDocsDel" : ...
    },
    "cacheStats" : {
        "collBlksHit" : ...,
        "collBlksRead" : ..,
        "collHitRatio" : ...,
        "idxBlksHit" : ...,
        "idxBlksRead" : ...,
        "idxHitRatio" : ...
    },
    "lastReset" : "2022-09-02 19:41:40.471473+00",
    "ok" : 1,
    "operationTime" : Timestamp(1662159707, 1)
}
```

この stats コマンドは、Mongo API を使用して挿入、更新、削除操作を行うコレクション固有のカウンターを表示する場合に使用してください。コレクション固有の操作カウンターを表示するもう 1 つの方法は、DML 監査を有効にすることです。Amazon DocumentDB と CloudWatch のモニタリング では、1 分間隔で発生した全コレクションの挿入、更新、削除操作の数を表示できます。

キャッシュのパフォーマンスを分析する方法

キャッシュのパフォーマンスを分析することで、データ取得の効率とシステムパフォーマンスを把握できます。分析は、キャッシュとディスクから読み取られるデータの量に基づいています。キャッシュのパフォーマンスを把握するために、キャッシュヒット (キャッシュから読み取られたデータ) とキャッシュミス (キャッシュに見つからず、ディスクから読み取られたデータ) の数に関するキャッシュ統計を提供します。特定のコレクションのキャッシュ統計は、そのコレクションで以下のコマンドを実行すると確認できます。

```
db.collection.stats()
```

このコマンドの出力の cacheStats フィールドの値は、コレクションのキャッシュ統計と、コレクション上に作成されたインデックスの合計キャッシュ統計を提供します。これらの統計を以下に示します。

- **collBlksHit**: このコレクションの操作中にキャッシュから読み取られたブロック数。
- **collBlksRead**: このコレクションの操作中にディスクから読み込まれた (キャッシュミス) ブロック数。
- **collHitRatio**: このコレクションのキャッシュヒット率 (`100 * [collBlksHit / (collBlksHit + collBlksRead)]`)。
- **idxBlksHit**: このコレクションで作成されたインデックスのキャッシュから読み取られたブロック数。
- **idxBlksRead**: このコレクションで作成されたインデックスのディスクから読み込まれた (キャッシュミス) ブロック数。
- idxHitRatio: このコレクションで作成されたインデックスのキャッシュヒット率 (`100 * [idxBlksHit / (idxBlksHit + idxBlksRead)]`)。
- **lastReset**: これらの統計が最後にリセットされた時刻。`db.collection.stats()` によって提供される統計は、クラスターを起動/停止したり、インスタンスをスケールアップ/スケールダウンしたりするとリセットされます。

各インデックスの idxBlksHit および idxBlksRead フィールドの内訳は、indexStats コマンドを使用して確認することもできます。インデックス固有のキャッシュ統計は、以下のコマンドを実行して参照できます。

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

各インデックスについて、以下のキャッシュ統計が cacheStats フィールドの下に表示されます。

- **blksHit**: このインデックスのキャッシュから読み取られるブロック数。
- **blksRead**: このインデックスのディスクから読み取られるブロック数。
- **blksHitRatio**: キャッシュヒット率は、100 * [blksHit / (blksHit + blksRead)] で計算され、小数点以下 4 桁に四捨五入されました。

長時間実行されているクエリやブロックされているクエリを見つけて終了する方法

ユーザークエリの実行は、最適でないクエリプランのために遅くなることも、リソースの競合のためにブロックされる可能性もあります。

クエリプランが最適でないために長時間実行されているクエリ、またはリソースの競合のためにブロックされているクエリを見つけるには、currentOp コマンドを使用します。コマンドをフィルタリングすると、終了する関連クエリのリストを絞り込むことができます。クエリを終了できるようにするには、長時間実行されているクエリに opid を関連付ける必要があります。

次のクエリでは、currentOp コマンドを使用して、ブロックされている、または 10 秒以上実行されているクエリをすべてリストします。

```
db.adminCommand({
    aggregate: 1,
    pipeline: [
        {$currentOp: {}},
        {$match:
            {$or: [
                {secs_running: {$gt: 10}},
                {WaitState: {$exists: true}}]}},
        {$project: {_id:0, opid: 1, secs_running: 1}}],
    cursor: {}
});
```

次に、クエリを絞り込み、10 秒以上実行されているクエリの opid を見つけて終了できます。

10 秒以上実行されているクエリを見つけ出して終了するには

1. クエリの opid を見つけます。

```
db.adminCommand({
    aggregate: 1,
    pipeline: [
        {$currentOp: {}},
        {$match:
            {$or:
                [{secs_running: {$gt: 10}},
                 {WaitState: {$exists: true}}]}}],
    cursor: {}
});
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "opid" : 24646,
                "secs_running" : 12
            }
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
    },
    "ok" : 1
}
```

2. killOp オペレーションを使用してクエリを終了します。

```
db.adminCommand({killOp: 1, op: 24646});
```

クエリプランを参照してクエリを最適化する方法

クエリの実行が遅い場合は、関連するドキュメントを選択するためにクエリの実行でコレクションのフルスキャンが必要である可能性があります。場合によっては、適切なインデックスの作成により、クエリの実行を高速化できます。このシナリオを検出し、インデックスを作成するフィールドを決定するには、explain コマンドを使用します。

> **Note**
>
> Amazon DocumentDB は分散型で耐障害性が高く、自己復旧ストレージシステムを利用する専用データベースエンジンで MongoDB 3.6 API をエミュレートしています。その結果、Amazon DocumentDB と MongoDB では、クエリプランと explain() の出力が異なる場合があります。クエリプランを制御する場合は、$hint 演算子を使用して優先インデックスの選択を強制できます。

次のように、explain コマンドで改善対象のクエリを実行します。

```
db.runCommand({explain: {<query document>}})
```

以下に示しているのは、オペレーションの例です。

```
db.runCommand({explain:{
    aggregate: "sample-document",
    pipeline: [{$match: {x: {$eq: 1}}}],
    cursor: {batchSize: 1}}
});
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "db.test",
        "winningPlan" : {
            "stage" : "COLLSCAN"
        }
    },
    "serverInfo" : {
        "host" : "...",
```

```
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}
```

前述の出力は、$match ステージでコレクション全体をスキャンし、各ドキュメントのフィールド "x" が 1 に等しいかどうか確認する必要があることを示しています。コレクションに多くのドキュメントがある場合、コレクションのスキャン (および全体的なクエリのパフォーマンス) は非常に低速になります。したがって、explain コマンドの出力に "COLLSCAN" が存在することは、適切なインデックスを作成することで、クエリのパフォーマンスを向上できることを示します。

この例では、クエリはすべてのドキュメントでフィールド "x" が 1 と等しいかどうかを確認します。したがって、フィールド "x" でインデックスを作成すると、クエリは完全なコレクションスキャンを回避し、インデックスを使用して関連ドキュメントをより早く返すことができます。

フィールド "x" でインデックスを作成した後、explain 出力は次のようになります。

```
{
    "queryPlanner" : {
         "plannerVersion" : 1,
         "namespace" : "db.test",
         "winningPlan" : {
             "stage" : "IXSCAN",
             "indexName" : "x_1",
             "direction" : "forward"
         }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}
```

フィールド "x" にインデックスを作成すると、$match ステージはインデックススキャンを使用して、述語 "x = 1" を評価する必要があるドキュメントの数を減らすことができます。

小規模なコレクションで、パフォーマンスの向上がごくわずかである場合、Amazon DocumentDB クエリプロセッサはインデックスを使用しないことがあります。

Elastic クラスターのクエリプランを確認する方法

Elastic クラスターのクエリプランを調べるには、explain コマンドを使用します。以下は、シャードコレクションを対象とする検索クエリの explain 操作の例です。

```
db.runCommand(
   {
     explain: { find: "cities", filter: {"name": "Seoul"}}
   }
)
```

> **Note**
>
> Amazon DocumentDB は、目的別データベースエンジンで MongoDB をエミュレートします。その結果、クエリプランと explain( ) の出力は、Amazon DocumentDB と MongoDB の間で異なる場合があります。$hint オペレーターを使ってクエリプランを制御し、優先インデックスの選択を強制することができます。

このオペレーションによる出力は、次のようになります (JSON 形式)。

```
{
  "queryPlanner" : {
    "elasticPlannerVersion" : 1,
    "winningPlan" : {
      "stage" : "SINGLE_SHARD",
      "shards" : [
        {
          "plannerVersion" : 1,
          "namespace" : "population.cities",
          "winningPlan" : {
            "stage" : "SHARD_MERGE",
            "shards" : [
              {
                "shardName" : "f2cf5cfd-fe9c-40ca-b4e5-298ca0d11111",
                "plannerVersion" : 1,
                "namespace" : "population.cities",
                "winningPlan" : {
                  "stage" : "PARTITION_MERGE",
                  "inputStages" : [
                    {
```

```
                    "stage" : "COLLSCAN",
                    "partitionCount" : 21
                  }
                ]
              }
            },
            {
              "shardName" : "8f3f80e2-f96c-446e-8e9d-aab8c7f22222",
              "plannerVersion" : 1,
              "namespace" : "population.cities",
              "winningPlan" : {
                "stage" : "PARTITION_MERGE",
                "inputStages" : [
                  {
                    "stage" : "COLLSCAN",
                    "partitionCount" : 21
                  }
                ]
              }
            },
            {
              "shardName" : "32c5a06f-1b2b-4af1-8849-d7c4a033333",
              "plannerVersion" : 1,
              "namespace" : "population.cities",
              "winningPlan" : {
                "stage" : "PARTITION_MERGE",
                "inputStages" : [
                  {
                    "stage" : "COLLSCAN",
                    "partitionCount" : 22
                  }
                ]
              }
            }
          ]
        },
        "shardName" : "32c5a06f-1b2b-4af1-8849-d7c4a0f3fb58"
      }
    ]
  }
},
"serverInfo" : {
  "host" : "example-4788267630.us-east-1.docdb-elastic.amazonaws.com:27017",
  "version" : "5.0.0"
```

```
  },
  "ok" : 1,
  "operationTime" : Timestamp(1695097923, 1)
}
```

上記の出力は、3 つのシャードクラスターでの find クエリのクエリプランを示しています。各シャードには複数のデータパーティションがあり、それぞれ異なる入力ステージを設定できます。この例では、各シャード内の「PARTITION_MERGE」ステージで結果がマージされる前に、すべてのパーティションで「COLLSCAN」(コレクションスキャン) が実行されます。その後、シャード全体の結果は「SHARD_MERGE」ステージでマージされ、その後クライアントに送り返されます。

インスタンスで実行中のすべての操作をリストする方法

ユーザーまたはプライマリユーザーが、診断およびトラブルシューティングのために、インスタンスで実行されている現在のすべてのオペレーションをリストアップしたい場合があります。(ユーザーの管理の詳細については、Amazon DocumentDB ユーザーの管理 を参照してください。)

mongo シェルで、次のクエリを使用して、Amazon DocumentDB インスタンスで実行中のすべてのオペレーションをリストできます。

```
db.adminCommand({currentOp: 1, $all: 1});
```

クエリは、インスタンス上で現在動作しているすべてのユーザークエリと内部システムタスクの完全なリストを返します。

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "inprog" : [
        {
            "desc" : "INTERNAL"
        },
        {
            "desc" : "TTLMonitor",
            "active" : false
        },
        {
            "client" : ...,
            "desc" : "Conn",
            "active" : true,
            "killPending" : false,
            "opid" : 195,
```

```
            "ns" : "admin.$cmd",
            "command" : {
                "currentOp" : 1,
                "$all" : 1
            },
            "op" : "command",
            "$db" : "admin",
            "secs_running" : 0,
            "microsecs_running" : NumberLong(68),
            "clientMetaData" : {
            "application" : {
                "name" : "MongoDB Shell"
            },
            "driver" : {
                ...
            },
            "os" : {
                ...
            }
          }
       },
       {
          "desc": "GARBAGE_COLLECTION",
          "garbageCollection": {
             "databaseName": "testdb",
             "collectionName": "testCollectionA"
          },
          "secs_running": 3,
          "microsecs_running": NumberLong(3123456)
       },
       {
          "desc": "GARBAGE_COLLECTION",
          "garbageCollection": {
             "databaseName": "testdb",
             "collectionName": "testCollectionB"
          },
          "secs_running": 4,
          "microsecs_running": NumberLong(4123456)
       }
    ],
    "ok" : 1
}
```

次に示すのは、"desc" フィールドの有効な値です。

- **INTERNAL** - カーソルのクリーンアップや古いユーザーのクリーンアップタスクなどの内部システムタスク。
- **TTLMonitor** - 有効期限 (TTL) モニタリングスレッド。その実行ステータスが "active" フィールドに反映されています。
- **GARBAGE_COLLECTION** - 内部ガベージコレクタースレッド。
- **CONN** - ユーザーのクエリ。
- **CURSOR** — 操作はユーザーが「getMore」コマンドを呼び出して次の結果のバッチを取得するのを待っているアイドルカーソルです。この状態では、カーソルはメモリを消費していますが、計算は消費されません。

前の出力では、システムで実行されているすべてのユーザークエリもリストされます。各ユーザークエリはデータベースとコレクションのコンテキストで実行され、これらの 2 つを統合したものが、名前空間と呼ばれます。各ユーザークエリの名前空間は、"ns" フィールドで利用できます。

特定の名前空間で実行されているすべてのユーザークエリをリストする必要がある場合があります。したがって、前の出力は "ns" フィールドでフィルタリングする必要があります。次にフィルター処理の出力を達成するためのクエリの例を示します。クエリは、データベース "db" とコレクション "test" (つまり、"db.test" 名前空間) で現在実行されているすべてのユーザークエリをリストします。

```
db.adminCommand({aggregate: 1,
    pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
               {$match: {ns: {$eq: "db.test"}}}],
    cursor: {}
});
```

システムのプライマリユーザーは、すべてのユーザーのクエリと、すべての内部システムタスクを確認できます。他のすべてのユーザーは、各自のクエリのみを表示できます。

クエリと内部システムタスクの合計数が、デフォルトのバッチカーソルサイズを超えている場合、mongo シェルは残りの結果を表示するイテレーターオブジェクト 'it' を自動的に生成します。すべての結果がなくなるまで、コマンド 'it' の実行を継続します。

クエリの進捗状況を知る方法

ユーザーのクエリは、最適とはいえないクエリプランが原因で実行速度が遅くなったり、リソースの競合が原因でブロックされたりすることがあります。このようなクエリのデバッグは複数ステップのプロセスであり、同じステップを複数回実行することが必要になる場合があります。

デバッグの最初のステップでは、長時間実行されているか、ブロックされているすべてのクエリをリストします。次のクエリは、10 秒以上実行されているか、リソースを待機しているすべてのユーザークエリをリストします。

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {}},
                            {$match: {$or: [{secs_running: {$gt: 10}},
                                            {WaitState: {$exists: true}}]}},
                            {$project: {_id:0,
                                        opid: 1,
                                        secs_running: 1,
                                        WaitState: 1,
                                        blockedOn: 1,
                                        command: 1}}],
                 cursor: {}
                });
```

前述のクエリを定期的に繰り返して、クエリのリストが変更されているかどうかを確認し、実行時間が長いクエリまたはブロックされているクエリを識別します。

対象のクエリの出力ドキュメントに WaitState フィールドがある場合、クエリの実行が低速であるかブロックされている理由は、リソースの競合であることを示しています。リソースの競合は、I/O、内部システムタスク、またはその他のユーザークエリが原因である可能性があります。

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "opid" : 201,
                "command" : {
                    "aggregate" : ...
                },
                "secs_running" : 208,
```

```
                "WaitState" : "IO"
            }
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
    },
    "ok" : 1
}
```

同じインスタンスで同時に実行されているさまざまなコレクションで多くのクエリが実行されているか、クエリを実行中のデータセットのインスタンスサイズが小さすぎると、I/O がボトルネックとなる場合があります。クエリが読み取り専用クエリである場合、各コレクションのクエリを別々のレプリカに分離することで、以前の状況を軽減できます。さまざまなコレクション間で同時更新を行う場合、またはデータセットに対するインスタンスサイズが小さすぎる場合の軽減策は、インスタンスをスケールアップすることです。

リソースの競合の原因が他のユーザークエリである場合、出力ドキュメントの "blockedOn" フィールドに、このクエリに影響しているクエリの "opid" が含まれます。"opid" を使用して、すべてのクエリの "WaitState" および "blockedOn" フィールドのチェーンに従って、チェーンの先頭にあるクエリを調べます。

チェーンの先頭にあるタスクが内部タスクである場合の唯一の緩和策は、クエリを終了し、後で再実行することです。

以下は、検索クエリが別のタスクによって所有されているコレクションロックでブロックされるサンプル出力です。

```
{
    "inprog" : [
        {
            "client" : "...",
            "desc" : "Conn",
            "active" : true,
            "killPending" : false,
            "opid" : 75,
            "ns" : "...",
            "command" : {
                "find" : "...",
                "filter" : {

                }
            },
```

```
            "op" : "query",
            "$db" : "test",
            "secs_running" : 9,
            "microsecs_running" : NumberLong(9449440),
            "threadId" : 24773,
            "clientMetaData" : {
                "application" : {
                   "name" : "MongoDB Shell"
                },
                "driver" : {
                    ...
                },
                "os" : {
                    ...
                }
            },
            "WaitState" : "CollectionLock",
            "blockedOn" : "INTERNAL"
        },
        {
            "desc" : "INTERNAL"
        },
        {
            "client" : "...",
            ...
            "command" : {
                "currentOp" : 1
            },
            ...
        }
    ],
    "ok" : 1
}
```

"WaitState" の値が "Latch"、"SystemLock"、"BufferLock"、"BackgroundActivity"、または "Other" の場合、リソース競合の原因は内部システムタスクにあります。この状態が長時間続く場合、唯一の緩和策は、クエリを終了し、後で再実行することです。

システムの実行速度が突然遅くなった理由を判断する方法

以下にシステムの速度が低下する一般的な理由を示します。

- 同時クエリ間の過剰なリソースの競合

- 時間の経過とともに増加するアクティブな同時クエリの数
- "GARBAGE_COLLECTION" などの内部システムタスク

時間の経過とともにシステムの使用状況をモニタリングするには、以下の "currentOp" クエリを定期的に実行し、外部ストアに結果を出力します。クエリは、システムの各名前空間内のクエリとオペレーションの数をカウントします。次に、システムの使用状況の結果を分析し、システムへの負荷を把握して適切なアクションを行うことができます。

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

このクエリは、各名前空間で実行されているすべてのクエリとすべての内部システムタスクの合計を返します。また、名前空間ごとに存在する場合は、待機状態の一意の数を返します。

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "_id" : {
                    "desc" : "Conn",
                    "ns" : "db.test",
                    "WaitState" : "CollectionLock"
                },
               "count" : 2
            },
            {
                "_id" : {
                    "desc" : "Conn",
                    "ns" : "admin.$cmd"
                },
                "count" : 1
            },
            {
                "_id" : {
```

```
                    "desc" : "TTLMonitor"
                },
                "count" : 1
            }
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
    },
    "ok" : 1
}
```

上記の出力では、コレクションロックでブロックされている名前空間 "db.test" に 2 つのユーザークエリ、名前空間 "admin.$cmd" に 1 つのクエリ、および 1 つの内部 "TTLMonitor" タスクがあります。

出力が、ブロック待機状態の多くのクエリを示す場合は、「長時間実行されているクエリやブロックされているクエリを見つけて終了する方法」を参照してください。

1 つ以上のクラスターインスタンスで CPU 使用率が高くなる原因を特定する方法

以下のセクションは、インスタンスの CPU 使用率が高い原因を特定するのに役立ちます。結果は、ワークロードによって異なります。

- インスタンスが突然遅くなる原因を特定するには、「システムの実行速度が突然遅くなった理由を判断する方法」を参照してください。
- 特定のインスタンスで長時間実行されるクエリを特定して終了するには、「長時間実行されているクエリやブロックされているクエリを見つけて終了する方法」を参照してください。
- クエリが進行中かどうかを理解するには、「クエリの進捗状況を知る方法」を参照してください。
- クエリの実行に時間がかかる理由を確認するには、「クエリプランを参照してクエリを最適化する方法」を参照してください。
- 実行時間が長いクエリを長期間にわたって追跡するには、「Amazon DocumentDB オペレーションのプロファイリング」を参照してください 。

インスタンスの CPU 使用率が高い理由に応じて、以下の 1 つ以上の操作を行うと役立ちます。

- プライマリインスタンスで高い CPU 使用率が示され、レプリカインスタンスではそうではない場合、クライアントの読み込み優先設定 (secondaryPreferred など) を使用してレプリカ間で

読み込みトラフィックを分散することを検討してください。詳細については、「レプリカセットとしての Amazon DocumentDB に接続する」を参照してください。

レプリカを読み込みに使用すると、プライマリインスタンスでより多くの書き込みトラフィックを処理できるようになり、クラスターのリソースをより有効に活用できます。レプリカからの読み込みには結果整合性があります。

- CPU 使用率が高いことが書き込みワークロードの結果である場合、クラスターのインスタンスのサイズをより大きいインスタンスタイプに変更すると、ワークロードに対応する CPU コアの数が増えます。詳細については、「インスタンス」および「インスタンスクラスの仕様」を参照してください。
- すべてのクラスターインスタンスで高い CPU 使用率が示され、ワークロードでレプリカが読み取りに使用されている場合、クラスターにレプリカを追加すると、読み取りトラフィックに使用できるリソースが増えます。詳細については、「クラスターへの Amazon DocumentDB インスタンスの追加」を参照してください。

インスタンスで開いているカーソルを確認する方法

Amazon DocumentDB インスタンスに接続している場合、コマンド `db.runCommand("listCursors")` を使用して、そのインスタンスで開いているカーソルを一覧表示できます。特定の Amazon DocumentDB インスタンスで特定の時間に開くことができる 4,560 個までのアクティブカーソルの数は 4,560 個に制限されています。インスタンスタイプによって異なります。通常、カーソルはインスタンスのリソースを消費し、開いている数に上限があるため、不要になったカーソルは閉じることをお勧めします。特定の制限については「Amazon DocumentDB のクォータと制限事項」を参照してください。

```
db.runCommand("listCursors")
```

現在の Amazon DocumentDB エンジンのバージョンを確認する方法

現在の Amazon DocumentDB エンジンのバージョンを確認するには、次のコマンドを実行します。

```
db.runCommand({getEngineVersion: 1})
```

このオペレーションによる出力は、次のようになります (JSON 形式) 。

```
{ "engineVersion" : "2.x.x", "ok" : 1 }
```

> **Note**
>
> Amazon DocumentDB 3.6 のエンジンバージョンは 1.x.x、Amazon DocumentDB 4.0 のエンジンバージョンは2.x.x です。

インデックスの使用状況を分析し、未使用のインデックスを特定する方法

特定のコレクションのインデックスを識別するには、次のコマンドを実行します。

```
db.collection.getIndexes()
```

コレクションに対して実行された操作中にどれだけのインデックスが使用されているかを分析するには、collStats および indexStats コマンドを使用できます。インデックスを使用して実行されたスキャン (インデックススキャン) の総数とインデックスなしで実行されたスキャンの数 (コレクションスキャン) を比較するには、次のコマンドを実行します。

```
db.collection.stats()
```

このコマンドの出力には、以下の値が含まれます。

- **idxScans**: インデックスを使用してこのコレクションに対して実行されたスキャンの数。
- **collScans**: インデックスを使用せずにこのコレクションに対して実行されたスキャンの数。これらのスキャンでは、コレクション内のドキュメントを一度に 1 つずつ調べる必要がありました。
- **lastReset**: これらのカウンタが最後にリセットされた時刻。このコマンドによって提供される統計は、クラスターを起動/停止したり、インスタンスをスケールアップ/スケールダウンしたりするとリセットされます。

各インデックスの使用量の内訳は、次のコマンドの出力で参照できます。パフォーマンスを向上させ、コストを削減するために、ベストプラクティスとして、未使用のインデックスを定期的に識別して削除することをお勧めします。インデックスを維持するために不必要なコンピューティング、ストレージ、I/O が使用されなくなるためです。

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

このコマンドの出力では、コレクションで作成された各インデックスについて次の値が表示されます。

- **ops** - インデックスを使用したオペレーションの数。ワークロードが十分に長時間実行されており、ワークロードが安定していることがはっきりわかる場合、ops 値 0 は、インデックスがまったく使用されていないことを示しています。
- **numDocsRead**: このインデックスを使用する操作中に読み取られたドキュメントの数。
- **since** - インデックス使用状況に関する統計情報の収集が Amazon DocumentDB により開始されてからの時間。通常は、前回のデータベース再起動またはメンテナンスアクション以降の値です。
- **size** - このインデックスのサイズをバイト数で表す。

次の例は、上記のコマンドを実行したときの出力例です。

```
{
    "name" : "_id_",
    "key" : {
        "_id" : 1
    },
    "host" : "example-host.com:12345",
    "size" : NumberLong(...),
    "accesses" : {
        "ops" : NumberLong(...),
        "docsRead" : NumberLong(...),
        "since" : ISODate("...")
    },
    "cacheStats" : {
        "blksRead" : NumberLong(...),
        "blksHit" : NumberLong(...),
        "hitRatio" : ...
    }
}
{
    "name" : "x_1",
    "key" : {
        "x" : 1
    },
    "host" : "example-host.com:12345",
    "size" : NumberLong(...),
    "accesses" : {
        "ops" : NumberLong(...),
        "docsRead" : NumberLong(...),
        "since" : ISODate("...")
    },
    "cacheStats" : {
```

```
        "blksRead" : NumberLong(...),
        "blksHit" : NumberLong(...),
        "hitRatio" : ...
    }
}
```

コレクションの全体的なインデックスサイズを確認するには、次のコマンドを実行します。

```
db.collection.stats()
```

未使用のインデックスを削除するには、次のコマンドを実行します。

```
db.collection.dropIndex("indexName")
```

欠落しているインデックスを特定する方法

Amazon DocumentDB プロファイラを使用して、低速なクエリをログに記録 できます。低速なクエリのログに繰り返し現れるクエリがある場合、そのクエリのパフォーマンスを向上させるために追加のインデックスが必要であることを示している可能性があります。

少なくとも 1 つのCOLLSCANステージを実行する 1 つ以上のステージを持つ長時間実行されるクエリを探すことで、有用なインデックスの機会を特定できます。つまり、クエリステージは、クエリに応答を提供するためにコレクション内のすべてのドキュメントを読み取る必要があります。

次の例は、大規模なコレクションで実行されたタクシー乗車のコレクションに対するクエリを示しています。

```
db.rides.count({"fare.totalAmount":{$gt:10.0}}))
```

この例を実行するには、fare.totalAmount フィールドにインデックスがないため、クエリはコレクションスキャン (つまり、コレクション内のすべてのドキュメントを読み込む) を実行する必要がありました。このクエリの Amazon DocumentDB プロファイラからの出力は、次のようになります。

```
{
    ...
    "cursorExhausted": true,
    "nreturned": 0,
```

```
    "responseLength": 0,
    "protocol": "op_query",
    "millis": 300679,
    "planSummary": "COLLSCAN",
    "execStats": {
        "stage": "COLLSCAN",
        "nReturned": "0",
        "executionTimeMillisEstimate": "300678.042"
    },
    "client": "172.31.5.63:53878",
    "appName": "MongoDB Shell",
    "user": "example"
}
```

この例のクエリを高速化するには、以下に示すように fare.totalAmount にインデックスを作成します。

```
db.rides.createIndex( {"fare.totalAmount": 1}, {background: true} )
```

> **Note**
>
> フォアグラウンドで作成されたインデックス (つまり、インデックスの作成時に {background:true} オプションが指定されなかった場合) は、排他的な書き込みロックを使用します。これにより、インデックス構築が完了するまでアプリケーションがコレクションにデータを書き込めなくなります。本番稼働クラスターでインデックスを作成する場合は、このような影響が生じる可能性がある点に注意してください。インデックスを作成するときは、{background:true} を設定することをお勧めします。

一般に、カーディナリティが高いフィールド (たとえば、多数の一意の値) でインデックスを作成します。カーディナリティの低いフィールドにインデックスを作成すると、使用されないインデックスが大きくなることがあります。Amazon DocumentDB クエリオプティマイザは、クエリプランの作成時に、コレクション全体のサイズとインデックスの選択性を考慮します。インデックスが存在する場合でも、クエリプロセッサが COLLSCAN を選択することがあります。これは、インデックスを利用しても、コレクション全体をスキャンした場合よりパフォーマンスが高くならないとクエリプロセッサが推定したときに発生します。特定のインデックスの利用をクエリプロセッサに強制する場合、以下に示すように、hint() 演算子を使用できます。

```
db.collection.find().hint("indexName")
```

便利なクエリの概要

以下のクエリは、Amazon DocumentDB のパフォーマンスとリソース使用率をモニタリングするのに役立ちます 。

- 以下のコマンドを使用して、オペレーションカウンター、キャッシュ統計、アクセス統計、サイズ統計など、特定のコレクションに関する統計を表示します。

```
db.collection.stats()
```

- 以下のコマンドを使用して、コレクション上に作成された各インデックスに関する統計情報 (インデックスのサイズ、インデックス固有のキャッシュ統計、インデックス使用統計など) を表示します。

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

- 次のクエリを使用して、すべてのアクティビティをリストします。

```
db.adminCommand({currentOp: 1, $all: 1});
```

- 次のコードは、実行時間が長いクエリまたはブロックされたすべてのクエリをリストします。

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {}},
                            {$match: {$or: [{secs_running: {$gt: 10}},
                                            {WaitState: {$exists: true}}]}},
                            {$project: {_id:0,
                                        opid: 1,
                                        secs_running: 1,
                                        WaitState: 1,
                                        blockedOn: 1,
                                        command: 1}}],
                 cursor: {}
                });
```

- 次のコードは、クエリを終了します。

```
db.adminCommand({killOp: 1, op: <opid of running or blocked query>});
```

- 次のコードを使用して、システム状態の集約ビューを取得します。

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

Amazon DocumentDB クラスター、インスタンス、およびリソース管理 API リファレンス

このセクションでは、HTTP、 ()、または AWS SDK を介してアクセス可能な Amazon DocumentDB (MongoDB 互換 AWS Command Line Interface AWS CLI) のクラスター、インスタンス、およびリソース管理オペレーションについて説明します。これらの API を使用して、クラスターおよびインスタンスの作成、削除、および変更を行うことができます。

> **Important**
>
> これらの API は、クラスター、インスタンス、および関連リソースの管理にのみ使用されます。稼働中の Amazon DocumentDB クラスターに接続する方法については 入門ガイド を参照してください。

トピック

- アクション
- データ型
- 共通エラー
- 共通パラメータ

アクション

以下のアクションは でサポートされています Amazon DocumentDB (with MongoDB compatibility)。

- AddSourceIdentifierToSubscription
- AddTagsToResource
- ApplyPendingMaintenanceAction
- CopyDBClusterParameterGroup
- CopyDBClusterSnapshot
- CreateDBCluster
- CreateDBClusterParameterGroup
- CreateDBClusterSnapshot

- CreateDBInstance
- CreateDBSubnetGroup
- CreateEventSubscription
- CreateGlobalCluster
- DeleteDBCluster
- DeleteDBClusterParameterGroup
- DeleteDBClusterSnapshot
- DeleteDBInstance
- DeleteDBSubnetGroup
- DeleteEventSubscription
- DeleteGlobalCluster
- DescribeCertificates
- DescribeDBClusterParameterGroups
- DescribeDBClusterParameters
- DescribeDBClusters
- DescribeDBClusterSnapshotAttributes
- DescribeDBClusterSnapshots
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeDBSubnetGroups
- DescribeEngineDefaultClusterParameters
- DescribeEventCategories
- DescribeEvents
- DescribeEventSubscriptions
- DescribeGlobalClusters
- DescribeOrderableDBInstanceOptions
- DescribePendingMaintenanceActions
- FailoverDBCluster
- FailoverGlobalCluster
- ListTagsForResource

- ModifyDBCluster
- ModifyDBClusterParameterGroup
- ModifyDBClusterSnapshotAttribute
- ModifyDBInstance
- ModifyDBSubnetGroup
- ModifyEventSubscription
- ModifyGlobalCluster
- RebootDBInstance
- RemoveFromGlobalCluster
- RemoveSourceIdentifierFromSubscription
- RemoveTagsFromResource
- ResetDBClusterParameterGroup
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime
- StartDBCluster
- StopDBCluster
- SwitchoverGlobalCluster

Amazon DocumentDB Elastic クラスターでは、以下のアクションがサポートされています。

- ApplyPendingMaintenanceAction
- CopyClusterSnapshot
- CreateCluster
- CreateClusterSnapshot
- DeleteCluster
- DeleteClusterSnapshot
- GetCluster
- GetClusterSnapshot
- GetPendingMaintenanceAction
- ListClusters
- ListClusterSnapshots

- ListPendingMaintenanceActions
- ListTagsForResource
- RestoreClusterFromSnapshot
- StartCluster
- StopCluster
- TagResource
- UntagResource
- UpdateCluster

Amazon DocumentDB (with MongoDB compatibility)

以下のアクションは でサポートされています Amazon DocumentDB (with MongoDB compatibility)。

- AddSourceIdentifierToSubscription
- AddTagsToResource
- ApplyPendingMaintenanceAction
- CopyDBClusterParameterGroup
- CopyDBClusterSnapshot
- CreateDBCluster
- CreateDBClusterParameterGroup
- CreateDBClusterSnapshot
- CreateDBInstance
- CreateDBSubnetGroup
- CreateEventSubscription
- CreateGlobalCluster
- DeleteDBCluster
- DeleteDBClusterParameterGroup
- DeleteDBClusterSnapshot
- DeleteDBInstance
- DeleteDBSubnetGroup
- DeleteEventSubscription
- DeleteGlobalCluster

- DescribeCertificates
- DescribeDBClusterParameterGroups
- DescribeDBClusterParameters
- DescribeDBClusters
- DescribeDBClusterSnapshotAttributes
- DescribeDBClusterSnapshots
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeDBSubnetGroups
- DescribeEngineDefaultClusterParameters
- DescribeEventCategories
- DescribeEvents
- DescribeEventSubscriptions
- DescribeGlobalClusters
- DescribeOrderableDBInstanceOptions
- DescribePendingMaintenanceActions
- FailoverDBCluster
- FailoverGlobalCluster
- ListTagsForResource
- ModifyDBCluster
- ModifyDBClusterParameterGroup
- ModifyDBClusterSnapshotAttribute
- ModifyDBInstance
- ModifyDBSubnetGroup
- ModifyEventSubscription
- ModifyGlobalCluster
- RebootDBInstance
- RemoveFromGlobalCluster
- RemoveSourceIdentifierFromSubscription
- RemoveTagsFromResource

- ResetDBClusterParameterGroup
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime
- StartDBCluster
- StopDBCluster
- SwitchoverGlobalCluster

AddSourceIdentifierToSubscription

サービス : Amazon DocumentDB (with MongoDB compatibility)

既存のイベント通知サブスクリプションにソース識別子を追加します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

SourceIdentifier

追加されるイベントソースの識別子。

- ソースタイプがインスタンスの場合は、DBInstanceIdentifier を指定する必要があります。
- ソースタイプがセキュリティグループである場合は、DBSecurityGroupName を指定する必要があります。
- ソースタイプがパラメータグループである場合は、DBParameterGroupName を指定する必要があります。
- ソースタイプが DB ナップショットである場合は、DBSnapshotIdentifier を指定する必要があります。

タイプ: 文字列

必須: はい

SubscriptionName

ソース識別子を追加する Amazon DocumentDB イベント通知サブスクリプション名。

タイプ: 文字列

必須: はい

レスポンス要素

以下の要素がサービスによって返されます。

EventSubscription

サブスクライブしたイベントの詳細情報。

型: EventSubscription オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

SourceNotFound

リクエストされたリソースが見つかりませんでした。

HTTP ステータスコード: 404

SubscriptionNotFound

サブスクリプション名が存在しません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

AddTagsToResource

サービス : Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB リソースにメタデータタグを追加します。これらのタグをコスト配分レポートで使用して、Amazon DocumentDB リソースに関連付けられているコストを追跡したり、Amazon DocumentDB の AWS Identity and Access Management Amazon DocumentDBポリシーのConditionステートメントで追跡したりできます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

ResourceName

タグが追加される Amazon DocumentDB リソース。この値は Amazon リソースネーム (ARN) です。

タイプ: 文字列

必須: はい

Tags.Tag.N

Amazon DocumentDB リソースに割り当てられるタグ。

型: Tag オブジェクトの配列

必須: はい

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

DBInstanceNotFound

`DBInstanceIdentifier` は既存のインスタンスを参照していません。

HTTP ステータスコード: 404

DBSnapshotNotFound

`DBSnapshotIdentifier` は既存のスナップショットを参照していません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ApplyPendingMaintenanceAction

サービス : Amazon DocumentDB (with MongoDB compatibility)

保留中のメンテナンスアクションをリソース (例えば Amazon DocumentDB) に適用します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

ApplyAction

このリソースに適用する保留中のメンテナンスアクション。

有効な値: `system-update`、`db-upgrade`

タイプ: 文字列

必須: はい

OptInType

オプトインリクエストのタイプを指定するか、オプトインクエストを元に戻す値。タイプ `immediate` のオプトインリクエストは元に戻すことができません。

有効な値:

- `immediate` - メンテナンスアクションをすぐに適用します。
- `next-maintenance` - リソースの次のメンテナンスウィンドウ中にメンテナンスアクションを適用します。
- `undo-opt-in` - 既存の `next-maintenance` オプトインリクエストをキャンセルします。

タイプ: 文字列

必須: はい

ResourceIdentifier

保留中のアクションが適用されるリソースの Amazon リソースネーム (ARN)。

タイプ: 文字列

必須: はい

レスポンス要素

以下の要素がサービスによって返されます。

ResourcePendingMaintenanceActions

ApplyPendingMaintenanceAction の出力を表します。

型: ResourcePendingMaintenanceActions オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

InvalidDBInstanceState

指定されたインスタンスは 利用可能な 状態ではありません。

HTTP ステータスコード: 400

ResourceNotFoundFault

指定されたリソース ID は見つかりませんでした。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CopyDBClusterParameterGroup

サービス : Amazon DocumentDB (with MongoDB compatibility)

指定されたクラスターパラメータグループをコピーします。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

SourceDBClusterParameterGroupIdentifier

ソースクラスターパラメータグループの識別子または Amazon リソースネーム (ARN)。

制約:

- 有効なクラスターグループを指定する必要があります。
- ソースクラスターパラメータグループがコピー AWS リージョン と同じ にある場合は、 などの有効なパラメータグループ識別子`my-db-cluster-param-group`、または有効な ARN を指定します。
- ソースパラメータグループがコピー AWS リージョン とは異なる にある場合は、有効なクラスターパラメータグループ ARN を指定します。たとえば、 です`arn:aws:rds:us-east-1:123456789012:sample-cluster:sample-parameter-group`。

タイプ: 文字列

必須: はい

TargetDBClusterParameterGroupDescription

コピーされたクラスターパラメータグループの説明。

タイプ: 文字列

必須: はい

TargetDBClusterParameterGroupIdentifier

コピーされたクラスターパラメータグループの識別子。

制約:

- null、空、または空白にすることはできません。

- 1 ~ 255 の文字、数字またはハイフンを使用する必要があります。
- 1 字目は文字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

例: `my-cluster-param-group1`

タイプ: 文字列

必須: はい

Tags.Tag.N

タグはパラメータグループに割り当てられます。

タイプ: Tag オブジェクトの配列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBClusterParameterGroup

クラスターパラメータグループに関する詳細情報。

型: DBClusterParameterGroup オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBParameterGroupAlreadyExists

同じ名前のパラメータグループがすでに存在します。

HTTP ステータスコード: 400

DBParameterGroupNotFound

`DBParameterGroupName` は既存のパラメータグループを参照していません。

HTTP ステータスコード: 404

DBParameterGroupQuotaExceeded

このリクエストにより、パラメータグループの許容数を超過します。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CopyDBClusterSnapshot

サービス : Amazon DocumentDB (with MongoDB compatibility)

クラスターのスナップショットをコピーします。

手動の共有クラススナップショットからクラススナップショットをコピーするには、`SourceDBClusterSnapshotIdentifier` が共有クラススナップショットの Amazon リソースネーム (ARN) でなければなりません 共有 DB クラススナップショットは、暗号化されているかどうかにかかわらず、同じ AWS リージョン内でコピーのみ行えます。

コピー操作の進行後にキャンセルするには、クラススナップショットが [copying] ステータスの間に `TargetDBClusterSnapshotIdentifier` で識別されるターゲットクラススナップショットを削除します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

SourceDBClusterSnapshotIdentifier

コピーするクラススナップショットの識別子。このパラメータは大文字と小文字が区別されません。

制約:

- [available (使用可能)] 状態の有効なシステムスナップショットを指定する必要があります。
- ソーススナップショットがコピー AWS リージョン と同じ にある場合は、有効なスナップショット識別子を指定します。
- ソーススナップショットがコピー AWS リージョン とは異なる にある場合は、有効なクラススナップショット ARN を指定します。

例: `my-cluster-snapshot1`

タイプ: 文字列

必須: はい

TargetDBClusterSnapshotIdentifier

ソースクラススナップショットから作成する新しいクラススナップショットの識別子。このパラメータは大文字と小文字が区別されません。

制約:

- 1 ~ 63 の文字、数字またはハイフンを使用する必要があります。
- 1 字目は文字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

例: `my-cluster-snapshot2`

タイプ: 文字列

必須: はい

CopyTags

ソースクラススナップショットからターゲットクラススナップショットにすべてのタグをコピーする場合は `true` に、それ以外の場合は `false` に設定します。デフォルトは `false` です。

型: ブール値

必須: いいえ

KmsKeyId

暗号化されたクラススナップショットの AWS KMS キー ID。キー ID AWS KMS は、Amazon リソースネーム (ARN)、 AWS KMS キー識別子、または AWS KMS 暗号化 AWS KMS キーのキーエイリアスです。

から暗号化されたクラススナップショットをコピーする場合 AWS アカウント、 の値を指定`KmsKeyId`して、新しい暗号化キーでコピーを AWS KMS 暗号化できます。の値を指定しない場合`KmsKeyId`、クラススナップショットのコピーはソースクラススナップショットと同じ AWS KMS キーで暗号化されます。

別の から共有されている暗号化されたクラススナップショットをコピーする場合は AWS アカウント、 の値を指定する必要があります`KmsKeyId`。

暗号化されたクラススナップショットを別の にコピーするには AWS リージョン、 を送信先リージョンのクラススナップショットのコピーを暗号化するために使用する AWS KMS キー ID `KmsKeyId`に設定します。 AWS KMS 暗号化キーは、作成された AWS リージョン に固有であり、 AWS リージョン 別の の暗号化キーを使用することはできません AWS リージョン。

暗号化されていないクラススナップショットをコピーして `KmsKeyId` パラメータに値を指定すると、エラーが返されます。

タイプ: 文字列

必須: いいえ

PreSignedUrl

コピーするソースクラスタースナップショットを含む の API アクションの署名バージョン 4 の署名付きリクエストCopyDBClusterSnapshot AWS リージョン を含む URL。クラスタースナップショットを別の AWS リージョンからコピーする場合は、PreSignedUrl パラメータを使用する必要があります。

AWS SDK ツールまたは を使用している場合は AWS CLI、 PreSignedUrl を手動で指定する代わりに SourceRegion (--source-regionの場合は AWS CLI) を指定できます。SourceRegion を指定すると、ソース AWS リージョンで実行できる操作の有効なリクエストである署名付き URL が自動生成されます。

署名付き URL は、コピーするクラスタースナップショット AWS リージョン を含むソースで実行できる CopyDBClusterSnapshot API アクションの有効なリクエストである必要があります。署名付き URL リクエストでは、以下のパラメータ値を指定する必要があります。

- SourceRegion - コピーするスナップショットを含むリージョンの ID。
- SourceDBClusterSnapshotIdentifier - コピーする暗号化されたクラスタースナップショットの識別子。この識別子は、ソース AWS リージョンの Amazon リソースネーム (ARN) 形式であることが必要です。例えば、us-east-1 AWS リージョンから暗号化されたクラスタスナップショットをコピーする場合、SourceDBClusterSnapshotIdentifier は arn:aws:rds:us-east-1:12345678012:sample-cluster:sample-cluster-snapshot のようになります。
- TargetDBClusterSnapshotIdentifier - 作成する新しい暗号化されたクラスタースナップショットの識別子。このパラメータでは大文字と小文字は区別されません。

タイプ: 文字列

必須: いいえ

Tags.Tag.N

クラスタースナップショットに割り当てられるタグ。

タイプ: Tag オブジェクトの配列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBClusterSnapshot

クラススナップショットに関する詳細情報。

型: DBClusterSnapshot オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterSnapshotAlreadyExistsFault

指定された識別子を持つクラススナップショットをすでに持っています。

HTTP ステータスコード: 400

DBClusterSnapshotNotFoundFault

DBClusterSnapshotIdentifier は既存のクラススナップショットを参照していません。

HTTP ステータスコード: 404

InvalidDBClusterSnapshotStateFault

指定された値は有効なクラススナップショット状態ではありません。

HTTP ステータスコード: 400

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

KMSKeyNotAccessibleFault

AWS KMS キーへのアクセス中にエラーが発生しました。

HTTP ステータスコード: 400

SnapshotQuotaExceeded

リクエストにより、スナップショットの許容数を超過します。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateDBCluster

サービス : Amazon DocumentDB (with MongoDB compatibility)

新しい Amazon DocumentDB クラスターを作成します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterIdentifier

クラスター識別子。このパラメータは小文字で保存されます。

制約:

- 1 ~ 63 の文字、数字またはハイフンを使用する必要があります。
- 1 字目は文字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

例: `my-cluster`

タイプ: 文字列

必須: はい

Engine

このクラスターに使用されるデータベースエンジン名。

有効な値: `docdb`

タイプ: 文字列

必須: はい

AvailabilityZones.AvailabilityZone.N

クラスター内のインスタンスを作成できる Amazon EC2 アベイラビリティーゾーンのリスト。

型: 文字列の配列

必須: いいえ

BackupRetentionPeriod

自動バックアップを保管する日数。最小値 1 を指定しなければなりません。

デフォルト: 1

制約:

- 1 ~ 35 の値にする必要があります。

タイプ: 整数

必須: いいえ

DBClusterParameterGroupName

このクラスターに関連付けるクラスターパラメータグループの名前。

タイプ: 文字列

必須: いいえ

DBSubnetGroupName

このクラスターに関連付けるサブネットグループ。

制約: 既存の DBSubnetGroup の名前と一致する必要があります。デフォルト値を使用することはできません。

例: `mySubnetgroup`

タイプ: 文字列

必須: いいえ

DeletionProtection

このクラスターを削除できるかどうかを指定します。DeletionProtection を有効にすると、クラスターが変更され DeletionProtection が無効にならない限り、クラスターを削除することができません。DeletionProtection はクラスターが誤って削除されるのを防ぎます。

型: ブール値

必須: いいえ

EnableCloudwatchLogsExports.member.N

Amazon CloudWatch Logs にエクスポートするために有効にする必要があるログタイプのリスト。監査ログまたはプロファイラーログを有効にすることができます。詳細については、「Amazon DocumentDB 監査イベント」と「Amazon DocumentDB オペレーションのプロファイリング」を参照してください。詳細については、「Amazon DocumentDB イベント監査」と「Amazon DocumentDB 操作プロファイリング」を参照してください。

型: 文字列の配列

必須: いいえ

EngineVersion

使用するデータベースエンジンのバージョン番号。`--engine-version` は、デフォルトで最新のメジャーエンジンバージョンになります。本稼働ワークロードの場合、目的のメジャーエンジンのバージョンで、このパラメータを明示的に宣言することを推奨します。

タイプ: 文字列

必須: いいえ

GlobalClusterIdentifier

新しいグローバルクラスターのクラスター識別子。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

必須: いいえ

KmsKeyId

暗号化されたクラスターの AWS KMS キー識別子。

AWS KMS キー識別子は、 AWS KMS 暗号化キーの Amazon リソースネーム (ARN) です。新しいクラスターの暗号化に使用される AWS KMS 暗号化キーを所有 AWS アカウント する と同じを使用してクラスターを作成する場合は、 AWS KMS 暗号化 AWS KMS キーの ARN の代わりにキーエイリアスを使用できます。

`KmsKeyId` で暗号化キーが指定されていない場合:

- `StorageEncrypted` パラメータが `true` の場合、Amazon DocumentDB はデフォルトの暗号化キーを使用します。

AWS KMS は、 のデフォルトの暗号化キーを作成します AWS アカウント。には、それぞれ異なるデフォルトの暗号化キー AWS アカウント があります AWS リージョン。

タイプ: 文字列

必須: いいえ

ManageMasterUserPassword

Amazon Web Services Secrets Manager でマスターユーザーパスワードを管理するかどうかを指定します。

制約: `MasterUserPassword`が指定されている場合、Amazon Web Services Secrets Manager でマスターユーザーパスワードを管理することはできません。

型: ブール値

必須: いいえ

MasterUsername

クラスターのマスターユーザーの名前。

制約:

- 1 ~ 63 文字の英数字である必要があります。
- 1 字目は文字である必要があります。
- 選択したデータベースエンジンの予約語は使用できません。

タイプ: 文字列

必須: いいえ

MasterUserPassword

マスターデータベースユーザーのパスワードです。このパスワードをには、スラッシュ (/)、二重引用符 (")、または「at」記号 (@) 以外の印刷可能な ASCII 文字を含むことができます。

制約: 8 ~ 100 文字の長さの英数字である必要があります。

タイプ: 文字列

必須: いいえ

MasterUserSecretKmsKeyId

Amazon Web Services Secrets Manager で自動的に生成および管理されるシークレットを暗号化するための Amazon Web Services KMS キー識別子。この設定は、マスターユーザーパスワードが DB クラスターの Amazon Web Services Secrets Manager で Amazon DocumentDB によって管理されている場合にのみ有効です。

Amazon Web Services KMS キー識別子は、KMS キーのキー ARN、キー ID、エイリアス ARN、またはエイリアス名です。別の Amazon Web Services アカウントで KMS キーを使用するには、キー ARN またはエイリアス ARN を指定します。

を指定しない場合`MasterUserSecretKmsKeyId`、KMS `aws/secretsmanager` キーを使用してシークレットを暗号化します。シークレットが別の Amazon Web Services アカウントにある場合、KMS `aws/secretsmanager` キーを使用してシークレットを暗号化することはできません。また、カスタマーマネージド KMS キーを使用する必要があります。

Amazon Web Services アカウントにはデフォルトの KMS キーがあります。Amazon Web Services アカウントには、Amazon Web Services リージョンごとに異なるデフォルト KMS キーがあります。

タイプ: 文字列

必須: いいえ

Port

クラスターのインスタンスが接続を受け付けることができるポート番号。

タイプ: 整数

必須: いいえ

PreferredBackupWindow

`BackupRetentionPeriod` パラメータを使用して自動バックアップが有効になっている場合に、自動バックアップが作成される毎日の時間範囲。

デフォルトは、それぞれ 8 時間の時間ブロックからランダムに選択された 30 分のウィンドウです AWS リージョン。

制約:

- `hh24:mi-hh24:mi` の形式である必要があります。

- 時間は協定世界時 (UTC) である必要があります。
- 必要なメンテナンス期間と競合してはいけません。
- 少なくとも 30 分以上必要です。

タイプ: 文字列

必須: いいえ

PreferredMaintenanceWindow

週 1 回のシステムメンテナンスを実行できる時間帯 (世界標準時 (UTC))。

形式: `ddd:hh24:mi-ddd:hh24:mi`

デフォルトは、それぞれ 8 時間の時間ブロックからランダムに選択された 30 分のウィンドウであり AWS リージョン、ランダムな曜日に発生します。

有効な曜日: 月、火、水、木、金、土、日

制約: 最小限の 30 分単位のウィンドウ。

タイプ: 文字列

必須: いいえ

PreSignedUrl

現在サポートされていません。

タイプ: 文字列

必須: いいえ

ServerlessV2ScalingConfiguration

Amazon DocumentDB Serverless クラスターのスケーリング設定が含まれます。

型: ServerlessV2ScalingConfiguration オブジェクト

必須: いいえ

StorageEncrypted

クラスターが暗号化されているかどうかを指定します。

型: ブール値

必須: いいえ

StorageType

DB クラスターに関連付けるストレージタイプ。

Amazon DocumentDB クラスターのストレージタイプの詳細については、「Amazon DocumentDB 開発者ガイド」の「クラスターストレージ設定」を参照してください。

ストレージタイプの有効な値 - `standard | iopt1`

デフォルト値は `standard` です

> **Note**
>
> ストレージタイプを に設定して Amazon DocumentDB クラスターを作成すると`iopt1`、ストレージタイプがレスポンスで返されます。この時 `standard` に設定すると、ストレージタイプは返されません。

タイプ: 文字列

必須: いいえ

Tags.Tag.N

クラスターに割り当てられるタグ。

タイプ: Tag オブジェクトの配列

必須: いいえ

VpcSecurityGroupIds.VpcSecurityGroupId.N

このクラスターに関連付ける EC2 VPC セキュリティグループのリスト。

型: 文字列の配列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBCluster

クラスターに関する詳細情報。

型: DBCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterAlreadyExistsFault

指定された識別子を持つクラスターをすでに持っています。

HTTP ステータスコード: 400

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

DBClusterParameterGroupNotFound

`DBClusterParameterGroupName` は既存のパラメータグループを参照していません。

HTTP ステータスコード: 404

DBClusterQuotaExceededFault

クラスターの最大許容クォータに達したため、クラスターを作成できません。

HTTP ステータスコード: 403

DBInstanceNotFound

`DBInstanceIdentifier` は既存のインスタンスを参照していません。

HTTP ステータスコード: 404

DBSubnetGroupDoesNotCoverEnoughAZs

アベイラビリティーゾーンが 1 つしかない場合を除き、サブネットグループ内のサブネットは 2 つ以上のアベイラビリティゾーンを力バーする必要があります。

HTTP ステータスコード: 400

DBSubnetGroupNotFoundFault

DBSubnetGroupName は既存のサブネットグループを参照していません。

HTTP ステータスコード: 404

GlobalClusterNotFoundFault

GlobalClusterIdentifier は既存のグローバルクラスターを参照していません。

HTTP ステータスコード: 404

InsufficientStorageClusterCapacity

現在のアクションに使用可能なストレージが不十分です。使用できるストレージがより多い別のアベイラビリティーゾーンを使用するようにサブネットグループを更新することで、このエラーを解決できる場合があります。

HTTP ステータスコード: 400

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

InvalidDBInstanceState

指定されたインスタンスは 利用可能な 状態ではありません。

HTTP ステータスコード: 400

InvalidDBSubnetGroupStateFault

サブネットグループは使用中のため削除できません。

HTTP ステータスコード: 400

InvalidGlobalClusterStateFault

要求された操作は、クラスターがこの状態にある間は実行できません。

HTTP ステータスコード: 400

InvalidSubnet

要求されたサブネットが無効であるか、共通の仮想プライベートクラウド（VPC）内に存在しない複数のサブネットが要求されました。

HTTP ステータスコード: 400

InvalidVPCNetworkStateFault

サブネットグループは、作成後に変更されたため、すべてのアベイラビリティーゾーンがカバーされません。

HTTP ステータスコード: 400

KMSKeyNotAccessibleFault

AWS KMS キーへのアクセス中にエラーが発生しました。

HTTP ステータスコード: 400

StorageQuotaExceeded

リクエストにより、すべてのインスタンスで使用可能な許容ストレージ容量を超過します。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateDBClusterParameterGroup

サービス : Amazon DocumentDB (with MongoDB compatibility)

新しいクラスターのパラメータグループを作成します。

クラスターパラメータグループのパラメータは、クラスター内のすべてのインスタンスに適用されます。

クラスターパラメータグループは、最初はクラスターのインスタンスで使用されるデータベースエンジンのデフォルトパラメータで作成されます。Amazon DocumentDBでは、`default.docdb3.6` クラスターパラメータグループに直接変更を加えることはできません。Amazon DocumentDB クラスターがデフォルトのクラスタパラメータグループを使用していて、その中の値を変更する場合は、まず 新しいパラメータグループを作成 するか、既存パラメータグループをコピー して変更した後に、変更したパラメータグループをクラスターに適用します。新しいクラスターパラメータグループと関連する設定を有効にするには、クラスターのインスタンスをフェイルオーバーなしで再起動する必要があります。詳細については、「Amazon DocumentDB クラスターパラメータグループの変更」を参照してください。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterParameterGroupName

クラスターパラメータグループの名前。

制約:

- 既存の `DBClusterParameterGroup` の名前と一致してはなりません。

> **Note**
>
> この値は小文字で保存されます。

タイプ: 文字列

必須: はい

DBParameterGroupFamily

クラスターパラメータグループファミリーの名前。

タイプ: 文字列

必須: はい

Description

クラスターパラメータグループの説明。

タイプ: 文字列

必須: はい

Tags.Tag.N

クラスターパラメータグループに割り当てられるタグ。

タイプ: Tag オブジェクトの配列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBClusterParameterGroup

クラスターパラメータグループに関する詳細情報。

型: DBClusterParameterGroup オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBParameterGroupAlreadyExists

同じ名前のパラメータグループがすでに存在します。

HTTP ステータスコード: 400

DBParameterGroupQuotaExceeded

このリクエストにより、パラメータグループの許容数を超過します。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateDBClusterSnapshot

サービス : Amazon DocumentDB (with MongoDB compatibility)

クラスターのスナップショットを作成します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterIdentifier

スナップショットを作成するクラスターの識別子。このパラメータは大文字と小文字が区別されません。

制約:

- 既存の DBCluster の識別子と一致する必要があります。

例: `my-cluster`

タイプ: 文字列

必須: はい

DBClusterSnapshotIdentifier

クラススナップショットの識別子。このパラメータは小文字で保存されます。

制約:

- 1 ~ 63 の文字、数字またはハイフンを使用する必要があります。
- 1 字目は文字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

例: `my-cluster-snapshot1`

タイプ: 文字列

必須: はい

Tags.Tag.N

クラススナップショットに割り当てられるタグ。

タイプ: Tag オブジェクトの配列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBClusterSnapshot

クラススナップショットに関する詳細情報。

型: DBClusterSnapshot オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

DBClusterSnapshotAlreadyExistsFault

指定された識別子を持つクラススナップショットをすでに持っています。

HTTP ステータスコード: 400

InvalidDBClusterSnapshotStateFault

指定された値は有効なクラススナップショット状態ではありません。

HTTP ステータスコード: 400

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

SnapshotQuotaExceeded

リクエストにより、スナップショットの許容数を超過します。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateDBInstance

サービス : Amazon DocumentDB (with MongoDB compatibility)

新しいインスタンスを作成します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterIdentifier

インスタンスが属するクラスターの識別子。

タイプ: 文字列

必須: はい

DBInstanceClass

インスタンスのコンピューティング能力とメモリ容量 (例: `db.r5.large`)。

タイプ: 文字列

必須: はい

DBInstanceIdentifier

インスタンス識別子。このパラメータは小文字で保存されます。

制約:

- 1 ~ 63 の文字、数字またはハイフンを使用する必要があります。
- 1 字目は文字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

例: `mydbinstance`

タイプ: 文字列

必須: はい

Engine

このインスタンスに使用するデータベースエンジンの名前。

有効な値: docdb

タイプ: 文字列

必須: はい

AutoMinorVersionUpgrade

このパラメータは Amazon DocumentDB には適用されません。Amazon DocumentDB は、値セットに関係なく、マイナーバージョンのアップグレードを実行しません。

デフォルト: false

タイプ: ブール値

必須: いいえ

AvailabilityZone

インスタンスが作成される Amazon EC2 アベイラビリティーゾーン。

デフォルト: エンドポイントのランダムなシステム選択のアベイラビリティーゾーン AWS リージョン。

例: us-east-1d

タイプ: 文字列

必須: いいえ

CACertificateIdentifier

DB インスタンスのサーバー証明書に使用する CA 証明書識別子。

詳細については、「Amazon DocumentDB デベロッパーガイド」の「Amazon DocumentDB TLS 証明書の更新」と「転送中のデータの暗号化」を参照してください。

タイプ: 文字列

必須: いいえ

CopyTagsToSnapshot

タグを DB インスタンスから DB インスタンスのスナップショットにコピーするかどうかを示す値。デフォルトでは、タグはコピーされません。

型: ブール値

必須: いいえ

EnablePerformanceInsights

DB インスタンスで Performance Insights を有効にするかどうかを示す値。詳細については、「Amazon Performance Insights の使用」を参照してください。

型: ブール値

必須: いいえ

PerformanceInsightsKMSKeyId

Performance Insights データを暗号化するための AWS KMS キー識別子。

AWS KMS キー識別子は、KMS キーのキー ARN、キー ID、エイリアス ARN、またはエイリアス名です。

PerformanceInsightsKMSKeyId に値を指定しない場合、Amazon DocumentDB はデフォルトの KMS キーを使用します。Amazon Web Services アカウントにはデフォルトの KMS キーがあります。Amazon Web Services アカウントには Amazon Web Services リージョンごとに異なるデフォルトの KMS キーがあります。

タイプ: 文字列

必須: いいえ

PreferredMaintenanceWindow

毎週 1 回のシステムメンテナンスを実行できる時間帯、世界標準時 (UTC)。

形式: `ddd:hh24:mi-ddd:hh24:mi`

デフォルトは、それぞれ 8 時間の時間ブロックからランダムに選択された 30 分のウィンドウであり AWS リージョン、ランダムな曜日に発生します。

有効な曜日: 月、火、水、木、金、土、日

制約: 最小限の 30 分単位のウィンドウ。

タイプ: 文字列

必須: いいえ

PromotionTier

既存のプライマリインスタンスの障害後に、Amazon DocumentDB レプリカがプライマリインスタンスに昇格する順序を指定する値。

デフォルト: 1

有効な値: 0 ~ 15

タイプ: 整数

必須: いいえ

Tags.Tag.N

インスタンスに割り当てられるタグ。インスタンスには最大 10 個のタグを割り当てることができます。

タイプ: Tag オブジェクトの配列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBInstance

インスタンスに関する詳細情報。

型: DBInstance オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AuthorizationNotFound

指定された CIDRIP または Amazon EC2 セキュリティグループは、指定されたセキュリティグループに対して権限がありません。

Amazon DocumentDB には、IAM を使用してお客様に代わって必要なアクションを実行する権限がない場合もあります。

HTTP ステータスコード: 404

DBClusterNotFoundFault

DBClusterIdentifier は既存のクラスターを参照していません。

HTTP ステータスコード: 404

DBInstanceAlreadyExists

指定された識別子を持つインスタンスをすでに持っています。

HTTP ステータスコード: 400

DBParameterGroupNotFound

DBParameterGroupName は既存のパラメータグループを参照していません。

HTTP ステータスコード: 404

DBSecurityGroupNotFound

DBSecurityGroupName は既存のセキュリティグループを参照していません。

HTTP ステータスコード: 404

DBSubnetGroupDoesNotCoverEnoughAZs

アベイラビリティーゾーンが 1 つしかない場合を除き、サブネットグループ内のサブネットは 2 つ以上のアベイラビリティゾーンをカバーする必要があります。

HTTP ステータスコード: 400

DBSubnetGroupNotFoundFault

DBSubnetGroupName は既存のサブネットグループを参照していません。

HTTP ステータスコード: 404

InstanceQuotaExceeded

リクエストにより、インスタンスの許容数を超過します。

HTTP ステータスコード: 400

InsufficientDBInstanceCapacity

指定されたインスタンスクラスは、指定されたアベイラビリティーゾーンで利用できません。

HTTP ステータスコード: 400

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

InvalidSubnet

要求されたサブネットが無効であるか、共通の仮想プライベートクラウド（VPC）内に存在しない複数のサブネットが要求されました。

HTTP ステータスコード: 400

InvalidVPCNetworkStateFault

サブネットグループは、作成後に変更されたため、すべてのアベイラビリティーゾーンがカバーされません。

HTTP ステータスコード: 400

KMSKeyNotAccessibleFault

AWS KMS キーへのアクセス中にエラーが発生しました。

HTTP ステータスコード: 400

StorageQuotaExceeded

リクエストにより、すべてのインスタンスで使用可能な許容ストレージ容量を超過します。

HTTP ステータスコード: 400

StorageTypeNotSupported

指定された `StorageType` のストレージはDBインスタンスに関連付けられません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateDBSubnetGroup

サービス : Amazon DocumentDB (with MongoDB compatibility)

新しいサブネットグループを作成します。サブネットグループには、 AWS リージョン内の 2 つ以上のアベイラビリティゾーンに 1 つ以上のサブネットが含まれる必要があります。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBSubnetGroupDescription

サブネットグループの説明。

タイプ: 文字列

必須: はい

DBSubnetGroupName

サブネットグループの名前。この値は小文字で保存されます。

制約: 最大 255 文字の英字、数字、ピリオド、アンダースコア、スペース、またはハイフンのみを使用できます。デフォルト値を使用することはできません。

例: `mySubnetgroup`

タイプ: 文字列

必須: はい

SubnetIds.SubnetIdentifier.N

サブネットグループの Amazon EC2 サブネット ID。

型: 文字列の配列

必須: はい

Tags.Tag.N

サブネットグループに割り当てられるタグ。

タイプ: Tag オブジェクトの配列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBSubnetGroup

サブネットグループに関する詳細情報。

型: DBSubnetGroup オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBSubnetGroupAlreadyExists

`DBSubnetGroupName` は既存のサブネットグループですでに使用されています。

HTTP ステータスコード: 400

DBSubnetGroupDoesNotCoverEnoughAZs

アベイラビリティーゾーンが 1 つしかない場合を除き、サブネットグループ内のサブネットは 2 つ以上のアベイラビリティゾーンをカバーする必要があります。

HTTP ステータスコード: 400

DBSubnetGroupQuotaExceeded

リクエストにより、サブネットグループ許容数を超過します。

HTTP ステータスコード: 400

DBSubnetQuotaExceededFault

リクエストにより、サブネットグループのサブネットの許容数を超過します。

HTTP ステータスコード: 400

InvalidSubnet

要求されたサブネットが無効であるか、共通の仮想プライベートクラウド（VPC）内に存在しない複数のサブネットが要求されました。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateEventSubscription

サービス : Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB のイベント通知サブスクリプションを作成します。このアクションには、Amazon DocumentDB コンソール、Amazon SNS コンソール、または Amazon SNS API を使用して作成したトピック ARN (Amazon リソースネーム) が必要です。SNS で ARN を取得するには、Amazon SNS でトピックを作成し、そのトピックをサブスクライブする必要があります。ARN は Amazon SNS コンソールに表示されます。

通知を受け取る対象となるソースの種類 (SourceType) を指定ができます。また、イベントをトリガーする Amazon DocumentDB ソース (SourceIds) のリストを提供して、通知を受けるイベントのカテゴリ (EventCategories) のリストを提供できます。例えば、SourceType = db-instance、SourceIds = mydbinstance1, mydbinstance2、EventCategories = Availability, Backup と指定できます。

SourceType と SourceIds を、[SourceType = db-instance] と [SourceIdentifier = myDBInstance1] などに指定すると、指定したソースのすべての db-instance イベントが通知されます。[SourceType SourceType] を指定し、[SourceIdentifier SourceIdentifier] を指定しない場合、指定したAmazon DocumentDB ソースのすべてのイベント通知を受け取ります。SourceType も SourceIdentifier も指定しない場合、顧客アカウントに属する Amazon DocumentDB ソースから生成されたすべてのイベントの通知を受け取ります。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

SnsTopicArn

イベント通知用に作成された SNS トピックの Amazon リソースネーム (ARN)。ARN は、トピックを作成してそれをサブスクライブするときに Amazon SNS によって作成されます。

タイプ: 文字列

必須: はい

SubscriptionName

サブスクリプションの名前。

制約: この名前は 255 文字未満である必要があります。

タイプ: 文字列

必須: はい

Enabled

ブール値。サブスクリプションを有効にする場合は `true` に設定し、サブスクリプションを作成するが有効にしない場合は `false` に設定します。

型: ブール値

必須: いいえ

EventCategories.EventCategory.N

サブスクライブする `SourceType` のイベントカテゴリのリスト。

型: 文字列の配列

必須: いいえ

SourceIds.SourceId.N

イベントが返されるイベントソースの識別子のリスト。指定しない場合は、すべてのソースはレスポンスに含まれます。識別子は文字で開始し、ASCII 文字、数字、ハイフンのみを使用できます。最後の文字をハイフンにすることはできず、ハイフンを 2 つ続けて使用することもできません。

制約:

- `SourceIds` が指定される場合、`SourceType` も指定する必要があります。
- ソースタイプがインスタンスである場合は、`DBInstanceIdentifier` を指定する必要があります。
- ソースタイプがセキュリティグループである場合は、`DBSecurityGroupName` を指定する必要があります。
- ソースタイプがパラメータグループである場合は、`DBParameterGroupName` を指定する必要があります。
- ソースタイプがスナップショットである場合は、`DBSnapshotIdentifier` を指定する必要があります。

型: 文字列の配列

必須: いいえ

SourceType

イベントを生成しているソースの種類。例えば、インスタンスが生成したイベントの通知を受ける場合は、このパラメータを db-instance に設定します。この値を指定しない場合、すべてのイベントが返されます。

有効な値:db-instance、db-cluster、db-parameter-group、db-security-group、db-cluster-snapshot

タイプ: 文字列

必須: いいえ

Tags.Tag.N

イベントサブスクリプションに割り当てられるタグ。

タイプ: Tag オブジェクトの配列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

EventSubscription

サブスクライブしたイベントの詳細情報。

型: EventSubscription オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

EventSubscriptionQuotaExceeded

イベントサブスクリプション数が最大に達しました。

HTTP ステータスコード: 400

SNSInvalidTopic

Amazon SNS は、指定されたトピックに問題があると回答しました。

HTTP ステータスコード: 400

SNSNoAuthorization

SNS トピックの Amazon リソースネーム (ARN) に発行する権限がありません。

HTTP ステータスコード: 400

SNSTopicArnNotFound

SNS トピックの Amazon リソースネーム (ARN) が存在しません。

HTTP ステータスコード: 404

SourceNotFound

リクエストされたリソースが見つかりませんでした。

HTTP ステータスコード: 404

SubscriptionAlreadyExist

指定されたサブスクリプション名はすでに存在します。

HTTP ステータスコード: 400

SubscriptionCategoryNotFound

指定されたカテゴリは存在しません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3

- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateGlobalCluster

サービス : Amazon DocumentDB (with MongoDB compatibility)

複数の AWS リージョンにまたがることができる Amazon DocumentDB グローバルクラスターを作成します。グローバルクラスターには、読み取り/書き込み機能を備えた 1 つのプライマリクラスターと、最大 5 つの読み取り専用セカンダリクラスターが含まれます。グローバルクラスターは、ワークロードのパフォーマンスに影響を与えることなく、専用のインフラストラクチャを使用して、1 秒未満のレイテンシーでリージョンにわたってストレージベースの高速レプリケーションを使用します。

最初は空のグローバルクラスターを作成し、そこにプライマリとセカンダリを追加できます。または、作成操作中に既存のクラスターを指定し、そのクラスターをグローバルクラスターのプライマリにすることもできます。

> **Note**
>
> このアクションは Amazon DocumentDB クラスターにのみ適用されます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

GlobalClusterIdentifier

新しいグローバルクラスターのクラスター識別子。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

必須: はい

DatabaseName

データベースの名前を、英数字 64 文字以内で入力します。名前を入力しない場合は、Amazon DocumentDB は作成するグローバルクラスター上にデータベースを作成しません。

タイプ: 文字列

必須: いいえ

DeletionProtection

新しいグローバルクラスターの削除保護設定。削除保護が有効な場合、グローバルクラスターは削除できません。

型: ブール値

必須: いいえ

Engine

このクラスターに使用されるデータベースエンジン名。

タイプ: 文字列

必須: いいえ

EngineVersion

グローバルクラスターのエンジンバージョン。

タイプ: 文字列

必須: いいえ

SourceDBClusterIdentifier

グローバルクラスターのプライマリクラスターとして使用する Amazon リソースネーム (ARN)。このパラメータはオプションです。

タイプ: 文字列

必須: いいえ

StorageEncrypted

新しいグローバルクラスターのストレージ暗号化設定。

型: ブール値

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

GlobalCluster

Amazon DocumentDB グローバルクラスターを表すデータ型。

型: GlobalCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

GlobalClusterAlreadyExistsFault

`GlobalClusterIdentifier` は既に存在します。新しいグローバルクラスター識別子 (一意の名前) を選択して、新しいグローバルクラスターを作成します。

HTTP ステータスコード: 400

GlobalClusterQuotaExceededFault

このアカウントのグローバルクラスター数は、すでに許容最大数に達しています。

HTTP ステータスコード: 400

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteDBCluster

サービス : Amazon DocumentDB (with MongoDB compatibility)

以前にプロビジョニングされたクラスターを削除します。クラスターを削除すると、そのクラスターの自動バックアップはすべて削除され、復旧できません。指定したクラスターの手動 DB クラスタースナップショットは削除されません。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterIdentifier

削除するクラスターのクラスター識別子。このパラメータでは大文字と小文字は区別されません。

制約:

- 既存の DBClusterIdentifier と一致する必要があります。

タイプ: 文字列

必須: はい

FinalDBSnapshotIdentifier

SkipFinalSnapshot が false に設定された場合の、作成された新しいクラスタースナップショットのクラスタースナップショット識別子。

> **Note**
>
> このパラメータを指定して SkipFinalShapshot パラメータを true に設定すると、エラーが発生します。

制約:

- 1 ~ 255 の英字、数字、ハイフンである必要があります。
- 1 字目は文字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

タイプ: 文字列

必須: いいえ

SkipFinalSnapshot

クラスターの削除前に最終のクラススナップショットを作成するかどうかを指定します。`true` を指定した場合、クラススナップショットが作成されます。`false` を指定した場合、DB クラスターの削除前にクラススナップショットが作成されます。

> **Note**
>
> `SkipFinalSnapshot` が `false` の場合は、`FinalDBSnapshotIdentifier` パラメータを指定する必要があります。

デフォルト: `false`

タイプ: ブール値

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBCluster

クラスターに関する詳細情報。

型: DBCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

DBClusterSnapshotAlreadyExistsFault

指定された識別子を持つクラススナップショットをすでに持っています。

HTTP ステータスコード: 400

InvalidDBClusterSnapshotStateFault

指定された値は有効なクラススナップショット状態ではありません。

HTTP ステータスコード: 400

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

SnapshotQuotaExceeded

リクエストにより、スナップショットの許容数を超過します。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteDBClusterParameterGroup

サービス : Amazon DocumentDB (with MongoDB compatibility)

指定されたクラスターパラメータグループを削除します。削除するクラスターパラメータグループは、どのクラスターにも関連付けることができません。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterParameterGroupName

クラスターパラメータグループの名前。

制約:

- 既存のクラスターパラメータグループ名にする必要があります。
- デフォルトのクラスターパラメータグループは削除できません。
- どのクラスターにも関連付けることができません。

タイプ: 文字列

必須: はい

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBParameterGroupNotFound

`DBParameterGroupName` は既存のパラメータグループを参照していません。

HTTP ステータスコード: 404

InvalidDBParameterGroupState

パラメータグループが使用中であるか、無効な状態です。パラメータグループを削除しようとする場合、パラメータグループがこの状態のときは削除できません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteDBClusterSnapshot

サービス : Amazon DocumentDB (with MongoDB compatibility)

クラススナップショットを削除します。スナップショットがコピーされている場合、コピー操作は終了します。

> **Note**
>
> クラススナップショットを削除するには `available` の状態にする必要があります。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterSnapshotIdentifier

削除するクラススナップショットの識別子。

制約: `available` の状態にある既存のクラススナップショット名にする必要があります。

タイプ: 文字列

必須: はい

レスポンス要素

以下の要素がサービスによって返されます。

DBClusterSnapshot

クラススナップショットに関する詳細情報。

型: DBClusterSnapshot オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier` は既存のクラススナップショットを参照していません。

HTTP ステータスコード: 404

InvalidDBClusterSnapshotStateFault

指定された値は有効なクラススナップショット状態ではありません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteDBInstance

サービス : Amazon DocumentDB (with MongoDB compatibility)

以前にプロビジョニングされたインスタンスを削除します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBInstanceIdentifier

削除するインスタンスのインスタンス識別子。このパラメータでは大文字と小文字は区別されません。

制約:

- 既存のインスタンス名と一致する必要があります。

タイプ: 文字列

必須: はい

レスポンス要素

以下の要素がサービスによって返されます。

DBInstance

インスタンスに関する詳細情報。

型: DBInstance オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBInstanceNotFound

`DBInstanceIdentifier` が既存の DB インスタンスを参照していません。

HTTP ステータスコード: 404

DBSnapshotAlreadyExists

DBSnapshotIdentifier は既存のスナップショットですでに使用されています。

HTTP ステータスコード: 400

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

InvalidDBInstanceState

指定されたインスタンスは 利用可能な 状態ではありません。

HTTP ステータスコード: 400

SnapshotQuotaExceeded

リクエストにより、スナップショットの許容数を超過します。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteDBSubnetGroup

サービス : Amazon DocumentDB (with MongoDB compatibility)

サブネットグループを削除します。

> **Note**
>
> 指定されたデータベースサブネットグループは、どの DB インスタンスにも関連付けないでください。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBSubnetGroupName

削除するデータベースサブネットグループの名前。

> **Note**
>
> デフォルトのサブネットグループを削除することはできません。

制約:

既存の DBSubnetGroup の名前と一致する必要があります。デフォルト値を使用することはできません。

例: `mySubnetgroup`

タイプ: 文字列

必須: はい

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBSubnetGroupNotFoundFault

DBSubnetGroupName は既存のサブネットグループを参照していません。

HTTP ステータスコード: 404

InvalidDBSubnetGroupStateFault

サブネットグループは使用中のため削除できません。

HTTP ステータスコード: 400

InvalidDBSubnetStateFault

サブネットは 利用可能な 状態にありません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteEventSubscription

サービス : Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB イベント通知サブスクリプション削除します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

SubscriptionName

削除する Amazon DocumentDB イベント通知サブスクリプションの名前。

タイプ: 文字列

必須: はい

レスポンス要素

以下の要素がサービスによって返されます。

EventSubscription

サブスクライブしたイベントの詳細情報。

型: EventSubscription オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

InvalidEventSubscriptionState

他の誰かがサブスクリプションを変更している可能性があります。数秒後に再度お試しください。

HTTP ステータスコード: 400

SubscriptionNotFound

サブスクリプション名が存在しません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteGlobalCluster

サービス : Amazon DocumentDB (with MongoDB compatibility)

グローバルクラスターを削除します。グローバルクラスターを削除する前に、プライマリクラスタとセカンダリクラスタがすでにデタッチまたは削除されている必要があります。

> **Note**
>
> このアクションは Amazon DocumentDB クラスターにのみ適用されます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

GlobalClusterIdentifier

削除されるグローバルクラスターのクラスター識別子。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

必須: はい

レスポンス要素

以下の要素がサービスによって返されます。

GlobalCluster

Amazon DocumentDB グローバルクラスターを表すデータ型。

型: GlobalCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

GlobalClusterNotFoundFault

`GlobalClusterIdentifier` は既存のグローバルクラスターを参照していません。

HTTP ステータスコード: 404

InvalidGlobalClusterStateFault

要求された操作は、クラスターがこの状態にある間は実行できません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeCertificates

サービス : Amazon DocumentDB (with MongoDB compatibility)

この AWS アカウントに対して Amazon DocumentDB から提供された認証局 (CA) 証明書のリストを返します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

CertificateIdentifier

ユーザーが指定する証明書識別子。このパラメータを指定した場合、指定された証明書のみに関する情報を返します。このパラメータを省略すると、最大 `MaxRecords` の証明書のリストが返されます。このパラメータは大文字と小文字が区別されません。

制約

- 既存の `CertificateIdentifier` と一致する必要があります。

タイプ: 文字列

必須: いいえ

Filters.Filter.N

このパラメータは、現在サポートされていません。

タイプ: Filter オブジェクトの配列

必須: いいえ

Marker

以前の `DescribeCertificates` リクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには `MaxRecords` で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された MaxRecords の値よりも多くのレコードが存在する場合、マーカーと呼ばれるページ割りトークンがレスポンスに含まれるため、残りの結果を取得できます。

デフォルト: 100

制約:

- 最小: 20
- 最大: 100

タイプ: 整数

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

Certificates.Certificate.N

この AWS アカウントに関する証明書のリスト。

型: Certificate オブジェクトの配列

Marker

取得されたレコード数が MaxRecords より大きい場合に提供されるオプションのページ割りトークン。このパラメータを指定した場合、マーカーはリスト内の次のレコードを指定します。Marker の値を次の DescribeCertificates への呼び出しに含めると、証明書の次のページに表示されます。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

CertificateNotFound

CertificateIdentifier は既存の証明書を参照していません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeDBClusterParameterGroups

サービス : Amazon DocumentDB (with MongoDB compatibility)

DBClusterParameterGroup の説明のリストを返します。DBClusterParameterGroupName パラメータを指定した場合、リストには指定した DB クラスターパラメータグループの説明のみが含まれます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterParameterGroupName

詳細を返す特定のクラスターパラメータグループ名。

制約:

- 指定した場合、既存の DBClusterParameterGroup の名前と一致する必要があります。

タイプ: 文字列

必須: いいえ

Filters.Filter.N

このパラメータは、現在サポートされていません。

タイプ: Filter オブジェクトの配列

必須: いいえ

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された MaxRecords の値よりも多くのレコードが存在する場合、ページ割りトークン (マーカー) がレスポンスに含まれるため、残りの結果を取得できます。

デフォルト: 100

制約: 最小 20、最大 100。

タイプ: 整数

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBClusterParameterGroups.DBClusterParameterGroup.N

クラスターパラメータグループのリスト。

型: DBClusterParameterGroup オブジェクトの配列

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBParameterGroupNotFound

DBParameterGroupName は既存のパラメータグループを参照していません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeDBClusterParameters

サービス : Amazon DocumentDB (with MongoDB compatibility)

特定のクラスターパラメータグループの詳細なパラメータリストを返します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterParameterGroupName

パラメータの詳細を返す特定のクラスターパラメータグループ名。

制約:

- 指定した場合、既存の DBClusterParameterGroup の名前と一致する必要があります。

タイプ: 文字列

必須: はい

Filters.Filter.N

このパラメータは、現在サポートされていません。

タイプ: Filter オブジェクトの配列

必須: いいえ

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された MaxRecords の値よりも多くのレコードが存在する場合、ページ割りトークン (マーカー) がレスポンスに含まれるため、残りの結果を取得できます。

デフォルト: 100

制約: 最小 20、最大 100。

タイプ: 整数

必須: いいえ

Source

特定のソースのパラメータのみを返すことを示す値。パラメータのソースは engine、service、または customer のいずれかとすることができます。

タイプ: 文字列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

Parameters.Parameter.N

クラスターパラメータグループのパラメータのリストを提供します。

型: Parameter オブジェクトの配列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBParameterGroupNotFound

DBParameterGroupName は既存のパラメータグループを参照していません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeDBClusters

サービス : Amazon DocumentDB (with MongoDB compatibility)

プロビジョニングされた Amazon DocumentDB クラスターに関する情報を返します。この API 操作はページ分割に対応しています。クラスターやインスタンスのライフサイクル管理などの特定の管理機能において、Amazon DocumentDB は Amazon RDS や Amazon Neptune と共有される運用テクノロジーを使用します。Amazon DocumentDB クラスターのみを返すには、`filterName=engine,Values=docdb` フィルターパラメータを使用します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterIdentifier

ユーザーが指定したクラスター識別子。このパラメータを指定した場合、特定のクラスターからの情報のみを返します。このパラメータでは大文字と小文字は区別されません。

制約:

- 指定した場合、既存の `DBClusterIdentifier` と一致する必要があります。

タイプ: 文字列

必須: いいえ

Filters.Filter.N

記述する 1 つ以上のクラスターを指定するフィルター。

サポートされているフィルター:

- `db-cluster-id` - クラスター識別子とクラスターの Amazon リソースネーム (ARN) を受け入れます。結果リストには、これらの ARN で識別されるクラスターに関する情報のみが含まれます。

タイプ: Filter オブジェクトの配列

必須: いいえ

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された MaxRecords の値よりも多くのレコードが存在する場合、ページ割りトークン (マーカー) がレスポンスに含まれるため、残りの結果を取得できます。

デフォルト: 100

制約: 最小 20、最大 100。

タイプ: 整数

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBClusters.DBCluster.N

クラスターのリスト。

型: DBCluster オブジェクトの配列

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeDBClusterSnapshotAttributes

サービス : Amazon DocumentDB (with MongoDB compatibility)

手動の DB クラスタースナップショットのクラスタースナップショット属性名と値のリストを返します。

スナップショットを他の と共有する場合 AWS アカウント、 は、手動クラスタースナップショットのコピーまたは復元 AWS アカウント が許可されている の `restore` 属性と IDs のリストDescribeDBClusterSnapshotAttributesを返します。`all` が `restore` 属性の値のリストに含まれている場合、手動クラスタースナップショットは公開されており、すべての AWS アカウントによってコピーまたは復元できます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterSnapshotIdentifier

属性を記述するクラスタースナップショット識別子。

タイプ: 文字列

必須: はい

レスポンス要素

以下の要素がサービスによって返されます。

DBClusterSnapshotAttributesResult

クラスタースナップショットに関連付けられている属性の詳細情報。

型: DBClusterSnapshotAttributesResult オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier` は既存のクラスタースナップショットを参照していません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeDBClusterSnapshots

サービス : Amazon DocumentDB (with MongoDB compatibility)

クラススナップショットに関する情報を返します。この API 操作はページ分割に対応しています。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterIdentifier

クラススナップショットのリストを取得する DB クラスターの ID。このパラメータは、DBClusterSnapshotIdentifier パラメータと併用できません。このパラメータは大文字と小文字が区別されません。

制約:

- 指定した場合、既存の DBCluster の識別子と一致する必要があります。

タイプ: 文字列

必須: いいえ

DBClusterSnapshotIdentifier

記述する特定のクラススナップショット識別子。このパラメータは、DBClusterIdentifier パラメータと併用できません。この値は小文字で保存されます。

制約:

- 指定した場合、既存の DBClusterSnapshot の識別子と一致する必要があります。
- この識別子が自動スナップショット用の場合、SnapshotType パラメータも指定する必要があります。

タイプ: 文字列

必須: いいえ

Filters.Filter.N

このパラメータは、現在サポートされていません。

タイプ: Filter オブジェクトの配列

必須: いいえ

IncludePublic

を に設定trueして、パブリックで、任意の によってコピーまたは復元できる手動クラスタースナップショットを含め AWS アカウント、それ以外の場合は を含めますfalse。デフォルトは false です。

型: ブール値

必須: いいえ

IncludeShared

に設定trueして、コピーまたは復元するアクセス許可 AWS アカウント が付与 AWS アカウントされている他の からの共有手動クラスタースナップショットを含め、それ以外の場合は を含めますfalse。デフォルトは false です。

型: ブール値

必須: いいえ

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された MaxRecords の値よりも多くのレコードが存在する場合、ページ割りトークン (マーカー) がレスポンスに含まれるため、残りの結果を取得できます。

デフォルト: 100

制約: 最小 20、最大 100。

タイプ: 整数

必須: いいえ

SnapshotType

返されるクラスタースナップショットのタイプ。次のいずれかの値を指定できます。

- automated - Amazon DocumentDB が AWS アカウントに対して自動作成したクラスタースナップショットをすべて返します。
- manual - AWS アカウントのために手動作成したクラスタースナップショットをすべて返します。
- shared - AWS アカウントに共有される手動クラスタースナップショットをすべて返します。
- public - 公開とマークされたクラスタースナップショットをすべて返します。

SnapshotType の値を指定しない場合、自動と手動の両方のクラスタースナップショットを返します。IncludeShared パラメータを true に設定すると、これらの結果に公開クラスタースナップショットを含めることができます。IncludePublic パラメータを true に設定すると、これらの結果に公開クラスタースナップショットを含めることができます。

IncludeShared および IncludePublic パラメータは、manual または automated の SnapshotType 値には適用されません。SnapshotType が shared に設定されている場合、IncludePublic パラメータは適用されません。SnapshotType が public に設定されている場合、IncludeShared パラメータは適用されません。

タイプ: 文字列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBClusterSnapshots.DBClusterSnapshot.N

ユーザーのクラスタースナップショットのリストを提供します。

型: DBClusterSnapshot オブジェクトの配列

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier` は既存のクラススナップショットを参照していません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeDBEngineVersions

サービス : Amazon DocumentDB (with MongoDB compatibility)

利用可能なエンジンのリストを返します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBParameterGroupFamily

詳細を返す特定のパラメータグループファミリー名。

制約:

- 指定した場合、既存の `DBParameterGroupFamily` と一致する必要があります。

タイプ: 文字列

必須: いいえ

DefaultOnly

指定されたエンジンまたはエンジンとメジャーバージョンの組み合わせのデフォルトバージョンのみが返されることを示します。

型: ブール値

必須: いいえ

Engine

返されるデータベースエンジン。

タイプ: 文字列

必須: いいえ

EngineVersion

返されるデータベースエンジンのバージョン。

例: `3.6.0`

タイプ: 文字列

必須: いいえ

Filters.Filter.N

このパラメータは、現在サポートされていません。

タイプ: Filter オブジェクトの配列

必須: いいえ

ListSupportedCharacterSets

このパラメータが指定され、要求されたエンジンが CreateDBInstance に対して CharacterSetName パラメータをサポートしている場合、レスポンスには各エンジンバージョンでサポートされている文字セットのリストが含まれます。

型: ブール値

必須: いいえ

ListSupportedTimezones

このパラメータが指定され、要求されたエンジンが CreateDBInstance に対して TimeZone パラメータをサポートしている場合、レスポンスには各エンジンバージョンでサポートされているタイムゾーンのリストが含まれます。

型: ブール値

必須: いいえ

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された MaxRecords の値よりも多くのレコードが存在する場合、ページ割りトークン (マーカー) がレスポンスに含まれるため、残りの結果を取得できます。

デフォルト: 100

制約: 最小 20、最大 100。

タイプ: 整数

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBEngineVersions.DBEngineVersion.N

1 つ以上のエンジンのバージョンに関する詳細情報。

型: DBEngineVersion オブジェクトの配列

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには `MaxRecords` で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeDBInstances

サービス : Amazon DocumentDB (with MongoDB compatibility)

プロビジョニングされた Amazon DocumentDB インスタンスに関する情報を返します。この API はページ分割をサポートします。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBInstanceIdentifier

ユーザーが指定したインスタンス識別子。このパラメータを指定した場合、特定のクラスターからの情報のみを返します。このパラメータでは大文字と小文字は区別されません。

制約:

- 指定した場合、既存の `DBInstance` の識別子と一致する必要があります。

タイプ: 文字列

必須: いいえ

Filters.Filter.N

記述する 1 つ以上のインスタンスを指定するフィルター。

サポートされているフィルター:

- `db-cluster-id` - クラスター識別子とクラスターの Amazon リソースネーム (ARN) を受け入れます。結果リストには、これらの ARN で識別されるクラスターに関連付けられたインスタンスに関する情報のみが含まれます。
- `db-instance-id` - インスタンス識別子とインスタンス ARN を受け入れます。結果リストには、これらの ARN で識別されるインスタンスに関する情報のみが含まれます。

タイプ: Filter オブジェクトの配列

必須: いいえ

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには `MaxRecords` で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された MaxRecords の値よりも多くのレコードが存在する場合、ページ割りトークン (マーカー) がレスポンスに含まれるため、残りの結果を取得できます。

デフォルト: 100

制約: 最小 20、最大 100。

タイプ: 整数

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBInstances.DBInstance.N

1 つ以上のインスタンスに関する詳細情報。

型: DBInstance オブジェクトの配列

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBInstanceNotFound

DBInstanceIdentifier が既存の DB インスタンスを参照していません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeDBSubnetGroups

サービス : Amazon DocumentDB (with MongoDB compatibility)

DBSubnetGroup の説明のリストを返します。DBSubnetGroupName を指定した場合、リストには指定した DBSubnetGroup 説明のみが含まれます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBSubnetGroupName

詳細を返すサブネットグループ名。

タイプ: 文字列

必須: いいえ

Filters.Filter.N

このパラメータは、現在サポートされていません。

タイプ: Filter オブジェクトの配列

必須: いいえ

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された MaxRecords の値よりも多くのレコードが存在する場合、ページ割りトークン (マーカー) がレスポンスに含まれるため、残りの結果を取得できます。

デフォルト: 100

制約: 最小 20、最大 100。

タイプ: 整数

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBSubnetGroups.DBSubnetGroup.N

1 つ以上のサブネットグループに関する詳細情報。

型: DBSubnetGroup オブジェクトの配列

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには `MaxRecords` で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBSubnetGroupNotFoundFault

`DBSubnetGroupName` は既存のサブネットグループを参照していません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeEngineDefaultClusterParameters

サービス : Amazon DocumentDB (with MongoDB compatibility)

クラスターのデータベースエンジンのデフォルトのエンジンおよびシステムパラメータ情報を返します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBParameterGroupFamily

エンジンパラメータ情報を返すクラスターパラメータグループファミリー名。

タイプ: 文字列

必須: はい

Filters.Filter.N

このパラメータは、現在サポートされていません。

タイプ: Filter オブジェクトの配列

必須: いいえ

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには `MaxRecords` で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された `MaxRecords` の値よりも多くのレコードが存在する場合、ページ割りトークン (マーカー) がレスポンスに含まれるため、残りの結果を取得できます。

デフォルト: 100

制約: 最小 20、最大 100。

タイプ: 整数

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

EngineDefaults

`DescribeEngineDefaultClusterParameters` アクションの呼び出しが成功した結果が含まれています。

型: EngineDefaults オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeEventCategories

サービス : Amazon DocumentDB (with MongoDB compatibility)

すべてのイベントソースタイプか、指定されている場合は、指定されたソースタイプのイベントカテゴリのリストを表示します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

Filters.Filter.N

このパラメータは、現在サポートされていません。

タイプ: Filter オブジェクトの配列

必須: いいえ

SourceType

イベントを生成しているソースの種類。

有効な値: `db-instance`、`db-parameter-group`、`db-security-group`

タイプ: 文字列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

EventCategoriesMapList.EventCategoriesMap.N

イベントカテゴリマップのリスト。

型: EventCategoriesMap オブジェクトの配列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeEvents

サービス : Amazon DocumentDB (with MongoDB compatibility)

インスタンス、セキュリティグループ、スナップショット、DB パラメータグループに関連する過去 14 日間のイベントを返します。特定の DB インスタンス、セキュリティグループ、データベーススナップショット、またはパラメータグループに固有のイベントは、名前をパラメータとして指定して取得します。デフォルトでは、過去 1 時間のイベントを返します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

Duration

イベントを取得するための分数。

デフォルト: 60

タイプ: 整数

必須: いいえ

EndTime

ISO 8601 形式で指定された、イベントを取得する時間間隔の終了時刻。

例: 2009-07-08T18:00Z

型: タイムスタンプ

必須: いいえ

EventCategories.EventCategory.N

イベント通知サブスクリプションの通知をトリガーするイベントカテゴリのリスト。

型: 文字列の配列

必須: いいえ

Filters.Filter.N

このパラメータは、現在サポートされていません。

タイプ: Filter オブジェクトの配列

必須: いいえ

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された MaxRecords の値よりも多くのレコードが存在する場合、ページ割りトークン (マーカー) がレスポンスに含まれるため、残りの結果を取得できます。

デフォルト: 100

制約: 最小 20、最大 100。

タイプ: 整数

必須: いいえ

SourceIdentifier

イベントが返されるイベントソースの識別子。指定しない場合は、すべてのソースはレスポンスに含まれます。

制約:

- SourceIdentifier を指定した場合、SourceType も指定する必要があります。
- ソースタイプが DBInstance である場合は、DBInstanceIdentifier を指定する必要があります。
- ソースタイプが DBSecurityGroup である場合は、DBSecurityGroupName を指定する必要があります。
- ソースタイプが DBParameterGroup である場合は、DBParameterGroupName を指定する必要があります。
- ソースタイプが DBSnapshot である場合は、DBSnapshotIdentifier を指定する必要があります。

- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

タイプ: 文字列

必須: いいえ

SourceType

イベントソースのイベントを取得します。値を指定しない場合、すべてのイベントが返されます。

型: 文字列

有効な値 : `db-instance | db-parameter-group | db-security-group | db-snapshot | db-cluster | db-cluster-snapshot`

必須 : いいえ

StartTime

ISO 8601 形式で指定された、イベントを取得する時間間隔の開始時刻。

例: 2009-07-08T18:00Z

型: タイムスタンプ

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

Events.Event.N

1 つ以上のイベントに関する詳細情報。

型: Event オブジェクトの配列

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには `MaxRecords` で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeEventSubscriptions

サービス : Amazon DocumentDB (with MongoDB compatibility)

顧客アカウントのサブスクリプションの説明をすべて表示します。サブスクリプションの説明には、SubscriptionName、SNSTopicARN、CustomerID、SourceType、SourceID、CreationTime、が含まれます。

SubscriptionName を指定すると、そのサブスクリプションの説明をリストアップします。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

Filters.Filter.N

このパラメータは、現在サポートされていません。

タイプ: Filter オブジェクトの配列

必須: いいえ

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された MaxRecords の値よりも多くのレコードが存在する場合、ページ割りトークン (マーカー) がレスポンスに含まれるため、残りの結果を取得できます。

デフォルト: 100

制約: 最小 20、最大 100。

タイプ: 整数

必須: いいえ

SubscriptionName

記述する Amazon DocumentDB イベント通知サブスクリプション名。

タイプ: 文字列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

EventSubscriptionsList.EventSubscription.N

イベントサブスクリプションのリスト。

型: EventSubscription オブジェクトの配列

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには `MaxRecords` で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

SubscriptionNotFound

サブスクリプション名が存在しません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeGlobalClusters

サービス : Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB グローバルクラスターに関する情報を返します。この API はページ分割をサポートします。

> **Note**
>
> このアクションは Amazon DocumentDB クラスターにのみ適用されます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

Filters.Filter.N

記述する 1 つ以上の DB クラスターを指定するフィルター。

対応するフィルター: `db-cluster-id` は、クラスター識別子とクラスター Amazon リソースネーム (ARN) を受け入れます。結果リストには、これらの ARN で識別されるクラスターに関連付けられたインスタンスに関する情報のみが含まれます。

タイプ: Filter オブジェクトの配列

必須: いいえ

GlobalClusterIdentifier

ユーザーが指定した DB クラスター識別子。このパラメータを指定した場合、特定のクラスターからの情報のみを返します。このパラメータでは大文字と小文字は区別されません。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

必須: いいえ

Marker

以前の DescribeGlobalClusters リクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された MaxRecords の値よりも多くのレコードが存在する場合、残りの結果を取得できるように、マーカーと呼ばれるページ割りトークンがレスポンスに含まれます。

タイプ: 整数

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

GlobalClusters.GlobalClusterMember.N

型: GlobalCluster オブジェクトの配列

Marker

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

GlobalClusterNotFoundFault

GlobalClusterIdentifier は既存のグローバルクラスターを参照していません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeOrderableDBInstanceOptions

サービス : Amazon DocumentDB (with MongoDB compatibility)

指定されたエンジンの指示可能なインスタンスオプションのリストを返します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

Engine

インスタンスオプションを取得するエンジン名。

タイプ: 文字列

必須: はい

DBInstanceClass

The instance class filter value。このパラメータを指定すると、指定したインスタンスクラスに一致する利用可能なオファリングみが表示されます。

タイプ: 文字列

必須: いいえ

EngineVersion

エンジンバージョンフィルターの値。このパラメータを指定すると、指定したエンジンのバージョンに一致する利用可能なオファリングみが表示されます。

タイプ: 文字列

必須: いいえ

Filters.Filter.N

このパラメータは、現在サポートされていません。

タイプ: Filter オブジェクトの配列

必須: いいえ

LicenseModel

ライセンスモデルフィルター値。このパラメータを指定すると、指定したエンジンのバージョンに一致する利用可能なライセンスモデルのみが表示されます。

タイプ: 文字列

必須: いいえ

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された MaxRecords の値よりも多くのレコードが存在する場合、ページ割りトークン (マーカー) がレスポンスに含まれるため、残りの結果を取得できます。

デフォルト: 100

制約: 最小 20、最大 100。

タイプ: 整数

必須: いいえ

Vpc

仮想プライベートクラウド (VPC) フィルター値。このパラメータを指定すると、 VPC 環境または非 VPC 環境のみが表示されます。

型: ブール値

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには MaxRecords で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

OrderableDBInstanceOptions.OrderableDBInstanceOption.N

特定の指示可能なインスタンスで使用できるオプション。

型: OrderableDBInstanceOption オブジェクトの配列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribePendingMaintenanceActions

サービス : Amazon DocumentDB (with MongoDB compatibility)

1 つ以上の保留中のメンテナンスアクションを含むリソース (例: インスタンス) のリストを返します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

Filters.Filter.N

保留中のメンテナンスアクションを返す 1 つ以上のリソースを指定するフィルター。

サポートされているフィルター:

- `db-cluster-id` - クラスター識別子とクラスターの Amazon リソースネーム (ARN) を受け入れます。結果リストには、これらの ARN で識別されるクラスターの保留中のメンテナンスアクションのみが含まれます。
- `db-instance-id` - インスタンス識別子とインスタンス ARN を受け入れます。結果リストには、これらの ARN で識別されるクラスターの保留中のメンテナンスアクションのみが含まれます。

タイプ: Filter オブジェクトの配列

必須: いいえ

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには `MaxRecords` で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

MaxRecords

レスポンスに含めるレコードの最大数。指定された `MaxRecords` の値よりも多くのレコードが存在する場合、ページ割りトークン (マーカー) がレスポンスに含まれるため、残りの結果を取得できます。

デフォルト: 100

制約: 最小 20、最大 100。

タイプ: 整数

必須: いいえ

ResourceIdentifier

保留中のメンテナンスアクションを返すリソースの ARN。

タイプ: 文字列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには `MaxRecords` で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

PendingMaintenanceActions.ResourcePendingMaintenanceActions.N

適用されるメンテナンスアクション。

型: ResourcePendingMaintenanceActions オブジェクトの配列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

ResourceNotFoundFault

指定されたリソース ID は見つかりませんでした。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

FailoverDBCluster

サービス : Amazon DocumentDB (with MongoDB compatibility)

クラスターのフェイルオーバーを強制実行します。

クラスターのフェイルオーバーにより、クラスター内のAmazon DocumentDB レプリカ（ 読み取り専用インスタンス) の1つをプライマリインスタンス (クラスターライター) に昇格されます。

プライマリインスタンスに障害が発生した場合、Amazon DocumentDB レプリカが存在すると、自動的にフェイルオーバーします。テストのため、プライマリインスタンスの失敗をシミュレートする場合は、フェイルオーバーを強制できます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterIdentifier

フェイルオーバーを強制するクラスター識別子。このパラメータは大文字と小文字が区別されません。

制約:

- 既存の `DBCluster` の識別子と一致する必要があります。

タイプ: 文字列

必須: いいえ

TargetDBInstanceIdentifier

プライマリインスタンスに昇格するインスタンスの名前。

クラスター内のAmazon DocumentDB レプリカのインスタンス識別子を指定する必要があります。例えば、`mydbcluster-replica1`。

型: 文字列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBCluster

クラスターに関する詳細情報。

型: DBCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

InvalidDBInstanceState

指定されたインスタンスは 利用可能な 状態ではありません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin

- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

FailoverGlobalCluster

サービス : Amazon DocumentDB (with MongoDB compatibility)

グローバルクラスターにフェイルオーバーが発生したときに、指定されたセカンダリ DB クラスターをそのグローバルクラスターのプライマリ DB クラスターに昇格させます。

このオペレーションにより、プライマリリージョンの地域的な災害などの計画外の事象に対応します。どちらのフェイルオーバー方法でも、フェイルオーバーイベントが発生する前に選択したセカンダリにレプリケートされていなかった書き込みトランザクションデータは失われる可能性があります。ただし、選択したセカンダリ DB クラスター上の DB インスタンスをプライマリライター DB インスタンスに昇格させるリカバリプロセスにより、データがトランザクション的に一貫した状態になることが保証されます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

GlobalClusterIdentifier

このオペレーションを適用する Amazon DocumentDB グローバルクラスターの識別子。識別子は、クラスターの作成時にユーザーによって割り当てられる一意のキーです。つまり、グローバルクラスターの名前です。

制約:

- 既存のグローバルクラスターの識別子と一致する必要があります。
- 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

必須: はい

TargetDbClusterIdentifier

そのグローバルクラスターのプライマリに昇格させるセカンダリ Amazon DocumentDB クラスターの識別子。Amazon DocumentDB が AWS リージョン内のクラスターを見つけることができるように、識別子に Amazon リソースネーム (ARN) を使用します。

制約:

- 既存のセカンダリクラスターの識別子と一致する必要があります。
- 最小長は 1 です。最大長は 255 です。

パターン: [A-Za-z][0-9A-Za-z-:._]*

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: [A-Za-z][0-9A-Za-z-:._]*

必須: はい

AllowDataLoss

このグローバルクラスターオペレーションのデータロスを許可するかどうかを指定します。データロスを許可すると、グローバルフェイルオーバーオペレーションがトリガーされます。

AllowDataLoss を指定しない場合、グローバルクラスターオペレーションはデフォルトでスイッチオーバーになります。

制約:

- Switchover パラメータと一緒に指定することはできません。

型: ブール値

必須: いいえ

Switchover

このグローバルデータベースクラスターをスイッチオーバーするかどうかを指定します。

制約:

- AllowDataLoss パラメータと一緒に指定することはできません。

型: ブール値

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

GlobalCluster

Amazon DocumentDB グローバルクラスターを表すデータ型。

型: GlobalCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

GlobalClusterNotFoundFault

`GlobalClusterIdentifier` は既存のグローバルクラスターを参照していません。

HTTP ステータスコード: 404

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

InvalidGlobalClusterStateFault

要求された操作は、クラスターがこの状態にある間は実行できません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2

- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListTagsForResource

サービス : Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB リソースのすべてのタグを一覧表示します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

ResourceName

一覧されるタグを持つ Amazon DocumentDB リソース。この値は Amazon リソースネーム (ARN) です。

タイプ: 文字列

必須: はい

Filters.Filter.N

このパラメータは、現在サポートされていません。

タイプ: Filter オブジェクトの配列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

TagList.Tag.N

1 つ以上のタグのリスト。

型: Tag オブジェクトの配列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

DBInstanceNotFound

`DBInstanceIdentifier` は既存のインスタンスを参照していません。

HTTP ステータスコード: 404

DBSnapshotNotFound

`DBSnapshotIdentifier` は既存のスナップショットを参照していません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ModifyDBCluster

サービス : Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB クラスターの設定を変更します。これらのパラメータとリクエストに新しい値を指定することで、1 つ以上のデータベース設定パラメータを変更できます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterIdentifier

変更するクラスターのクラスター識別子。このパラメータは大文字と小文字が区別されません。

制約:

- 既存の DBCluster の識別子と一致する必要があります。

タイプ: 文字列

必須: はい

AllowMajorVersionUpgrade

メジャーバージョンアップグレードを許可するかどうかを示す値。

制約:

- クラスターの現在のバージョンとは異なるメジャーバージョンであるEngineVersionパラメータの値を指定するときは、メジャーバージョンのアップグレードを許可する必要があります。
- 一部のパラメータはバージョン固有であるため、変更するには、インプレース MVU の完了後に新しい ModifyDBCluster API コールを実行する必要があります。

> **Note**
>
> MVU を実行すると、次のパラメータに直接影響します。
>
> - MasterUserPassword
> - NewDBClusterIdentifier
> - VpcSecurityGroupIds

- Port

型: ブール値

必須: いいえ

ApplyImmediately

クラスターの PreferredMaintenanceWindow の設定に関係なく、このリクエストの変更と保留中の変更をできるだけ早く非同期に適用するかどうかを指定する値。このパラメータを false に設定した場合、クラスターへの変更は次のメンテナンスウィンドウ中に適用されます。

ApplyImmediately パラメータは、NewDBClusterIdentifier 値と MasterUserPassword 値のみに影響します。このパラメータ値を false に設定すると、NewDBClusterIdentifier 値と MasterUserPassword 値への変更は次のメンテナンスウィンドウ中に適用されます。他のすべての変更は、ApplyImmediately パラメータの値に関係なく、即時に適用されます。

デフォルト: false

タイプ: ブール値

必須: いいえ

BackupRetentionPeriod

自動バックアップを保管する日数。最小値 1 を指定しなければなりません。

デフォルト: 1

制約:

- 1 ~ 35 の値にする必要があります。

タイプ: 整数

必須: いいえ

CloudwatchLogsExportConfiguration

特定のインスタンスまたはクラスターについて、Amazon CloudWatch Logs へのエクスポートを有効にするためのログタイプの設定。EnableLogTypes および DisableLogTypes の配列では、CloudWatch Logs にエクスポートする (または、エクスポートしない) ログを決定します。

型: CloudwatchLogsExportConfiguration オブジェクト

必須: いいえ

DBClusterParameterGroupName

クラスターに使用するクラスターパラメータグループの名前。

タイプ: 文字列

必須: いいえ

DeletionProtection

このクラスターを削除できるかどうかを指定します。DeletionProtection を有効にすると、クラスターが変更され DeletionProtection が無効にならない限り、クラスターを削除することができません。DeletionProtection はクラスターが誤って削除されるのを防ぎます。

型: ブール値

必須: いいえ

EngineVersion

アップグレードするデータベースエンジンのバージョン番号。このパラメータを変更すると、機能停止が発生します。ApplyImmediately パラメータが true に設定されされない限り、変更は次のメンテナンスウィンドウ中に適用されます。

Amazon DocumentDB に利用可能なエンジンバージョンをすべて一覧表示するには、次のコマンドを使用します。

```
aws docdb describe-db-engine-versions --engine docdb --query
"DBEngineVersions[].EngineVersion"
```

タイプ: 文字列

必須: いいえ

ManageMasterUserPassword

Amazon Web Services Secrets Manager でマスターユーザーパスワードを管理するかどうかを指定します。クラスターが Amazon Web Services Secrets Manager でマスターユーザーパスワードを管理していない場合は、この管理を有効にできます。この場合、 を指定することはできませんMasterUserPassword。クラスターが Amazon Web Services Secrets Manager でマスターユーザーパスワードを既に管理しており、マスターユーザーパスワードが Amazon Web Services Secrets Manager で管理されないように指定している場合は、 を指定する必要がありま

すMasterUserPassword。この場合、Amazon DocumentDB はシークレットを削除し、 で指定されたマスターユーザーの新しいパスワードを使用しますMasterUserPassword。

型: ブール値

必須: いいえ

MasterUserPassword

マスターデータベースユーザーのパスワードです。このパスワードをには、スラッシュ (/)、二重引用符 (")、または「at」記号 (@) 以外の印刷可能な ASCII 文字を含むことができます。

制約: 8 ~ 100 文字の長さの英数字である必要があります。

タイプ: 文字列

必須: いいえ

MasterUserSecretKmsKeyId

Amazon Web Services Secrets Manager で自動的に生成および管理されるシークレットを暗号化するための Amazon Web Services KMS キー識別子。

この設定は、次の条件の両方が満たされている場合にのみ有効です。

- クラスターは、Amazon Web Services Secrets Manager でマスターユーザーパスワードを管理しません。クラスターが Amazon Web Services Secrets Manager でマスターユーザーパスワードを既に管理している場合、シークレットの暗号化に使用される KMS キーを変更することはできません。
- ManageMasterUserPassword が Amazon Web Services Secrets Manager でマスターユーザーパスワードを管理できるようにします。をオンにManageMasterUserPasswordしていて、 を指定しない場合MasterUserSecretKmsKeyId、KMS aws/secretsmanager キーを使用してシークレットを暗号化します。シークレットが別の Amazon Web Services アカウントにある場合、KMS aws/secretsmanager キーを使用してシークレットを暗号化することはできません。また、カスタマーマネージド KMS キーを使用する必要があります。

Amazon Web Services KMS キー識別子は、KMS キーのキー ARN、キー ID、エイリアス ARN、またはエイリアス名です。別の Amazon Web Services アカウントで KMS キーを使用するには、キー ARN またはエイリアス ARN を指定します。

Amazon Web Services アカウントにはデフォルトの KMS キーがあります。Amazon Web Services アカウントには、Amazon Web Services リージョンごとに異なるデフォルトの KMS キーがあります。

タイプ: 文字列

必須: いいえ

NewDBClusterIdentifier

クラスターの名前を変更する場合のクラスターの新しいクラスター識別子。この値は小文字で保存されます。

制約:

- 1 ~ 63 の文字、数字またはハイフンを使用する必要があります。
- 1 字目は文字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

例: `my-cluster2`

タイプ: 文字列

必須: いいえ

Port

クラスターが接続を受け入れるポート番号。

制限: `1150` ~ `65535` の値である必要があります。

デフォルト : 元のクラスターと同じポート。

タイプ: 整数

必須: いいえ

PreferredBackupWindow

`BackupRetentionPeriod` パラメータを使用して自動バックアップが有効になっている場合に、自動バックアップが作成される毎日の時間範囲。

デフォルトは、それぞれ 8 時間の時間ブロックからランダムに選択された 30 分のウィンドウです AWS リージョン。

制約:

- `hh24:mi-hh24:mi` の形式である必要があります。
- 時間は協定世界時 (UTC) である必要があります。

- 必要なメンテナンス期間と競合してはいけません。
- 少なくとも 30 分以上必要です。

タイプ: 文字列

必須: いいえ

PreferredMaintenanceWindow

週 1 回のシステムメンテナンスを実行できる時間帯 (世界標準時 (UTC))。

形式: `ddd:hh24:mi-ddd:hh24:mi`

デフォルトは、それぞれ 8 時間の時間ブロックからランダムに選択された 30 分のウィンドウであり AWS リージョン、ランダムな曜日に発生します。

有効な曜日: 月、火、水、木、金、土、日

制約: 最小限の 30 分単位のウィンドウ。

タイプ: 文字列

必須: いいえ

RotateMasterUserPassword

Amazon Web Services Secrets Manager によって管理されるシークレットをマスターユーザーパスワードにローテーションするかどうかを指定します。

この設定は、マスターユーザーパスワードがクラスターの Amazon Web Services Secrets Manager の Amazon DocumentDB によって管理されている場合にのみ有効です。シークレット値には、更新されたパスワードが含まれます。

制約: マスターユーザーのパスワードを更新するときは、すぐに変更を適用する必要があります。

型: ブール値

必須: いいえ

ServerlessV2ScalingConfiguration

Amazon DocumentDB Serverless クラスターのスケーリング設定が含まれます。

型: ServerlessV2ScalingConfiguration オブジェクト

必須: いいえ

StorageType

DB クラスターに関連付けるストレージタイプ。

Amazon DocumentDB クラスターのストレージタイプの詳細については、「Amazon DocumentDB 開発者ガイド」の「クラスターストレージ設定」を参照してください。

ストレージタイプの有効な値 - `standard | iopt1`

デフォルト値は `standard` です

タイプ: 文字列

必須: いいえ

VpcSecurityGroupIds.VpcSecurityGroupId.N

クラスターに関連付けられる仮想プライベートクラウド（VPC）セキュリティグループのリスト。

型: 文字列の配列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBCluster

クラスターに関する詳細情報。

型: DBCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterAlreadyExistsFault

指定された識別子を持つクラスターをすでに持っています。

HTTP ステータスコード: 400

DBClusterNotFoundFault

DBClusterIdentifier は既存のクラスターを参照していません。

HTTP ステータスコード: 404

DBClusterParameterGroupNotFound

DBClusterParameterGroupName は既存のパラメータグループを参照していません。

HTTP ステータスコード: 404

DBSubnetGroupNotFoundFault

DBSubnetGroupName は既存のサブネットグループを参照していません。

HTTP ステータスコード: 404

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

InvalidDBInstanceState

指定されたインスタンスは 利用可能な 状態ではありません。

HTTP ステータスコード: 400

InvalidDBSecurityGroupState

セキュリティグループの状態では削除できません。

HTTP ステータスコード: 400

InvalidDBSubnetGroupStateFault

サブネットグループは使用中のため削除できません。

HTTP ステータスコード: 400

InvalidSubnet

要求されたサブネットが無効であるか、共通の仮想プライベートクラウド（VPC）内に存在しない複数のサブネットが要求されました。

HTTP ステータスコード: 400

InvalidVPCNetworkStateFault

サブネットグループは、作成後に変更されたため、すべてのアベイラビリティーゾーンがカバーされません。

HTTP ステータスコード: 400

StorageQuotaExceeded

リクエストにより、すべてのインスタンスで使用可能な許容ストレージ容量を超過します。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ModifyDBClusterParameterGroup

サービス : Amazon DocumentDB (with MongoDB compatibility)

クラスターパラメータグループのパラメータを変更します。複数のパラメータを変更するには、以下のリストを送信します: `ParameterName`、`ParameterValue`、`ApplyMethod`。最大 20 個のパラメータを単一のリクエストで修正できます。

> **Note**
>
> 動的パラメータの変更は直ちに適用されます。静的パラメータの変更の場合、変更を反映させる前に再起動またはメンテナンスウィンドウが必要です。

> **Important**
>
> クラスターパラメータグループの作成後、デフォルトのパラメータグループとしてそのクラスターパラメータグループを使用する最初のクラスターが作成されるまで、5 分以上かかります。その間に、Amazon DocumentDB が作成アクションを完了させ、パラメータグループが新しいクラスターのデフォルトとして使用されます。このステップは、クラスターのデフォルトデータベースの作成時に必須になるパラメータ (`character_set_database` パラメータにより定義されるデフォルトデータベースの文字セットなど) に特に重要です。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterParameterGroupName

変更するクラスターパラメータグループの名前。

タイプ: 文字列

必須: はい

Parameters.Parameter.N

変更するクラスターパラメータグループ内のパラメータのリスト。

型: Parameter オブジェクトの配列

必須: はい

レスポンス要素

以下の要素がサービスによって返されます。

DBClusterParameterGroupName

クラスターパラメータグループの名前。

制約:

- 1 ~ 255 の英数字である必要があります。
- 1 字目は文字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

> **Note**
> この値は小文字で保存されます。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBParameterGroupNotFound

`DBParameterGroupName` は既存のパラメータグループを参照していません。

HTTP ステータスコード: 404

InvalidDBParameterGroupState

パラメータグループが使用中であるか、無効な状態です。パラメータグループを削除しようとする場合、パラメータグループがこの状態のときは削除できません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ModifyDBClusterSnapshotAttribute

サービス : Amazon DocumentDB (with MongoDB compatibility)

属性および値を、手動クラスタースナップショットに追加するか、削除します。

手動クラスタースナップショットを他の と共有するには AWS アカウント、 `restore`として を指定し`AttributeName`、 `ValuesToAdd`パラメータを使用して、手動クラスタースナップショットの復元 AWS アカウント が許可されている の IDs のリストを追加します。値 `all` を使用して、手動クラスタースナップショットを公開します。これは、すべての AWS アカウントによってコピーまたは復元できることを意味します。すべての `all` に利用させたくないプライベート情報を含む手動クラスタースナップショットの AWS アカウント値は追加しません。手動クラスタースナップショットが暗号化されている場合は、 `ValuesToAdd`パラメータの AWS アカウント IDsのリストを指定することによってのみ共有できます。この場合、そのパラメータの値として `all` を使用することはできません。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

AttributeName

変更するクラスタースナップショット属性名。

手動クラスタースナップショットをコピーまたは復元 AWS アカウント する他のユーザーの認可を管理するには、この値を に設定します`restore`。

タイプ: 文字列

必須: はい

DBClusterSnapshotIdentifier

属性を変更するクラスタースナップショットの識別子。

タイプ: 文字列

必須: はい

ValuesToAdd.AttributeValue.N

`AttributeName` で指定された属性に追加するクラスタースナップショット属性のリスト。

手動クラスタースナップショットのコピーまたは復元 AWS アカウント を他のユーザーに許可するには、1 つ以上の AWS アカウント IDs を含めるようにこのリストを設定します。手動クラスタースナップショットを復元可能にするには AWS アカウント、 に設定しますall。すべての all に利用させたくないプライベート情報を含む手動クラスタースナップショットの AWS アカウント値は追加しません。

型: 文字列の配列

必須: いいえ

ValuesToRemove.AttributeValue.N

AttributeName で指定された属性から削除するクラスタースナップショット属性のリスト。

手動クラスタースナップショットをコピーまたは復元 AWS アカウント する他のユーザーの認可を削除するには、1 つ以上の AWS アカウント 識別子を含めるようにこのリストを設定します。クラスタースナップショットをコピーまたは復元 AWS アカウント する の認可を削除するには、に設定しますall。を指定してもall、アカウント ID AWS アカウント が restore 属性に明示的に追加された は、手動クラスタースナップショットをコピーまたは復元できます。

型: 文字列の配列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBClusterSnapshotAttributesResult

クラスタースナップショットに関連付けられている属性の詳細情報。

型: DBClusterSnapshotAttributesResult オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterSnapshotNotFoundFault

DBClusterSnapshotIdentifier は既存のクラスタースナップショットを参照していません。

HTTP ステータスコード: 404

InvalidDBClusterSnapshotStateFault

指定された値は有効なクラススナップショット状態ではありません。

HTTP ステータスコード: 400

SharedSnapshotQuotaExceeded

手動 DB スナップショットを共有できるアカウントの最大数を超えました。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ModifyDBInstance

サービス : Amazon DocumentDB (with MongoDB compatibility)

インスタンスの設定を変更します。これらのパラメータとリクエストに新しい値を指定することで、1 つ以上のデータベース設定パラメータを変更できます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBInstanceIdentifier

インスタンス識別子。この値は小文字で保存されます。

制約:

- 既存の DBInstance の識別子と一致する必要があります。

タイプ: 文字列

必須: はい

ApplyImmediately

インスタンスの PreferredMaintenanceWindow の設定に関係なく、このリクエストの変更と保留中の変更をできるだけ早く非同期に適用するかどうかを指定します。

このパラメータを false に設定した場合、クラスターへの変更は次のメンテナンスウィンドウ中に適用されます。パラメータの変更によっては、機能停止を引き起こす可能性があり、次回の再起動時に適用されます。

デフォルト: false

タイプ: ブール値

必須: いいえ

AutoMinorVersionUpgrade

このパラメータは Amazon DocumentDB には適用されません。Amazon DocumentDB は、設定値に関係なく、マイナーバージョンのアップグレードを実行しません。

型: ブール値

必須: いいえ

CACertificateIdentifier

インスタンスに関連付ける必要のある証明書を示します。

タイプ: 文字列

必須: いいえ

CertificateRotationRestart

SSL/TLS 証明書をローテーションするときに DB インスタンスが再起動されるかどうかを指定します。

デフォルトでは、SSL/TLS 証明書をローテーションすると DB インスタンスが再起動されます。DB インスタンスが再起動されるまで、証明書は更新されません。

> **Important**
>
> DB インスタンスへの接続に SSL/TLS を使用しない場合のみ、このパラメータを設定します。

SSL/TLS を使用して DB インスタンスに接続する場合は、「Amazon DocumentDB 開発者ガイド」の「Amazon DocumentDB TLS 証明書の更新」と「転送中のデータの暗号化」を参照してください。

型: ブール値

必須: いいえ

CopyTagsToSnapshot

すべてのタグを DB インスタンスから DB インスタンスのスナップショットにコピーするかどうかを示す値。デフォルトでは、タグはコピーされません。

型: ブール値

必須: いいえ

DBInstanceClass

インスタンスの新しいコンピューティング能力とメモリ容量 (例: `db.r5.large`)。すべてのインスタンスクラスがすべての AWS リージョンで利用できるわけではありません。

インスタンスクラスを変更すると、変更中に機能停止が発生します。ApplyImmediately がこのリクエストの true として指定されない限り、変更は次のメンテナンスウィンドウ中に適用されます。

デフォルト: 既存の設定を仕様。

タイプ: 文字列

必須: いいえ

EnablePerformanceInsights

DB インスタンスで Performance Insights を有効にするかどうかを示す値。詳細については、「Amazon Performance Insights の使用」を参照してください。

型: ブール値

必須: いいえ

NewDBInstanceIdentifier

インスタンス名を変更する場合のインスタンスの新しいインスタンス識別子。インスタンス識別子を変更し、Apply Immediately を true に設定するとインスタンスの再起動がすぐに実行されます。Apply Immediately を false に設定した場合、次のメンテナンスウィンドウ中に発生します。この値は小文字で保存されます。

制約:

- 1 ~ 63 の文字、数字またはハイフンを使用する必要があります。
- 1 字目は文字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

例: mydbinstance

タイプ: 文字列

必須: いいえ

PerformanceInsightsKMSKeyId

Performance Insights データを暗号化するための AWS KMS キー識別子。

AWS KMS キー識別子は、KMS キーのキー ARN、キー ID、エイリアス ARN、またはエイリアス名です。

PerformanceInsightsKMSKeyId に値を指定しない場合、Amazon DocumentDB はデフォルトの KMS キーを使用します。Amazon Web Services アカウントにはデフォルトの KMS キーがあります。Amazon Web Services アカウントには Amazon Web Services リージョンごとに異なるデフォルトの KMS キーがあります。

タイプ: 文字列

必須: いいえ

PreferredMaintenanceWindow

システムメンテナンスが発生する可能性がある週単位の時間範囲 (UTC) で、停止する可能性があります。次の場合を除き、このパラメータを変更しても機能停止にはならず、変更はできるだけ早く非同期的に適用されます。再起動を引き起こす保留中のアクションがあり、メンテナンスウィンドウが現在の時刻を含めて変更された場合、このパラメータを変更するとインスタンスが再起動されます。このウィンドウを現在の時刻に変更した場合、保留中の変更が確実に適用されるには、現在の時刻からウィンドウの終了まで 30 分以上必要です。

デフォルト: 既存の設定を使用。

形式: `ddd:hh24:mi-ddd:hh24:mi`

有効な曜日: 月、火、水、木、金、土、日

制約: 30 分以上にする必要があります。

タイプ: 文字列

必須: いいえ

PromotionTier

既存のプライマリインスタンスの障害後に、Amazon DocumentDB レプリカがプライマリインスタンスに昇格する順序を指定する値。

デフォルト: 1

有効な値: 0 ~ 15

タイプ: 整数

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBInstance

インスタンスに関する詳細情報。

型: DBInstance オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AuthorizationNotFound

指定された CIDRIP または Amazon EC2 セキュリティグループは、指定されたセキュリティグループに対して権限がありません。

Amazon DocumentDB には、IAM を使用してお客様に代わって必要なアクションを実行する権限がない場合もあります。

HTTP ステータスコード: 404

CertificateNotFound

`CertificateIdentifier` は既存の証明書を参照していません。

HTTP ステータスコード: 404

DBInstanceAlreadyExists

指定された識別子を持つインスタンスをすでに持っています。

HTTP ステータスコード: 400

DBInstanceNotFound

`DBInstanceIdentifier` は既存のインスタンスを参照していません。

HTTP ステータスコード: 404

DBParameterGroupNotFound

`DBParameterGroupName` は既存のパラメータグループを参照していません。

HTTP ステータスコード: 404

DBSecurityGroupNotFound

DBSecurityGroupName は既存のセキュリティグループを参照していません。

HTTP ステータスコード: 404

DBUpgradeDependencyFailure

依存するリソースが変更できないため、アップグレードに失敗しました。

HTTP ステータスコード: 400

InsufficientDBInstanceCapacity

指定されたインスタンスクラスは、指定されたアベイラビリティーゾーンで利用できません。

HTTP ステータスコード: 400

InvalidDBInstanceState

指定されたインスタンスは 利用可能な 状態ではありません。

HTTP ステータスコード: 400

InvalidDBSecurityGroupState

セキュリティグループの状態では削除できません。

HTTP ステータスコード: 400

InvalidVPCNetworkStateFault

サブネットグループは、作成後に変更されたため、すべてのアベイラビリティーゾーンがカバーされません。

HTTP ステータスコード: 400

StorageQuotaExceeded

リクエストにより、すべてのインスタンスで使用可能な許容ストレージ容量を超過します。

HTTP ステータスコード: 400

StorageTypeNotSupported

指定された StorageType のストレージはDBインスタンスに関連付けられません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ModifyDBSubnetGroup

サービス : Amazon DocumentDB (with MongoDB compatibility)

既存サブネットグループを変更します。サブネットグループには、 AWS リージョン内の 2 つ以上のアベイラビリティゾーンに 1 つ以上のサブネットが含まれる必要があります。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBSubnetGroupName

サブネットグループの名前。この値は小文字で保存されます。デフォルトのサブネットグループを変更することはできません。

制約: 既存の DBSubnetGroup の名前と一致する必要があります。デフォルト値を使用することはできません。

例: `mySubnetgroup`

タイプ: 文字列

必須: はい

SubnetIds.SubnetIdentifier.N

サブネットグループの Amazon EC2 サブネット ID。

型: 文字列の配列

必須: はい

DBSubnetGroupDescription

サブネットグループの説明。

タイプ: 文字列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBSubnetGroup

サブネットグループに関する詳細情報。

型: DBSubnetGroup オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBSubnetGroupDoesNotCoverEnoughAZs

アベイラビリティーゾーンが 1 つしかない場合を除き、サブネットグループ内のサブネットは 2 つ以上のアベイラビリティゾーンをカバーする必要があります。

HTTP ステータスコード: 400

DBSubnetGroupNotFoundFault

`DBSubnetGroupName` は既存のサブネットグループを参照していません。

HTTP ステータスコード: 404

DBSubnetQuotaExceededFault

リクエストにより、サブネットグループのサブネットの許容数を超過します。

HTTP ステータスコード: 400

InvalidSubnet

要求されたサブネットが無効であるか、共通の仮想プライベートクラウド（VPC）内に存在しない複数のサブネットが要求されました。

HTTP ステータスコード: 400

SubnetAlreadyInUse

サブネットはアベイラビリティーゾーンにすでに使われています。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ModifyEventSubscription

サービス : Amazon DocumentDB (with MongoDB compatibility)

既存の Amazon DocumentDB イベント通知サブスクリプションを変更します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

SubscriptionName

Amazon DocumentDB イベント通知サブスクリプション名。

タイプ: 文字列

必須: はい

Enabled

ブール値。サブスクリプションを有効にする場合は `true` に設定します。

型: ブール値

必須: いいえ

EventCategories.EventCategory.N

サブスクライブする `SourceType` のイベントカテゴリのリスト。

型: 文字列の配列

必須: いいえ

SnsTopicArn

イベント通知用に作成された SNS トピックの Amazon リソースネーム (ARN)。ARN は、トピックを作成してそれをサブスクライブするときに Amazon SNS によって作成されます。

タイプ: 文字列

必須: いいえ

SourceType

イベントを生成しているソースの種類。例えば、インスタンスが生成したイベントの通知を受ける場合は、このパラメータを db-instance に設定します。この値を指定しない場合、すべてのイベントが返されます。

有効な値: db-instance、db-parameter-group、db-security-group

タイプ: 文字列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

EventSubscription

サブスクライブしたイベントの詳細情報。

型: EventSubscription オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

EventSubscriptionQuotaExceeded

イベントサブスクリプション数が最大に達しました。

HTTP ステータスコード: 400

SNSInvalidTopic

Amazon SNS は、指定されたトピックに問題があると回答しました。

HTTP ステータスコード: 400

SNSNoAuthorization

SNS トピックの Amazon リソースネーム (ARN) に発行する権限がありません。

HTTP ステータスコード: 400

SNSTopicArnNotFound

SNS トピックの Amazon リソースネーム (ARN) が存在しません。

HTTP ステータスコード: 404

SubscriptionCategoryNotFound

指定されたカテゴリは存在しません。

HTTP ステータスコード: 404

SubscriptionNotFound

サブスクリプション名が存在しません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ModifyGlobalCluster

サービス : Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB グローバルクラスターの設定を変更します。リクエストでこれらのパラメータと新しい値を指定すると、1 つ以上の設定パラメータ (例: 削除保護) またはグローバルクラスター識別子を変更できます。

> **Note**
>
> このアクションは Amazon DocumentDB クラスターにのみ適用されます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

GlobalClusterIdentifier

変更するグローバルクラスターの識別子。このパラメータでは大文字と小文字は区別されません。

制約:

- 既存のグローバルクラスターの識別子と一致する必要があります。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

必須: はい

DeletionProtection

グローバルクラスターで削除保護を有効にするかどうかを指定します。削除保護が有効な場合、グローバルクラスターは削除できません。

型: ブール値

必須: いいえ

NewGlobalClusterIdentifier

グローバルクラスターを変更するときのグローバルクラスターの新しい識別子。この値は小文字で保存されます。

- 1 ~ 63 の文字、数字またはハイフンを使用する必要があります。

  1 字目は文字である必要があります

  文字列の最後にハイフンを使用したり、ハイフンを 2 つ続けて使用したりすることはできません。

例: `my-cluster2`

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

GlobalCluster

Amazon DocumentDB グローバルクラスターを表すデータ型。

型: GlobalCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

GlobalClusterNotFoundFault

`GlobalClusterIdentifier` は既存のグローバルクラスターを参照していません。

HTTP ステータスコード: 404

InvalidGlobalClusterStateFault

要求された操作は、クラスターがこの状態にある間は実行できません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

RebootDBInstance

サービス : Amazon DocumentDB (with MongoDB compatibility)

通常はメンテナンスのためにインスタンスを再起動する必要があります。例えば、特定の変更を行う場合や、インスタンスに関連付けられたクラスターパラメーターグループを変更した場合は、変更を有効にするためにインスタンスを再起動する必要があります。

インスタンスを再起動すると、データベースエンジンサービスが再起動されます。インスタンスを再起動すると一時的に機能停止になります。その間、DB インスタンスのステータスは [rebooting] に設定されます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBInstanceIdentifier

インスタンス識別子。このパラメータは小文字で保存されます。

制約:

- 既存の `DBInstance` の識別子と一致する必要があります。

タイプ: 文字列

必須: はい

ForceFailover

`true` の場合、再起動はマルチ AZ フェイルオーバーによって行われます。

制約: インスタンスがマルチ AZ 用に設定されていない場合、`true` を指定することはできません。

型: ブール値

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBInstance

インスタンスに関する詳細情報。

型: DBInstance オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBInstanceNotFound

`DBInstanceIdentifier` が既存の DB インスタンスを参照していません。

HTTP ステータスコード: 404

InvalidDBInstanceState

指定されたインスタンスは 利用可能な 状態ではありません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

RemoveFromGlobalCluster

サービス : Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB セカンダリクラスターをグローバルクラスターからデタッチします。クラスターは、読み取り専用で、別のリージョンのプライマリからデータを受信するのではなく、読み取り・書き込み機能を備えたスタンドアロンクラスターになります。

> **Note**
>
> このアクションは Amazon DocumentDB クラスターにのみ適用されます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DbClusterIdentifier

Amazon DocumentDB グローバルクラスターからデタッチされたクラスターを識別する Amazon リソースネーム (ARN)。

タイプ: 文字列

必須: はい

GlobalClusterIdentifier

Amazon DocumentDB グローバルクラスターからデタッチするクラスター識別子。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

必須: はい

レスポンス要素

以下の要素がサービスによって返されます。

GlobalCluster

Amazon DocumentDB グローバルクラスターを表すデータ型。

型: GlobalCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

GlobalClusterNotFoundFault

`GlobalClusterIdentifier` は既存のグローバルクラスターを参照していません。

HTTP ステータスコード: 404

InvalidGlobalClusterStateFault

要求された操作は、クラスターがこの状態にある間は実行できません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin

- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

RemoveSourceIdentifierFromSubscription

サービス : Amazon DocumentDB (with MongoDB compatibility)

既存の Amazon DocumentDB イベント通知サブスクリプションからソース識別子を削除します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

SourceIdentifier

インスタンス用のインスタンス識別子、セキュリティグループ名など、サブスクリプションから削除されるソース識別子。

タイプ: 文字列

必須: はい

SubscriptionName

ソース識別子を削除する Amazon DocumentDB イベント通知サブスクリプション名。

タイプ: 文字列

必須: はい

レスポンス要素

以下の要素がサービスによって返されます。

EventSubscription

サブスクライブしたイベントの詳細情報。

型: EventSubscription オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

SourceNotFound

リクエストされたリソースが見つかりませんでした。

HTTP ステータスコード: 404

SubscriptionNotFound

サブスクリプション名が存在しません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

RemoveTagsFromResource

サービス : Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB リソースからメタデータタグを削除します。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

ResourceName

タグが削除されたAmazon DocumentDB リソース。この値は Amazon リソースネーム (ARN) です。

タイプ: 文字列

必須: はい

TagKeys.member.N

削除するタグのタグキー (名前)。

型: 文字列の配列

必須: はい

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

DBInstanceNotFound

`DBInstanceIdentifier` は既存のインスタンスを参照していません。

HTTP ステータスコード: 404

DBSnapshotNotFound

`DBSnapshotIdentifier` は既存のスナップショットを参照していません。

HTTP ステータスコード: 404

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ResetDBClusterParameterGroup

サービス : Amazon DocumentDB (with MongoDB compatibility)

クラスターパラメータグループのパラメータをデフォルト値に変更します。特定のパラメータをリセットするには、ParameterName と ApplyMethod のリストを送信します。クラスターパラメータグループ全体をリセットするには、DBClusterParameterGroupName と ResetAllParameters パラメータを指定します。

グループ全体をリセットすると、動的パラメータは直ちに更新され、静的パラメータは pending-reboot に設定され、次回のDB インスタンス再起動時に有効になります。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterParameterGroupName

リセットするクラスターパラメータグループの名前。

タイプ: 文字列

必須: はい

Parameters.Parameter.N

デフォルト値にリセットするクラスターパラメータグループ内のパラメータ名のリスト。ResetAllParameters パラメータが true に設定されている場合は、このパラメータを設定することはできません。

タイプ: Parameter オブジェクトの配列

必須: いいえ

ResetAllParameters

true に設定する場合はクラスターパラメータグループ内のすべてのパラメータがデフォルト値にリセットされ、それ以外の場合は false に設定される値。Parameters パラメータに指定されたパラメータ名のリストがある場合、このパラメータは使用できません。

型: ブール値

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBClusterParameterGroupName

クラスターパラメータグループの名前。

制約:

- 1 ~ 255 の英数字である必要があります。
- 1 字目は文字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

> **Note**
>
> この値は小文字で保存されます。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBParameterGroupNotFound

`DBParameterGroupName` は既存のパラメータグループを参照していません。

HTTP ステータスコード: 404

InvalidDBParameterGroupState

パラメータグループが使用中であるか、無効な状態です。パラメータグループを削除しようとする場合、パラメータグループがこの状態のときは削除できません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

RestoreDBClusterFromSnapshot

サービス : Amazon DocumentDB (with MongoDB compatibility)

スナップショットまたはクラスタースナップショットから新しいクラスターを作成します。

スナップショットを指定した場合、ターゲットクラスターは、デフォルト設定とデフォルトセキュリティグループを使用してソース DB スナップショットから作成されます。

クラスタースナップショットを指定した場合、新しいクラスターが既定のセキュリティグループを使用して作成される場合以外は、ターゲットクラスターは元のソース DB クラスターと同じ構成でソースクラスターの復元ポイントから作成されます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterIdentifier

スナップショットまたはクラスタースナップショットから作成するクラスター名。このパラメータでは大文字と小文字は区別されません。

制約:

- 1 ~ 63 の文字、数字またはハイフンを使用する必要があります。
- 1 字目は文字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

例: `my-snapshot-id`

タイプ: 文字列

必須: はい

Engine

新しいクラスターに使用するデータベースエンジン。

デフォルト: ソースオブジェクトと同じ。

制約: ソースのエンジンに対応する必要があります。

タイプ: 文字列

必須: はい

SnapshotIdentifier

復元元のスナップショットまたはクラスタースナップショットの識別子。

クラスタースナップショットを指定するには、名前または Amazon リソースネーム (ARN) のいずれかを使用できます。ただし、スナップショットを指定するには ARN のみを使用できます。

制約:

- 既存のスナップショットの識別子と一致する必要があります。

タイプ: 文字列

必須: はい

AvailabilityZones.AvailabilityZone.N

DB クラスターのインスタンスを作成できる Amazon EC2 アベイラビリティーゾーンのリストを提供します。

型: 文字列の配列

必須: いいえ

DBClusterParameterGroupName

このクラスターに関連付ける DB クラスターパラメータグループの名前。

型:文字列　　必須: いいえ

この引数を省略すると、指定した DB クラスターのデフォルトのパラメータグループが使用されます。指定した場合、既存の DB クラスターのパラメータグループの名前と一致する必要があります。1 ~ 255 の文字、数字またはハイフンを使用する必要があります。名前は文字で開始し、ハイフン以外で終了する必要があります。ハイフンを 2 つ続けて使用することはできません。

タイプ: 文字列

必須: いいえ

DBSubnetGroupName

新しいクラスターに使用するサブネットグループ名。

制約: 指定した場合、既存の `DBSubnetGroup` の名前と一致する必要があります。

例: mySubnetgroup

タイプ: 文字列

必須: いいえ

DeletionProtection

このクラスターを削除できるかどうかを指定します。DeletionProtection を有効にすると、クラスターが変更され DeletionProtection が無効にならない限り、クラスターを削除することができません。DeletionProtection はクラスターが誤って削除されるのを防ぎます。

型: ブール値

必須: いいえ

EnableCloudwatchLogsExports.member.N

Amazon CloudWatch Logs にエクスポートするために有効にする必要がある口グタイプのリスト。

型: 文字列の配列

必須: いいえ

EngineVersion

新しいクラスターに使用するデータベースエンジンのバージョン。

タイプ: 文字列

必須: いいえ

KmsKeyId

DB スナップショットまたはクラススナップショットから暗号化されたクラスターを復元するときに使用する AWS KMS キー識別子。

AWS KMS キー識別子は、 AWS KMS 暗号化キーの Amazon リソースネーム (ARN) です。新しいクラスターの暗号化に使用される AWS KMS 暗号化キーを所有 AWS アカウント する と同じを持つクラスターを復元する場合は、 AWS KMS 暗号化 AWS KMS キーの ARN の代わりにキーエイリアスを使用できます。

KmsKeyId パラメータの値を指定しない場合は、以下のようになります。

- のスナップショットまたはクラスタースナップショットSnapshotIdentifierが暗号化されている場合、復元されたクラスターは、スナップショットまたはクラスタースナップショットの暗号化に使用された AWS KMS キーを使用して暗号化されます。
- SnapshotIdentifier のスナップショットまたはクラスタースナップショットが暗号化されていない場合、復元されたクラスターは暗号化されません。

タイプ: 文字列

必須: いいえ

Port

クラスターが接続を受け入れるポート番号。

制約: 1150 ~ 65535 の値である必要があります。

デフォルト : 元のクラスターと同じポート。

タイプ: 整数

必須: いいえ

ServerlessV2ScalingConfiguration

Amazon DocumentDB Serverless クラスターのスケーリング設定が含まれます。

型: ServerlessV2ScalingConfiguration オブジェクト

必須: いいえ

StorageType

DB クラスターに関連付けるストレージタイプ。

Amazon DocumentDB クラスターのストレージタイプの詳細については、「Amazon DocumentDB 開発者ガイド」の「クラスターストレージ設定」を参照してください。

ストレージタイプの有効な値 - standard | iopt1

デフォルト値は standard です

タイプ: 文字列

必須: いいえ

Tags.Tag.N

復元されたクラスターに割り当てられるタグ。

タイプ: Tag オブジェクトの配列

必須: いいえ

VpcSecurityGroupIds.VpcSecurityGroupId.N

クラスターに関連付けられる仮想プライベートクラウド（VPC）セキュリティグループのリスト。

型: 文字列の配列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBCluster

クラスターに関する詳細情報。

型: DBCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterAlreadyExistsFault

指定された識別子を持つクラスターをすでに持っています。

HTTP ステータスコード: 400

DBClusterQuotaExceededFault

クラスターの最大許容クォータに達したため、クラスターを作成できません。

HTTP ステータスコード: 403

DBClusterSnapshotNotFoundFault

DBClusterSnapshotIdentifier は既存のクラススナップショットを参照していません。

HTTP ステータスコード: 404

DBSnapshotNotFound

DBSnapshotIdentifier は既存のスナップショットを参照していません。

HTTP ステータスコード: 404

DBSubnetGroupNotFoundFault

DBSubnetGroupName は既存のサブネットグループを参照していません。

HTTP ステータスコード: 404

DBSubnetGroupNotFoundFault

DBSubnetGroupName は既存のサブネットグループを参照していません。

HTTP ステータスコード: 404

InsufficientDBClusterCapacityFault

クラスターには現在の操作に十分な容量がありません。

HTTP ステータスコード: 403

InsufficientStorageClusterCapacity

クラスターには現在の操作に十分な容量がありません。使用できるストレージがより多い別のアベイラビリティーゾーンを使用するようにサブネットグループを更新することで、このエラーを解決できる場合があります。

HTTP ステータスコード: 400

InvalidDBClusterSnapshotStateFault

指定された値は有効なクラススナップショット状態ではありません。

HTTP ステータスコード: 400

InvalidDBSnapshotState

スナップショットの状態では削除できません。

HTTP ステータスコード: 400

InvalidRestoreFault

仮想プライベートクラウド（VPC）のバックアップから VPC以外の DB インスタンスには復元できません。

HTTP ステータスコード: 400

InvalidSubnet

要求されたサブネットが無効であるか、共通の仮想プライベートクラウド（VPC）内に存在しない複数のサブネットが要求されました。

HTTP ステータスコード: 400

InvalidVPCNetworkStateFault

サブネットグループは、作成後に変更されたため、すべてのアベイラビリティーゾーンがカバーされません。

HTTP ステータスコード: 400

KMSKeyNotAccessibleFault

AWS KMS キーへのアクセス中にエラーが発生しました。

HTTP ステータスコード: 400

StorageQuotaExceeded

リクエストにより、すべてのインスタンスで使用可能な許容ストレージ容量を超過します。

HTTP ステータスコード: 400

StorageQuotaExceeded

リクエストにより、すべてのインスタンスで使用可能な許容ストレージ容量を超過します。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

RestoreDBClusterToPointInTime

サービス : Amazon DocumentDB (with MongoDB compatibility)

クラスターを任意の時点に復元します。ユーザーは `LatestRestorableTime` より前の任意の時点に最大 `BackupRetentionPeriod` 日間復元できます。ターゲットクラスターは、新しいクラスターがデフォルトのセキュリティグループで作成されない場合、元のクラスターと同じ構成で作成されます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterIdentifier

作成する新しいクラスターの名前。

制約:

- 1 ~ 63 の文字、数字またはハイフンを使用する必要があります。
- 1 字目は文字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

タイプ: 文字列

必須: はい

SourceDBClusterIdentifier

復元されるソースクラスターの識別子。

制約:

- 既存の `DBCluster` の識別子と一致する必要があります。

タイプ: 文字列

必須: はい

DBSubnetGroupName

新しいクラスターに使用するサブネットグループ名。

制約: 指定した場合、既存の `DBSubnetGroup` の名前と一致する必要があります。

例: mySubnetgroup

タイプ: 文字列

必須: いいえ

DeletionProtection

このクラスターを削除できるかどうかを指定します。DeletionProtection を有効にすると、クラスターが変更され DeletionProtection が無効にならない限り、クラスターを削除することができません。DeletionProtection はクラスターが誤って削除されるのを防ぎます。

型: ブール値

必須: いいえ

EnableCloudwatchLogsExports.member.N

Amazon CloudWatch Logs にエクスポートするために有効にする必要がある口グタイプのリスト。

型: 文字列の配列

必須: いいえ

KmsKeyId

暗号化されたクラスターを暗号化されたクラスターから復元するときに使用する AWS KMS キー識別子。

AWS KMS キー識別子は、 AWS KMS 暗号化キーの Amazon リソースネーム (ARN) です。新しいクラスターの暗号化に使用される AWS KMS 暗号化キーを所有 AWS アカウント する と同じを持つクラスターを復元する場合は、 AWS KMS 暗号化 AWS KMS キーの ARN の代わりにキーエイリアスを使用できます。

新しいクラスターに復元し、ソースクラスターの暗号化に使用される AWS KMS キーとは異なる AWS KMS キーを使用して新しいクラスターを暗号化できます。新しい DB クラスターは、KmsKeyIdパラメータで識別される AWS KMS キーで暗号化されます。

KmsKeyId パラメータの値を指定しない場合は、以下のようになります。

- クラスターが暗号化されている場合、復元されたクラスターは、ソースクラスターの暗号化に使用された AWS KMS キーを使用して暗号化されます。

- クラスターが暗号化されていない場合、復元されたクラスターは暗号化されません。

  DBClusterIdentifier が暗号化されていないクラスターを参照している場合、復元リクエストは拒否されます。

  タイプ: 文字列

  必須: いいえ

Port

  クラスターが接続を受け入れるポート番号。

  制約:1150 ~ 65535 の値である必要があります。

  デフォルト : エンジンのデフォルトポート。

  タイプ: 整数

  必須: いいえ

RestoreToTime

  クラスターを復元する日時。

  有効な値: 協定世界時 (UTC) 形式の時刻でなければなりません。

  制約:

- インスタンスの復元可能な最新時刻より前である必要があります。
- UseLatestRestorableTime パラメータを指定しない場合は、指定する必要があります。
- UseLatestRestorableTime パラメータが true である場合は、指定することはできません。
- RestoreType パラメータが copy-on-write である場合は、指定することはできません。

  例: 2015-03-07T23:45:00Z

  型: タイムスタンプ

  必須: いいえ

RestoreType

  実行する復元のタイプ。次のいずれかの値を指定できます。

- `full-copy` - 新しい DB クラスターは、ソース DB クラスターのフルコピーとして復元されます。
- `copy-on-write` - 新しい DB クラスターは、ソース DB クラスターのクローンとして復元されます。

制約 : ソース DB クラスターのエンジンバージョンが 1.11 より前である場合、`copy-on-write` を指定できません。

`RestoreType` 値を指定しない場合、新しい DB クラスターは、ソース DB クラスターのフルコピーとして復元されます。

タイプ: 文字列

必須: いいえ

ServerlessV2ScalingConfiguration

Amazon DocumentDB Serverless クラスターのスケーリング設定が含まれます。

型: ServerlessV2ScalingConfiguration オブジェクト

必須: いいえ

StorageType

DB クラスターに関連付けるストレージタイプ。

Amazon DocumentDB クラスターのストレージタイプの詳細については、「Amazon DocumentDB 開発者ガイド」の「クラスターストレージ設定」を参照してください。

ストレージタイプの有効な値 - `standard | iopt1`

デフォルト値は `standard` です

タイプ: 文字列

必須: いいえ

Tags.Tag.N

復元されたクラスターに割り当てられるタグ。

タイプ: Tag オブジェクトの配列

必須: いいえ

UseLatestRestorableTime

`true` に設定する場合はクラスターを最新の復元可能なバックアップ時間に復元し、それ以外の場合は `false` に設定される値。

デフォルト: `false`

制約: `RestoreToTime` パラメータを指定した場合は、指定することはできません。

型: ブール値

必須: いいえ

VpcSecurityGroupIds.VpcSecurityGroupId.N

新しいクラスターが属する VPC セキュリティグループのリスト。

型: 文字列の配列

必須: いいえ

レスポンス要素

以下の要素がサービスによって返されます。

DBCluster

クラスターに関する詳細情報。

型: DBCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterAlreadyExistsFault

指定された識別子を持つクラスターをすでに持っています。

HTTP ステータスコード: 400

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

DBClusterQuotaExceededFault

クラスターの最大許容クォータに達したため、クラスターを作成できません。

HTTP ステータスコード: 403

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier` は既存のクラススナップショットを参照していません。

HTTP ステータスコード: 404

DBSubnetGroupNotFoundFault

`DBSubnetGroupName` は既存のサブネットグループを参照していません。

HTTP ステータスコード: 404

InsufficientDBClusterCapacityFault

クラスターには現在の操作に十分な容量がありません。

HTTP ステータスコード: 403

InsufficientStorageClusterCapacity

クラスターには現在の操作に十分な容量がありません。使用できるストレージがより多い別のアベイラビリティーゾーンを使用するようにサブネットグループを更新することで、この工ラーを解決できる場合があります。

HTTP ステータスコード: 400

InvalidDBClusterSnapshotStateFault

指定された値は有効なクラススナップショット状態ではありません。

HTTP ステータスコード: 400

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

InvalidDBSnapshotState

スナップショットの状態では削除できません。

HTTP ステータスコード: 400

InvalidRestoreFault

仮想プライベートクラウド（VPC）のバックアップから VPC以外の DB インスタンスには復元できません。

HTTP ステータスコード: 400

InvalidSubnet

要求されたサブネットが無効であるか、共通の仮想プライベートクラウド（VPC）内に存在しない複数のサブネットが要求されました。

HTTP ステータスコード: 400

InvalidVPCNetworkStateFault

サブネットグループは、作成後に変更されたため、すべてのアベイラビリティーゾーンがカバーされません。

HTTP ステータスコード: 400

KMSKeyNotAccessibleFault

AWS KMS キーへのアクセス中にエラーが発生しました。

HTTP ステータスコード: 400

StorageQuotaExceeded

リクエストにより、すべてのインスタンスで使用可能な許容ストレージ容量を超過します。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2

- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

StartDBCluster

サービス : Amazon DocumentDB (with MongoDB compatibility)

`DBClusterIdentifier` で指定された停止中のクラスターを再起動します。詳細については、「Amazon DocumentDB クラスターの停止と起動」を参照してください。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterIdentifier

再開するクラスターの識別子。例: `docdb-2019-05-28-15-24-52`

タイプ: 文字列

必須: はい

レスポンス要素

以下の要素がサービスによって返されます。

DBCluster

クラスターに関する詳細情報。

型: DBCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

InvalidDBInstanceState

指定されたインスタンスは 利用可能な 状態ではありません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

StopDBCluster

サービス : Amazon DocumentDB (with MongoDB compatibility)

`DBClusterIdentifier` で指定された実行中のクラスターを停止します。クラスターは 利用可能な状態でなければなりません。詳細については、「Amazon DocumentDB クラスターの停止と起動」を参照してください。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

DBClusterIdentifier

停止するクラスターの識別子。例: `docdb-2019-05-28-15-24-52`

タイプ: 文字列

必須: はい

レスポンス要素

以下の要素がサービスによって返されます。

DBCluster

クラスターに関する詳細情報。

型: DBCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

InvalidDBInstanceState

指定されたインスタンスは 利用可能な 状態ではありません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

SwitchoverGlobalCluster

サービス : Amazon DocumentDB (with MongoDB compatibility)

指定されたセカンダリ Amazon DocumentDB クラスターをグローバルデータベースクラスターの新しいプライマリ Amazon DocumentDB クラスターに切り替えます。

リクエストパラメーター

すべてのアクションに共通のパラメータの詳細については、「共通パラメータ」を参照してください。

GlobalClusterIdentifier

切り替える Amazon DocumentDB グローバルデータベースクラスターの識別子。識別子は、クラスターの作成時にユーザーによって割り当てられる一意のキーです。つまり、グローバルクラスターの名前と言えます。このパラメータでは大文字と小文字は区別されません。

制約:

- 既存のグローバルクラスター (Amazon DocumentDB グローバルデータベース) の識別子と一致する必要があります。
- 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

必須: はい

TargetDbClusterIdentifier

グローバルデータベースクラスターの新しいプライマリに昇格させるセカンダリ Amazon DocumentDB クラスターの識別子。Amazon DocumentDB が AWS リージョン内のクラスターを見つけることができるように、識別子に Amazon リソースネーム (ARN) を使用します。

制約:

- 既存のセカンダリクラスターの識別子と一致する必要があります。
- 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

必須: はい

レスポンス要素

以下の要素がサービスによって返されます。

GlobalCluster

Amazon DocumentDB グローバルクラスターを表すデータ型。

型: GlobalCluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

DBClusterNotFoundFault

`DBClusterIdentifier` は既存のクラスターを参照していません。

HTTP ステータスコード: 404

GlobalClusterNotFoundFault

`GlobalClusterIdentifier` は既存のグローバルクラスターを参照していません。

HTTP ステータスコード: 404

InvalidDBClusterStateFault

クラスターは有効な状態ではありません。

HTTP ステータスコード: 400

InvalidGlobalClusterStateFault

要求された操作は、クラスターがこの状態にある間は実行できません。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Amazon DocumentDB Elastic Clusters

Amazon DocumentDB Elastic クラスターでは、以下のアクションがサポートされています。

- ApplyPendingMaintenanceAction
- CopyClusterSnapshot
- CreateCluster
- CreateClusterSnapshot
- DeleteCluster
- DeleteClusterSnapshot
- GetCluster
- GetClusterSnapshot
- GetPendingMaintenanceAction
- ListClusters
- ListClusterSnapshots

- ListPendingMaintenanceActions
- ListTagsForResource
- RestoreClusterFromSnapshot
- StartCluster
- StopCluster
- TagResource
- UntagResource
- UpdateCluster

ApplyPendingMaintenanceAction

サービス : Amazon DocumentDB Elastic Clusters

リソースに適用される保留中のメンテナンスアクションのタイプ。

リクエストの構文

```
POST /pending-action HTTP/1.1
Content-type: application/json

{
   "applyAction": "string",
   "applyOn": "string",
   "optInType": "string",
   "resourceArn": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

applyAction

リソースに適用する保留中のメンテナンスアクション。

有効なアクションは以下のとおりです。

- ENGINE_UPDATE
- ENGINE_UPGRADE
- SECURITY_UPDATE
- OS_UPDATE
- MASTER_USER_PASSWORD_UPDATE

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

必須: はい

optInType

オプトインリクエストのタイプを指定するか、オプトインクエストを元に戻す値。タイプ IMMEDIATE のオプトインリクエストは元に戻すことができません。

型: 文字列

有効な値 : IMMEDIATE | NEXT_MAINTENANCE | APPLY_ON | UNDO_OPT_IN

必須: はい

resourceArn

保留中のメンテナンスアクションが適用されるリソースの Amazon DocumentDB Amazon リソースネーム (ARN)。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

必須: はい

applyOn

保留中のメンテナンスアクションを適用する特定の日付。opt-in-typeが の場合は必須ですAPPLY_ON。形式: yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 256 です。

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "resourcePendingMaintenanceAction": {
      "pendingMaintenanceActionDetails": [
         {
            "action": "string",
```

```
            "autoAppliedAfterDate": "string",
            "currentApplyDate": "string",
            "description": "string",
            "forcedApplyDate": "string",
            "optInStatus": "string"
         }
      ],
      "resourceArn": "string"
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

resourcePendingMaintenanceAction

適用されている保留中のメンテナンスアクションの出力。

型: ResourcePendingMaintenanceAction オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

ConflictException

アクセス競合が発生しました。

HTTP ステータスコード: 409

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CopyClusterSnapshot

サービス : Amazon DocumentDB Elastic Clusters

Elastic クラスターのスナップショットをコピーします。

リクエストの構文

```
POST /cluster-snapshot/snapshotArn/copy HTTP/1.1
Content-type: application/json

{
   "copyTags": boolean,
   "kmsKeyId": "string",
   "tags": {
      "string" : "string"
   },
   "targetSnapshotName": "string"
}
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

snapshotArn

Elastic クラススナップショットの Amazon リソースネーム (ARN) 識別子。

必須: はい

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

targetSnapshotName

コピー元のクラススナップショットから作成される新しい Elastic クラススナップショットの識別子。このパラメータは大文字と小文字が区別されません。

制約:

- 1 ~ 63 の文字、数字またはハイフンを使用する必要があります。
- 1 字目は文字である必要があります。

- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

例: `elastic-cluster-snapshot-5`

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 63 です。

必須: はい

copyTags

コピー元のクラススナップショットからコピー先のクラススナップショットにすべてのタグをコピーする場合は `true` に設定します。デフォルトは `false` です。

型: ブール値

必須: いいえ

kmsKeyId

暗号化された Elastic クラススナップショットの AWS KMS キー ID。 AWS KMS キー ID は、Amazon リソースネーム (ARN)、 AWS KMS キー識別子、または AWS KMS 暗号化キーの KMS AWS キーエイリアスです。

AWS アカウントから暗号化された Elastic クラススナップショットをコピーする場合、 の値を指定`KmsKeyId`して、新しい S AWS KMS 暗号化キーでコピーを暗号化できます。`KmsKeyId` の値を指定しない場合、Elastic クラススナップショットのコピーはコピー元の Elastic クラススナップショットと同じ AWS KMS キーで暗号化されます。

暗号化されていない Elastic クラススナップショットをコピーして `KmsKeyId` パラメータに値を指定すると、エラーが返されます。

タイプ: 文字列

必須: いいえ

tags

Elastic クラススナップショットに割り当てるタグ。

型: 文字列から文字列へのマップ

キーの長さ制限: 最小長さは 1 です。最大長は 128 です。

キーパターン: (?!aws:)[a-zA-Z+-=._:/]+

値の長さの制限: 最小長は 0 です。最大長は 256 です。

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "snapshot": {
      "adminUserName": "string",
      "clusterArn": "string",
      "clusterCreationTime": "string",
      "kmsKeyId": "string",
      "snapshotArn": "string",
      "snapshotCreationTime": "string",
      "snapshotName": "string",
      "snapshotType": "string",
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

snapshot

特定のエラスティッククラスタースナップショットに関する情報を返します。

型: ClusterSnapshot オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

ConflictException

アクセス競合が発生しました。

HTTP ステータスコード: 409

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ServiceQuotaExceededException

アクションのサービスクォータを超えました。

HTTP ステータスコード: 402

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateCluster

サービス : Amazon DocumentDB Elastic Clusters

新しい Amazon DocumentDB エラスティッククラスターを作成し、そのクラスターの構造を返します。

リクエストの構文

```
POST /cluster HTTP/1.1
Content-type: application/json

{
   "adminUserName": "string",
   "adminUserPassword": "string",
   "authType": "string",
   "backupRetentionPeriod": number,
   "clientToken": "string",
   "clusterName": "string",
   "kmsKeyId": "string",
   "preferredBackupWindow": "string",
   "preferredMaintenanceWindow": "string",
   "shardCapacity": number,
   "shardCount": number,
   "shardInstanceCount": number,
   "subnetIds": [ "string" ],
   "tags": {
      "string" : "string"
   },
   "vpcSecurityGroupIds": [ "string" ]
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

adminUserName

Amazon DocumentDB エラスティッククラスター管理者の名前。

制約:

- 1～63 文字の英数字である必要があります。
- 1 字目は文字である必要があります。
- 予約語は使用できません。

タイプ: 文字列

必須: はい

adminUserPassword

Amazon DocumentDB エラスティッククラスター管理者のパスワード。パスワードは、印刷可能な任意の ASCII 文字で構成することができます。

制約:

- 8 ～ 100 文字で構成する必要があります。
- スラッシュ (/)、二重引用符 (「」)、または「at」記号 (@) を含むことはできません。

タイプ: 文字列

必須: はい

authType

エラスティッククラスターへのアクセスに使用するパスワードの取得先の特定に使用している認証タイプ。有効なタイプは、PLAIN_TEXT または SECRET_ARN です。

型: 文字列

有効な値：PLAIN_TEXT | SECRET_ARN

必須: はい

clusterName

新しいエラスティッククラスターの名前。このパラメータは小文字で保存されます。

制約:

- 1 ～ 63 の文字、数字またはハイフンを使用する必要があります。
- 1 字目は文字である必要があります。
- ハイフンを、文字列の最後に使用したり、2 つ続けて使用したりすることはできません。

例: `my-cluster`

タイプ: 文字列

必須: はい

shardCapacity

各エラスティッククラスターシャードに割り当てられている vCPU の数。最大数は 64 です。使用できる値は、2、4、8、16、32、64 です。

タイプ: 整数

必須: はい

shardCount

エラスティッククラスターに割り当てられているシャードの数。最大数は 32 です。

タイプ: 整数

必須: はい

backupRetentionPeriod

自動スナップショットが保持される日数。

タイプ: 整数

必須: いいえ

clientToken

エラスティッククラスターのクライアントトークン。

タイプ: 文字列

必須: いいえ

kmsKeyId

新しいエラスティッククラスターの暗号化に使用する KMS キー識別子。

KMS キー識別子は、KMS 暗号化キーの Amazon リソースネーム (ARN) です。KMS 暗号化キーを所有する Amazon アカウントと同じアカウントを使用して DB クラスターを作成する場合、KMS 暗号化キーの ARN の代わりに KMS キーのエイリアスを使用できます。

暗号化キーが指定されていない場合、Amazon DocumentDB は KMS がアカウント用に作成したデフォルトの暗号化キーを使用します。各アカウントは Amazon のリージョンごとにデフォルトの暗号化キーを持ちます。

タイプ: 文字列

必須: いいえ

preferredBackupWindow

backupRetentionPeriod に応じた、自動バックアップが有効な場合に毎日自動バックアップが作成される時間範囲を指定します。

タイプ: 文字列

必須: いいえ

preferredMaintenanceWindow

週 1 回のシステムメンテナンスを実行できる時間帯 (世界標準時 (UTC))。

形式: ddd:hh24:mi-ddd:hh24:mi

デフォルト: それぞれ 8 時間の時間ブロックからランダムに選択された AWS リージョン 30 分のウィンドウ。ランダムな曜日に発生します。

有効な曜日: 月、火、水、木、金、土、日

制約: 最小限の 30 分単位のウィンドウ。

タイプ: 文字列

必須: いいえ

shardInstanceCount

Elastic クラスター内のすべてのシャードに適用されるレプリカインスタンスの数。shardInstanceCount 値 が 1 の場合、ライターインスタンスが 1 つあり、それ以外の追加のインスタンスはすべてレプリカであり、読み取りや可用性の向上に使用できます。

タイプ: 整数

必須: いいえ

subnetIds

新しい エラスティッククラスターの Amazon EC2 サブネット ID。

型: 文字列の配列

必須: いいえ

tags

新しいエラスティッククラスターに割り当てるタグ。

型: 文字列から文字列へのマップ

キーの長さ制限: 最小長さは 1 です。最大長は 128 です。

キーパターン: `(?!aws:)[a-zA-Z+-=._:/]+`

値の長さの制限: 最小長は 0 です。最大長は 256 です。

必須: いいえ

vpcSecurityGroupIds

このクラスターに関連付ける EC2 VPC セキュリティグループのリスト。

型: 文字列の配列

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
```

```
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
      "shardCapacity": number,
      "shardCount": number,
      "shardInstanceCount": number,
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
      ],
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

cluster

作成された新しい エラスティッククラスター。

型: Cluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

ConflictException

アクセス競合が発生しました。

HTTP ステータスコード: 409

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ServiceQuotaExceededException

アクションのサービスクォータを超えました。

HTTP ステータスコード: 402

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python

- AWS SDK for Ruby V3

CreateClusterSnapshot

サービス : Amazon DocumentDB Elastic Clusters

エラスティッククラスターのスナップショットを作成します。

リクエストの構文

```
POST /cluster-snapshot HTTP/1.1
Content-type: application/json

{
   "clusterArn": "string",
   "snapshotName": "string",
   "tags": {
      "string" : "string"
   }
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

clusterArn

スナップショットを作成するエラスティッククラスターの ARN 識別子。

タイプ: 文字列

必須: はい

snapshotName

新しい エラスティッククラススナップショットの名前。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 63 です。

必須: はい

tags

新しいエラスティッククラスターに割り当てるタグ。

型: 文字列から文字列へのマップ

キーの長さ制限: 最小長さは 1 です。最大長は 128 です。

キーパターン: (?!aws:)[a-zA-Z+-=._:/]+

値の長さの制限: 最小長は 0 です。最大長は 256 です。

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "snapshot": {
      "adminUserName": "string",
      "clusterArn": "string",
      "clusterCreationTime": "string",
      "kmsKeyId": "string",
      "snapshotArn": "string",
      "snapshotCreationTime": "string",
      "snapshotName": "string",
      "snapshotType": "string",
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

snapshot

新しいエラスティッククラススナップショットに関する情報を返します。

型: ClusterSnapshot オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

ConflictException

アクセス競合が発生しました。

HTTP ステータスコード: 409

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ServiceQuotaExceededException

アクションのサービスクォータを超えました。

HTTP ステータスコード: 402

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteCluster

サービス : Amazon DocumentDB Elastic Clusters

エラスティッククラスターを削除します。

リクエストの構文

```
DELETE /cluster/clusterArn HTTP/1.1
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

clusterArn

削除するエラスティッククラスターの ARN 識別子。

必須: はい

リクエストボディ

リクエストにリクエスト本文がありません。

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
      "shardCapacity": number,
      "shardCount": number,
```

```
      "shardInstanceCount": number,
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
      ],
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

cluster

新たに削除されたエラスティッククラススナップショットに関する情報を返します。

型: Cluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

ConflictException

アクセス競合が発生しました。

HTTP ステータスコード: 409

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteClusterSnapshot

サービス : Amazon DocumentDB Elastic Clusters

エラスティッククラスタースナップショットを削除します。

リクエストの構文

```
DELETE /cluster-snapshot/snapshotArn HTTP/1.1
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

snapshotArn

削除する エラスティッククラスタースナップショットの ARN 識別子。

必須: はい

リクエストボディ

リクエストにリクエスト本文がありません。

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "snapshot": {
      "adminUserName": "string",
      "clusterArn": "string",
      "clusterCreationTime": "string",
      "kmsKeyId": "string",
      "snapshotArn": "string",
      "snapshotCreationTime": "string",
      "snapshotName": "string",
      "snapshotType": "string",
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
```

```
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

snapshot

新たに削除されたエラスティッククラスタースナップショットに関する情報を返します。

型: ClusterSnapshot オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

ConflictException

アクセス競合が発生しました。

HTTP ステータスコード: 409

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetCluster

サービス : Amazon DocumentDB Elastic Clusters

特定のエラスティッククラスターに関する情報を返します。

リクエストの構文

```
GET /cluster/clusterArn HTTP/1.1
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

clusterArn

エラスティッククラスターの ARN 識別子。

必須: はい

リクエストボディ

リクエストにリクエスト本文がありません。

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
      "shardCapacity": number,
      "shardCount": number,
```

```
      "shardInstanceCount": number,
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
      ],
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

cluster

特定のエラスティッククラスターに関する情報を返します。

型: Cluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetClusterSnapshot

サービス : Amazon DocumentDB Elastic Clusters

特定のエラスティッククラスターナップショットに関する情報を返す

リクエストの構文

```
GET /cluster-snapshot/snapshotArn HTTP/1.1
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

snapshotArn

エラスティッククラススナップショットの ARN 識別子。

必須: はい

リクエストボディ

リクエストにリクエスト本文がありません。

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "snapshot": {
      "adminUserName": "string",
      "clusterArn": "string",
      "clusterCreationTime": "string",
      "kmsKeyId": "string",
      "snapshotArn": "string",
      "snapshotCreationTime": "string",
      "snapshotName": "string",
      "snapshotType": "string",
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
```

```
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

snapshot

特定のエラスティッククラスタースナップショットに関する情報を返します。

型: ClusterSnapshot オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetPendingMaintenanceAction

サービス : Amazon DocumentDB Elastic Clusters

保留中のすべてのメンテナンスアクションを取得します。

リクエストの構文

```
GET /pending-action/resourceArn HTTP/1.1
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

resourceArn

特定の Amazon リソースネーム (ARN) の保留中のメンテナンスアクションを取得します。

長さの制約: 最小長は 1 です。最大長は 256 です。

必須: はい

リクエストボディ

リクエストにリクエスト本文がありません。

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "resourcePendingMaintenanceAction": {
      "pendingMaintenanceActionDetails": [
         {
            "action": "string",
            "autoAppliedAfterDate": "string",
            "currentApplyDate": "string",
            "description": "string",
            "forcedApplyDate": "string",
            "optInStatus": "string"
         }
```

```
        ],
        "resourceArn": "string"
    }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

resourcePendingMaintenanceAction

リソースの保留中のメンテナンスアクションに関する情報を入力します。

型: ResourcePendingMaintenanceAction オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

ConflictException

アクセス競合が発生しました。

HTTP ステータスコード: 409

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListClusters

サービス : Amazon DocumentDB Elastic Clusters

プロビジョニングした Amazon DocumentDB クラスターに関する情報を返します。

リクエストの構文

```
GET /clusters?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

maxResults

レスポンスで受け取るエラスティッククラスタースナップショット結果の最大数。

有効範囲: 最小値 1 最大値は 100 です。

nextToken

以前のリクエストで提供されたページ分割トークン。このパラメータが指定されている場合、レスポンスにはこのトークンより大きく、max-results で指定された値までのレコードが含まれます。

レスポンスにデータがなくなると、nextToken は返されません。

リクエスト本文

リクエストにリクエスト本文がありません。

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "clusters": [
      {
         "clusterArn": "string",
         "clusterName": "string",
```

```
            "status": "string"
        }
    ],
    "nextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

clusters

Amazon DocumentDB エラスティッククラスターのリスト。

型: ClusterInList オブジェクトの配列

nextToken

以前のリクエストで提供されたページ分割トークン。このパラメータが指定されている場合、レスポンスにはこのトークンより大きく、max-results で指定された値までのレコードが含まれます。

レスポンスにデータがなくなると、nextToken は返されません。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListClusterSnapshots

サービス : Amazon DocumentDB Elastic Clusters

指定された Elastic クラスターのスナップショットに関する情報を返します。

リクエストの構文

```
GET /cluster-snapshots?
clusterArn=clusterArn&maxResults=maxResults&nextToken=nextToken&snapshotType=snapshotType
 HTTP/1.1
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

clusterArn

エラスティッククラスターの ARN 識別子。

maxResults

レスポンスで受け取るエラスティッククラスタースナップショット結果の最大数。

有効範囲: 最小値 は 20 です。最大値は 100 です。

nextToken

以前のリクエストで提供されたページ分割トークン。このパラメータが指定されている場合、レスポンスにはこのトークンより大きく、max-results で指定された値までのレコードが含まれます。

レスポンスにデータがなくなると、nextToken は返されません。

snapshotType

返されるクラスタースナップショットのタイプ。次のいずれかの値を指定できます。

- automated - Amazon DocumentDB が AWS アカウントに対して自動的に作成したすべてのクラスタースナップショットを返します。
- manual - AWS アカウント用に手動で作成したすべてのクラスタースナップショットを返します。

リクエスト本文

リクエストにリクエスト本文がありません。

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "snapshots": [
      {
         "clusterArn": "string",
         "snapshotArn": "string",
         "snapshotCreationTime": "string",
         "snapshotName": "string",
         "status": "string"
      }
   ]
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

nextToken

以前のリクエストで提供されたページ分割トークン。このパラメータが指定されている場合、レスポンスにはこのトークンより大きく、max-results で指定された値までのレコードが含まれます。

レスポンスにデータがなくなると、nextToken は返されません。

タイプ: 文字列

snapshots

指定した Elastic クラスターのスナップショットのリスト。

型: ClusterSnapshotInList オブジェクトの配列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListPendingMaintenanceActions

サービス : Amazon DocumentDB Elastic Clusters

保留中のすべてのメンテナンスアクションのリストを取得します。

リクエストの構文

```
GET /pending-actions?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

maxResults

レスポンスに含める結果の最大数。指定された maxResults の値よりも多くのレコードが存在する場合、ページ割りトークン (マーカー) がレスポンスに含まれるため、残りの結果を取得できます。

有効範囲: 最小値 1 最大値は 100 です。

nextToken

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには maxResults で指定された値まで、マーカーを超えるレコードのみが含まれます。

リクエスト本文

リクエストにリクエスト本文がありません。

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "resourcePendingMaintenanceActions": [
      {
         "pendingMaintenanceActionDetails": [
```

```
            {
               "action": "string",
               "autoAppliedAfterDate": "string",
               "currentApplyDate": "string",
               "description": "string",
               "forcedApplyDate": "string",
               "optInStatus": "string"
            }
         ],
         "resourceArn": "string"
      }
   ]
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

resourcePendingMaintenanceActions

リソースの保留中のメンテナンスアクションに関する情報を入力します。

型: ResourcePendingMaintenanceAction オブジェクトの配列

nextToken

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータが表示される場合、レスポンスには、 で指定された値まで、マーカー以外のレコードのみが含まれますmaxResults。

タイプ: 文字列

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListTagsForResource

サービス : Amazon DocumentDB Elastic Clusters

エラスティッククラスターリソースのすべてのタグを一覧表示します。

リクエストの構文

```
GET /tags/resourceArn HTTP/1.1
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

resourceArn

エラスティッククラスターリソースの ARN 識別子。

長さの制限：最小長 1、最大長は 1011 です。

必須: はい

リクエストボディ

リクエストにリクエスト本文がありません。

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "tags": {
      "string" : "string"
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

tags

指定されたエラスティッククラスターリソースに対するタグのリスト。

型: 文字列から文字列へのマップ

キーの長さ制限: 最小長さは 1 です。最大長は 128 です。

キーパターン: `(?!aws:)[a-zA-Z+-=._:/]+`

値の長さの制限: 最小長は 0 です。最大長は 256 です。

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

RestoreClusterFromSnapshot

サービス : Amazon DocumentDB Elastic Clusters

スナップショットからエラスティッククラスターの復元します。

リクエストの構文

```
POST /cluster-snapshot/snapshotArn/restore HTTP/1.1
Content-type: application/json

{
   "clusterName": "string",
   "kmsKeyId": "string",
   "shardCapacity": number,
   "shardInstanceCount": number,
   "subnetIds": [ "string" ],
   "tags": {
      "string" : "string"
   },
   "vpcSecurityGroupIds": [ "string" ]
}
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

snapshotArn

エラスティッククラスタースナップショットの ARN 識別子。

必須: はい

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

clusterName

エラスティッククラスターの名前。

タイプ: 文字列

必須: はい

kmsKeyId

新しい Amazon DocumentDB エラスティッククラスターの暗号化に使用する KMS キー識別子。

KMS キー識別子は、KMS 暗号化キーの Amazon リソースネーム (ARN) です。KMS 暗号化キーを所有する Amazon アカウントと同じアカウントを使用して DB クラスターを作成する場合、KMS 暗号化キーの ARN の代わりに KMS キーのエイリアスを使用できます。

ここで暗号化キーが指定されていない場合、Amazon DocumentDB は KMS がアカウント用に作成したデフォルトの暗号化キーを使用します。各アカウントは Amazon のリージョンごとにデフォルトの暗号化キーを持ちます。

タイプ: 文字列

必須: いいえ

shardCapacity

新しく復元された Elastic クラスター内の各シャードの容量。

タイプ: 整数

必須: いいえ

shardInstanceCount

Elastic クラスター内のすべてのシャードに適用されるレプリカインスタンスの数。`shardInstanceCount` 値 が 1 の場合、ライターインスタンスが 1 つあり、それ以外の追加のインスタンスはすべてレプリカであり、読み取りや可用性の向上に使用できます。

タイプ: 整数

必須: いいえ

subnetIds

エラスティッククラスターの Amazon EC2 サブネット ID。

型: 文字列の配列

必須: いいえ

tags

復元されたエラスティッククラスターに割り当てるキーと値のペアの配列形式でのタグ名のリスト。キーがタグ名、値がキー値です。

型: 文字列から文字列へのマップ

キーの長さ制限: 最小長さは 1 です。最大長は 128 です。

キーパターン: (?!aws:)[a-zA-Z+-=._:/]+

値の長さの制限: 最小長は 0 です。最大長は 256 です。

必須: いいえ

vpcSecurityGroupIds

このエラスティッククラスターに関連付ける EC2 VPC セキュリティグループのリスト。

型: 文字列の配列

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
      "shardCapacity": number,
      "shardCount": number,
      "shardInstanceCount": number,
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
```

```
        ],
        "status": "string",
        "subnetIds": [ "string" ],
        "vpcSecurityGroupIds": [ "string" ]
    }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

cluster

復元されたエラスティッククラスターに関する情報を返します。

型: Cluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

ConflictException

アクセス競合が発生しました。

HTTP ステータスコード: 409

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ServiceQuotaExceededException

アクションのサービスクォータを超えました。

HTTP ステータスコード: 402

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

StartCluster

サービス : Amazon DocumentDB Elastic Clusters

clusterARN で指定された停止中の Elastic クラスターを再開します。

リクエストの構文

```
POST /cluster/clusterArn/start HTTP/1.1
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

clusterArn

エラスティッククラスターの ARN 識別子。

必須: はい

リクエストボディ

リクエストにリクエスト本文がありません。

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
      "shardCapacity": number,
      "shardCount": number,
```

```
      "shardInstanceCount": number,
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
      ],
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

cluster

特定のエラスティッククラスターに関する情報を返します。

型: Cluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

StopCluster

サービス : Amazon DocumentDB Elastic Clusters

clusterArn で指定された実行中の Elastic クラスターを停止します。これには、この Elastic クラスターが 利用可能な 状態である必要があります。

リクエストの構文

```
POST /cluster/clusterArn/stop HTTP/1.1
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

clusterArn

エラスティッククラスターの ARN 識別子。

必須: はい

リクエストボディ

リクエストにリクエスト本文がありません。

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
```

```
      "shardCapacity": number,
      "shardCount": number,
      "shardInstanceCount": number,
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
      ],
      "status": "string",
      "subnetIds": [ "string" ],
      "vpcSecurityGroupIds": [ "string" ]
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

cluster

特定のエラスティッククラスターに関する情報を返します。

型: Cluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

TagResource

サービス : Amazon DocumentDB Elastic Clusters

エラスティッククラスターリソースにメタデータタグを追加します

リクエストの構文

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "tags": {
      "string" : "string"
   }
}
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

resourceArn

エラスティッククラスターリソースの ARN 識別子。

長さの制限 : 最小長 1、最大長は 1011 です。

必須: はい

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

tags

エラスティッククラスターリソースに割り当てられているタグ。

型: 文字列から文字列へのマップ

キーの長さ制限: 最小長さは 1 です。最大長は 128 です。

キーパターン: `(?!aws:)[a-zA-Z+-=._:/]+`

値の長さの制限: 最小長は 0 です。最大長は 256 です。

必須: はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UntagResource

サービス : Amazon DocumentDB Elastic Clusters

エラスティッククラスターリソースからメタデータタグを削除します。

リクエストの構文

```
DELETE /tags/resourceArn?tagKeys=tagKeys HTTP/1.1
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

resourceArn

エラスティッククラスターリソースの ARN 識別子。

長さの制限：最小長 1、最大長は 1011 です。

必須: はい

tagKeys

エラスティッククラスターリソースから削除するタグのキー。

配列メンバー: 最小数は 0 項目です。最大数は 50 項目です。

長さの制限：最小長 1、最大長は 128 です。

パターン: (?!aws:)[a-zA-Z+-=._:/]+

必須: はい

リクエストボディ

リクエストにリクエスト本文がありません。

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3

- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateCluster

サービス : Amazon DocumentDB Elastic Clusters

エラスティッククラスターを変更します。これには、管理者ユーザー名/パスワードの更新、API バージョンのアップグレード、バックアップウィンドウとメンテナンスウィンドウの設定が含まれます

リクエストの構文

```
PUT /cluster/clusterArn HTTP/1.1
Content-type: application/json

{
   "adminUserPassword": "string",
   "authType": "string",
   "backupRetentionPeriod": number,
   "clientToken": "string",
   "preferredBackupWindow": "string",
   "preferredMaintenanceWindow": "string",
   "shardCapacity": number,
   "shardCount": number,
   "shardInstanceCount": number,
   "subnetIds": [ "string" ],
   "vpcSecurityGroupIds": [ "string" ]
}
```

URI リクエストパラメータ

リクエストでは、次の URI パラメータを使用します。

clusterArn

エラスティッククラスターの ARN 識別子。

必須: はい

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

adminUserPassword

エラスティッククラスター管理者に関連付けられているパスワード。このパスワードをには、スラッシュ (/)、二重引用符 (")、または「at」記号 (@) 以外の印刷可能な ASCII 文字を含むことができます。

制約: 8 ~ 100 文字の長さの英数字である必要があります。

タイプ: 文字列

必須: いいえ

authType

エラスティッククラスターへのアクセスに使用するパスワードの取得先の特定に使用している認証タイプ。有効なタイプは、PLAIN_TEXT または SECRET_ARN です。

型: 文字列

有効な値：PLAIN_TEXT | SECRET_ARN

必須：いいえ

backupRetentionPeriod

自動スナップショットが保持される日数。

タイプ: 整数

必須: いいえ

clientToken

エラスティッククラスターのクライアントトークン。

タイプ: 文字列

必須: いいえ

preferredBackupWindow

backupRetentionPeriod に応じた、自動バックアップが有効な場合に毎日自動バックアップが作成される時間範囲を指定します。

タイプ: 文字列

必須: いいえ

preferredMaintenanceWindow

週 1 回のシステムメンテナンスを実行できる時間帯 (世界標準時 (UTC))。

形式: `ddd:hh24:mi-ddd:hh24:mi`

デフォルト: それぞれ 8 時間の時間ブロックからランダムに選択された AWS リージョン 30 分のウィンドウ。ランダムな曜日に発生します。

有効な曜日: 月、火、水、木、金、土、日

制約: 最小限の 30 分単位のウィンドウ。

タイプ: 文字列

必須: いいえ

shardCapacity

各エラスティッククラスターシャードに割り当てられている vCPU の数。最大数は 64 です。使用できる値は、2、4、8、16、32、64 です。

タイプ: 整数

必須: いいえ

shardCount

エラスティッククラスターに割り当てられているシャードの数。最大数は 32 です。

タイプ: 整数

必須: いいえ

shardInstanceCount

Elastic クラスター内のすべてのシャードに適用されるレプリカインスタンスの数。`shardInstanceCount` 値 が 1 の場合、ライターインスタンスが 1 つあり、それ以外の追加のインスタンスはすべてレプリカであり、読み取りや可用性の向上に使用できます。

タイプ: 整数

必須: いいえ

subnetIds

エラスティッククラスターの Amazon EC2 サブネット ID。

型: 文字列の配列

必須: いいえ

vpcSecurityGroupIds

このエラスティッククラスターに関連付ける EC2 VPC セキュリティグループのリスト。

型: 文字列の配列

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "cluster": {
      "adminUserName": "string",
      "authType": "string",
      "backupRetentionPeriod": number,
      "clusterArn": "string",
      "clusterEndpoint": "string",
      "clusterName": "string",
      "createTime": "string",
      "kmsKeyId": "string",
      "preferredBackupWindow": "string",
      "preferredMaintenanceWindow": "string",
      "shardCapacity": number,
      "shardCount": number,
      "shardInstanceCount": number,
      "shards": [
         {
            "createTime": "string",
            "shardId": "string",
            "status": "string"
         }
      ],
      "status": "string",
```

```
        "subnetIds": [ "string" ],
        "vpcSecurityGroupIds": [ "string" ]
    }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

cluster

更新されたエラスティッククラスターに関する情報を返します。

型: Cluster オブジェクト

エラー

すべてのアクションに共通のエラーについては、「共通エラー」を参照してください。

AccessDeniedException

アクションを実行するための十分なアクセス許可がない場合に発生する例外。

HTTP ステータスコード: 403

ConflictException

アクセス競合が発生しました。

HTTP ステータスコード: 409

InternalServerException

内部サーバーエラーが発生しました。

HTTP ステータスコード: 500

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

ThrottlingException

ThrottlingException は、リクエストのスロットリングによりリクエストが拒否された場合にスローされます。

HTTP ステータスコード: 429

ValidationException

検証例外の定義がある構造。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

データ型

次のデータ型は でサポートされています Amazon DocumentDB (with MongoDB compatibility)。

- AvailabilityZone
- Certificate
- CertificateDetails
- CloudwatchLogsExportConfiguration

- ClusterMasterUserSecret
- DBCluster
- DBClusterMember
- DBClusterParameterGroup
- DBClusterRole
- DBClusterSnapshot
- DBClusterSnapshotAttribute
- DBClusterSnapshotAttributesResult
- DBEngineVersion
- DBInstance
- DBInstanceStatusInfo
- DBSubnetGroup
- Endpoint
- EngineDefaults
- Event
- EventCategoriesMap
- EventSubscription
- Filter
- GlobalCluster
- GlobalClusterMember
- OrderableDBInstanceOption
- Parameter
- PendingCloudwatchLogsExports
- PendingMaintenanceAction
- PendingModifiedValues
- ResourcePendingMaintenanceActions
- ServerlessV2FeaturesSupport
- ServerlessV2ScalingConfiguration
- ServerlessV2ScalingConfigurationInfo
- Subnet

- Tag
- UpgradeTarget
- VpcSecurityGroupMembership

Amazon DocumentDB Elastic クラスターでは、以下のデータ型がサポートされています。

- Cluster
- ClusterInList
- ClusterSnapshot
- ClusterSnapshotInList
- PendingMaintenanceActionDetails
- ResourcePendingMaintenanceAction
- Shard
- ValidationExceptionField

Amazon DocumentDB (with MongoDB compatibility)

次のデータ型は でサポートされています Amazon DocumentDB (with MongoDB compatibility)。

- AvailabilityZone
- Certificate
- CertificateDetails
- CloudwatchLogsExportConfiguration
- ClusterMasterUserSecret
- DBCluster
- DBClusterMember
- DBClusterParameterGroup
- DBClusterRole
- DBClusterSnapshot
- DBClusterSnapshotAttribute
- DBClusterSnapshotAttributesResult
- DBEngineVersion

- DBInstance
- DBInstanceStatusInfo
- DBSubnetGroup
- Endpoint
- EngineDefaults
- Event
- EventCategoriesMap
- EventSubscription
- Filter
- GlobalCluster
- GlobalClusterMember
- OrderableDBInstanceOption
- Parameter
- PendingCloudwatchLogsExports
- PendingMaintenanceAction
- PendingModifiedValues
- ResourcePendingMaintenanceActions
- ServerlessV2FeaturesSupport
- ServerlessV2ScalingConfiguration
- ServerlessV2ScalingConfigurationInfo
- Subnet
- Tag
- UpgradeTarget
- VpcSecurityGroupMembership

AvailabilityZone

サービス: Amazon DocumentDB (with MongoDB compatibility)

アベイラビリティーゾーンに関する情報。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

Name

アベイラビリティーゾーンの名前。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Certificate

サービス: Amazon DocumentDB (with MongoDB compatibility)

の認証局 (CA) 証明書 AWS アカウント。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

CertificateArn

証明書の Amazon リソースネーム (ARN)。

例: `arn:aws:rds:us-east-1::cert:rds-ca-2019`

タイプ: 文字列

必須: いいえ

CertificateIdentifier

証明書を識別する一意のキー。

例: `rds-ca-2019`

タイプ: 文字列

必須: いいえ

CertificateType

証明書の種類を入力します。

例: `CA`

タイプ: 文字列

必須: いいえ

Thumbprint

証明書のサムプリント。

タイプ: 文字列

必須: いいえ

ValidFrom

証明書が有効である開始日時。

例: 2019-07-31T17:57:09Z

型: タイムスタンプ

必須: いいえ

ValidTill

証明書が有効でなくなる日時。

例: 2024-07-31T17:57:09Z

型: タイムスタンプ

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CertificateDetails

サービス: Amazon DocumentDB (with MongoDB compatibility)

DB インスタンスのサーバー証明書の詳細を返します。

詳細については、「Amazon DocumentDB デベロッパーガイド」の「Amazon DocumentDB TLS 証明書の更新」と「転送中のデータの暗号化」を参照してください。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

CAIdentifier

DB インスタンスのサーバー証明書に使用される CA 証明書の CA 識別子。

タイプ: 文字列

必須: いいえ

ValidTill

DB インスタンスのサーバー証明書の有効期限。

型: タイムスタンプ

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CloudwatchLogsExportConfiguration

サービス: Amazon DocumentDB (with MongoDB compatibility)

特定のインスタンスまたはクラスターについて、Amazon CloudWatch Logs へのエクスポートを有効にするためのログタイプの設定。

`EnableLogTypes` および `DisableLogTypes` の配列では、CloudWatch Logs にエクスポートする (または、エクスポートしない) ログを決定します。これらの配列内の値は、使用されているエンジンによって異なります。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

DisableLogTypes.member.N

無効にするログタイプのリスト。

タイプ: 文字列の配列

必須: いいえ

EnableLogTypes.member.N

有効にするログタイプのリスト。

タイプ: 文字列の配列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ClusterMasterUserSecret

サービス: Amazon DocumentDB (with MongoDB compatibility)

マスターユーザーパスワードの Secrets Manager で Amazon DocumentDB によって管理される AWS シークレットが含まれます。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

KmsKeyId

シークレットの暗号化に使用される AWS KMS キー識別子。

タイプ: 文字列

必須: いいえ

SecretArn

シークレットの Amazon リソースネーム (ARN)。

タイプ: 文字列

必須: いいえ

SecretStatus

シークレットのステータス。

設定可能なステータス値は以下のとおりです。

- creating - シークレットを作成中です。
- active - シークレットは通常の使用とローテーションに使用できます。
- ローテーション - シークレットはローテーション中です。
- 障害 - シークレットはデータベース認証情報へのアクセスに使用できますが、ローテーションすることはできません。例えば、Amazon DocumentDB がシークレットまたはシークレットの KMS キーにアクセスできなくなるようにアクセス許可が変更された場合、シークレットはこのステータスになる可能性があります。

シークレットのステータスがこの場合は、ステータスの原因となった状態を修正できます。または、データベース認証情報の自動管理を無効にするようにインスタンスを変更し、データベース認証情報の自動管理を有効にするようにインスタンスを再度変更します。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBCluster

サービス: Amazon DocumentDB (with MongoDB compatibility)

クラスターに関する詳細情報。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

AssociatedRoles.DBClusterRole.N

クラスターに関連付けられている AWS Identity and Access Management (IAM) ロールのリストを提供します。クラスターに関連付けられている (IAM) ロールは、クラスターがユーザーに代わって他の AWS サービスにアクセスするためのアクセス許可を付与します。

タイプ: DBClusterRole オブジェクトの配列

必須: いいえ

AvailabilityZones.AvailabilityZone.N

クラスターのインスタンスを作成できる Amazon EC2 アベイラビリティーゾーン (AZ) のリストを入力します。

型: 文字列の配列

必須: いいえ

BackupRetentionPeriod

自動スナップショットが保持される日数を指定します。

タイプ: 整数

必須: いいえ

CloneGroupId

DB クラスターが関連付けられているクローングループを識別します。

タイプ: 文字列

必須: いいえ

ClusterCreateTime

クラスターが作成された時刻を協定世界時 (UTC) で指定します。

型: タイムスタンプ

必須: いいえ

DBClusterArn

クラスターの Amazon リソースネーム (ARN) を返します。

タイプ: 文字列

必須: いいえ

DBClusterIdentifier

ユーザーが指定したクラスター識別子が含まれています。この識別子は、クラスターを識別する一意のキーです。

タイプ: 文字列

必須: いいえ

DBClusterMembers.DBClusterMember.N

クラスターを構成するインスタンスのリストを入力します。

タイプ: DBClusterMember オブジェクトの配列

必須: いいえ

DBClusterParameterGroup

クラスターに使用するクラスターパラメータグループの名前を指定します。

タイプ: 文字列

必須: いいえ

DbClusterResourceId

クラスターの AWS リージョン一意でイミュータブルな識別子。この識別子は、クラスターの AWS KMS キーにアクセスするたびに AWS CloudTrail ログエントリに記録されます。

タイプ: 文字列

必須: いいえ

DBSubnetGroup

サブネットグループ内の名前、説明、サブネットなど、クラスターに関連付けられているサブネットグループに関する情報を指定します。

タイプ: 文字列

必須: いいえ

DeletionProtection

このクラスターを削除できるかどうかを指定します。DeletionProtection を有効にすると、クラスターが変更され DeletionProtection が無効にならない限り、クラスターを削除することができません。DeletionProtection はクラスターが誤って削除されるのを防ぎます。

型: ブール値

必須: いいえ

EarliestRestorableTime

特定時点への復元によりデータベースを復元できる最も早い時刻。

型: タイムスタンプ

必須: いいえ

EnabledCloudwatchLogsExports.member.N

このクラスターが Amazon CloudWatch Logs にエクスポートするように設定されている口グタイプのリスト。

型: 文字列の配列

必須: いいえ

Endpoint

クラスターのプライマリインスタンスの接続エンドポイントを指定します。

タイプ: 文字列

必須: いいえ

Engine

このクラスターに使用されるデータベースエンジンの名前を入力します。

タイプ: 文字列

必須: いいえ

EngineVersion

データベースエンジンのバージョンを示します。

タイプ: 文字列

必須: いいえ

HostedZoneId

ホストゾーンを作成するときに Amazon Route 53 が割り当てる ID を指定します。

タイプ: 文字列

必須: いいえ

IOOptimizedNextAllowedModificationTime

次回は、Amazon DocumentDB クラスターを変更して iopt1 ストレージタイプを使用できます。

型: タイムスタンプ

必須: いいえ

KmsKeyId

`StorageEncrypted` が の場合`true`、暗号化されたクラスターの AWS KMS キー識別子。

タイプ: 文字列

必須: いいえ

LatestRestorableTime

特定時点への復元によりデータベースを復元できる最新の日時を指定します。

型: タイムスタンプ

必須: いいえ

MasterUsername

クラスターのマスターユーザー名が含まれています。

タイプ: 文字列

必須: いいえ

MasterUserSecret

マスターユーザーパスワードの AWS Secrets Manager で Amazon DocumentDB によって管理されるシークレット。

型: ClusterMasterUserSecret オブジェクト

必須: いいえ

MultiAZ

クラスターが複数のアベイラビリティーゾーンにインスタンスを持つかどうかを指定します。

型: ブール値

必須: いいえ

PercentProgress

オペレーションの進行状況をパーセンテージで指定します。

タイプ: 文字列

必須: いいえ

Port

データベースエンジンがリッスンするポートを指定します。

タイプ: 整数

必須: いいえ

PreferredBackupWindow

BackupRetentionPeriod に応じた、自動バックアップが有効な場合に自動バックアップが作成される毎日の時間範囲を指定します。

タイプ: 文字列

必須: いいえ

PreferredMaintenanceWindow

週 1 回のシステムメンテナンスを実行できる時間範囲を世界標準時 (UTC) で指定します。

タイプ: 文字列

必須: いいえ

ReaderEndpoint

クラスターの読み込みエンドポイント。クラスター用のリーダーエンドポイントは、クラスタ内の使用可能な Amazon DocumentDB レプリカ間で接続の負荷分散を行います。クライアントがリーダーエンドポイントへの新規接続をリクエストすると、Amazon DocumentDB によって接続リクエストがクラスターの Amazon DocumentDB レプリカ間で配分されます。この機能は、クラスターの複数の Amazon DocumentDB レプリカ間の読み取りワークロードを分散させる役に立ちます。

フェイルオーバーが発生し、接続している Amazon DocumentDB レプリカがプライマリインスタンスに昇格すると、接続は削除されます。読み取りワークロードをクラスター内の他の Amazon DocumentDB レプリカに送信し続けるために、リーダーエンドポイントに再接続することができます。

タイプ: 文字列

必須: いいえ

ReadReplicaIdentifiers.ReadReplicaIdentifier.N

このクラスターに関連付けられているセカンダリクラスターの 1 つ以上の識別子を含みます。

型: 文字列の配列

必須: いいえ

ReplicationSourceIdentifier

このクラスターがセカンダリークラスターの場合は、ソースクラスターの識別子が含まれます。

タイプ: 文字列

必須: いいえ

ServerlessV2ScalingConfiguration

Amazon DocumentDB Serverless クラスターのスケーリング設定。

型: ServerlessV2ScalingConfigurationInfo オブジェクト

必須: いいえ

Status

このクラスターの現在の状態を指定します。

タイプ: 文字列

必須: いいえ

StorageEncrypted

クラスターが暗号化されているかどうかを指定します。

型: ブール値

必須: いいえ

StorageType

クラスターに関連付けられたストレージタイプ

Amazon DocumentDB クラスターのストレージタイプの詳細については、「Amazon DocumentDB 開発者ガイド」の「クラスターストレージ設定」を参照してください。

ストレージタイプの有効な値 - `standard | iopt1`

デフォルト値は `standard` です

タイプ: 文字列

必須: いいえ

VpcSecurityGroups.VpcSecurityGroupMembership.N

クラスターに関連付けられる VPC (Virtual Private Cloud) セキュリティグループのリスト。

タイプ: VpcSecurityGroupMembership オブジェクトの配列

必須：いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBClusterMember

サービス: Amazon DocumentDB (with MongoDB compatibility)

クラスターの一部であるインスタンスに関する情報が含まれています。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

DBClusterParameterGroupStatus

クラスターのこのメンバーの DB クラスターパラメータグループのステータスを指定します。

タイプ: 文字列

必須: いいえ

DBInstanceIdentifier

クラスターのこのメンバーのインスタンス ID を指定します。

タイプ: 文字列

必須: いいえ

IsClusterWriter

クラスターメンバーがクラスターのプライマリインスタンスの場合は `true`、それ以外の場合は `false` です。

型: ブール値

必須: いいえ

PromotionTier

既存のプライマリインスタンスの障害後に、Amazon DocumentDB レプリカをプライマリインスタンスに昇格される順序を指定する値。

タイプ: 整数

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBClusterParameterGroup

サービス: Amazon DocumentDB (with MongoDB compatibility)

クラスターパラメータグループに関する詳細情報。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

DBClusterParameterGroupArn

クラスターパラメータグループの Amazon リソースネーム (ARN)。

タイプ: 文字列

必須: いいえ

DBClusterParameterGroupName

クラスターパラメータグループの名前を指定します。

タイプ: 文字列

必須: いいえ

DBParameterGroupFamily

このクラスターパラメータグループと互換性があるクラスターパラメータグループファミリーの名前を指定します。

タイプ: 文字列

必須: いいえ

Description

このクラスターパラメータグループに対するユーザー定義の説明を指定します。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBClusterRole

サービス: Amazon DocumentDB (with MongoDB compatibility)

クラスターに関連付けられている AWS Identity and Access Management (IAM) ロールについて説明します。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

RoleArn

DB クラスターに関連付けられる IAM ロール Amazon リソースネーム (ARN)。

タイプ: 文字列

必須: いいえ

Status

IAM ロールとクラスターの間の関連付けの状態を説明します。`Status` プロパティは以下のいずれかの値を返します。

- `ACTIVE` - IAMrole ARN はクラスターに関連付けられており、ユーザーに代わって他の AWS サービスにアクセスするために使用できます。
- `PENDING` - IAM ロール ARN はクラスターに関連付けられています。
- `INVALID` - IAMrole ARN はクラスターに関連付けられていますが、クラスターはユーザーに代わって他の AWS のサービスにアクセスする IAMroleを引き受けることはできません。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBClusterSnapshot

サービス: Amazon DocumentDB (with MongoDB compatibility)

クラススナップショットに関する詳細情報。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

AvailabilityZones.AvailabilityZone.N

クラススナップショットのインスタンスを復元できる Amazon EC2 アベイラビリティーゾーン (AZ) のリストを入力します。

タイプ: 文字列の配列

必須: いいえ

ClusterCreateTime

クラスターが作成された時刻を協定世界時 (UTC) で指定します。

型: タイムスタンプ

必須: いいえ

DBClusterIdentifier

このクラススナップショットの作成元のクラスターのクラスター識別子を指定します。

タイプ: 文字列

必須: いいえ

DBClusterSnapshotArn

クラススナップショットの Amazon リソースネーム (ARN)。

タイプ: 文字列

必須: いいえ

DBClusterSnapshotIdentifier

クラススナップショットの識別子を指定します。

タイプ: 文字列

必須: いいえ

Engine

データベースエンジンの名前を指定します。

タイプ: 文字列

必須: いいえ

EngineVersion

このクラススナップショットに使用されるデータベースエンジンのバージョンを入力します。

タイプ: 文字列

必須: いいえ

KmsKeyId

StorageEncrypted が の場合true、暗号化されたクラススナップショットの AWS KMS キー識別子。

タイプ: 文字列

必須: いいえ

MasterUsername

クラススナップショットのマスターユーザー名を入力します。

タイプ: 文字列

必須: いいえ

PercentProgress

転送された推定データの割合を指定します。

タイプ: 整数

必須: いいえ

Port

スナップショット時にクラスターが待機していたポートを指定します。

タイプ: 整数

必須: いいえ

SnapshotCreateTime

スナップショットが取られた時刻を協定世界時 (UTC) で入力します。

型: タイムスタンプ

必須: いいえ

SnapshotType

クラスタースナップショットのタイプを入力します。

タイプ: 文字列

必須: いいえ

SourceDBClusterSnapshotArn

クラスタースナップショットがソースクラスタースナップショットからコピーされた場合は、ソースクラスタースナップショットの ARŃ、それ以外の場合は null 値。

タイプ: 文字列

必須: いいえ

Status

このクラスタースナップショットのステータスを指定します。

タイプ: 文字列

必須: いいえ

StorageEncrypted

クラスタースナップショットが暗号化されているかどうかを指定します。

型: ブール値

必須: いいえ

StorageType

クラスタースナップショットに関連付けられているストレージタイプ。

Amazon DocumentDB クラスターのストレージタイプの詳細については、「Amazon DocumentDB 開発者ガイド」の「クラスターストレージ設定」を参照してください。

ストレージタイプの有効な値 - `standard | iopt1`

デフォルト値は `standard` です

タイプ: 文字列

必須: いいえ

VpcId

クラスタースナップショットに関連付けられている仮想プライベートクラウド (VPC) ID を提供します。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBClusterSnapshotAttribute

サービス: Amazon DocumentDB (with MongoDB compatibility)

手動クラススナップショット属性の名前と値を含みます。

手動クラススナップショット属性は、手動クラススナップショットの復元 AWS アカウントを他の に許可するために使用されます。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

AttributeName

手動クラススナップショット属性の名前。

という名前の属性は、手動クラススナップショットをコピーまたは復元するアクセス許可 AWS アカウント を持つ のリストrestoreを参照します。

タイプ: 文字列

必須: いいえ

AttributeValues.AttributeValue.N

手動クラススナップショット属性の値。

AttributeName フィールドが に設定されている場合restore、この要素は手動クラススナップショットのコピーまたは復元が AWS アカウント 許可されている の IDs のリストを返します。の値がリストallにある場合、手動クラススナップショットはパブリックであり、 が AWS アカウント コピーまたは復元できます。

タイプ: 文字列の配列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBClusterSnapshotAttributesResult

サービス: Amazon DocumentDB (with MongoDB compatibility)

クラススナップショットに関連付けられている属性に関する詳細情報。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

DBClusterSnapshotAttributes.DBClusterSnapshotAttribute.N

クラススナップショットの属性と値のリスト。

型: DBClusterSnapshotAttribute オブジェクトの配列

必須: いいえ

DBClusterSnapshotIdentifier

属性が適用されるクラススナップショットの識別子。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBEngineVersion

サービス: Amazon DocumentDB (with MongoDB compatibility)

エンジンのバージョンに関する詳細情報。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

DBEngineDescription

データベースエンジンの説明。

タイプ: 文字列

必須: いいえ

DBEngineVersionDescription

データベースエンジンのバージョンの説明。

タイプ: 文字列

必須: いいえ

DBParameterGroupFamily

データベースエンジンのパラメータグループファミリーの名前。

タイプ: 文字列

必須: いいえ

Engine

データベースエンジンの名前。

タイプ: 文字列

必須: いいえ

EngineVersion

データベースエンジンのバージョン番号。

タイプ: 文字列

必須: いいえ

ExportableLogTypes.member.N

データベースエンジンが Amazon CloudWatch Logs にエクスポートできるログのタイプ。

型: 文字列の配列

必須: いいえ

ServerlessV2FeaturesSupport

Amazon DocumentDB エンジンのバージョンが異なる Amazon DocumentDB Serverless プロパティまたは制限を指定します。新しいクラスターまたはアップグレードされたクラスターで使用する Amazon DocumentDB バージョンを決定するときに、この属性の値をテストできます。また、既存のクラスターのバージョンを取得し、それらの機能を使用する前に、そのバージョンが特定の Amazon DocumentDB Serverless 機能をサポートしているかどうかを確認することもできます。

型: ServerlessV2FeaturesSupport オブジェクト

必須: いいえ

SupportedCACertificateIdentifiers.member.N

サポートされている CA 証明書識別子のリスト。

詳細については、「Amazon DocumentDB デベロッパーガイド」の「Amazon DocumentDB TLS 証明書の更新」と「転送中のデータの暗号化」を参照してください。

型: 文字列の配列

必須: いいえ

SupportsCertificateRotationWithoutRestart

エンジンが DB インスタンスを再起動せずにサーバー証明書をローテーションすることをサポートしているバージョンかどうかを示します。

型: ブール値

必須: いいえ

SupportsLogExportsToCloudwatchLogs

エンジンのバージョンが `ExportableLogTypes` で指定されたログタイプの CloudWatch Logs へのエクスポートをサポートするかどうかを示す値。

型: ブール値

必須: いいえ

ValidUpgradeTarget.UpgradeTarget.N

このデータベースエンジンのバージョンをアップグレードできるエンジンのバージョンのリスト。

タイプ: UpgradeTarget オブジェクトの配列

必須：いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBInstance

サービス: Amazon DocumentDB (with MongoDB compatibility)

インスタンスに関する詳細情報。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

AutoMinorVersionUpgrade

適用されません。このパラメータは Amazon DocumentDB には適用されません。Amazon DocumentDB は、設定値に関係なく、マイナーバージョンのアップグレードを実行しません。

型: ブール値

必須: いいえ

AvailabilityZone

インスタンスを配置するアベイラビリティーゾーンの名前を指定します。

タイプ: 文字列

必須: いいえ

BackupRetentionPeriod

自動スナップショットが保持される日数を指定します。

タイプ: 整数

必須: いいえ

CACertificateIdentifier

この DB インスタンスの CA 証明書の識別子。

タイプ: 文字列

必須: いいえ

CertificateDetails

DB インスタンスのサーバー証明書の詳細。

型: CertificateDetails オブジェクト

必須: いいえ

CopyTagsToSnapshot

タグを DB インスタンスから DB インスタンスのスナップショットにコピーするかどうかを示す値。デフォルトでは、タグはコピーされません。

型: ブール値

必須: いいえ

DBClusterIdentifier

インスタンスがクラスターのメンバーである場合は、インスタンスがメンバーとなっているクラスターの名前が含まれます。

タイプ: 文字列

必須: いいえ

DBInstanceArn

インスタンスの Amazon リソースネーム (ARN)。

タイプ: 文字列

必須: いいえ

DBInstanceClass

インスタンスのコンピューティングおよびメモリ容量クラスの名前を指定します。

タイプ: 文字列

必須: いいえ

DBInstanceIdentifier

ユーザーが指定したデータベース識別子が含まれています。この識別子は、インスタンスを識別する一意のキーです。

タイプ: 文字列

必須: いいえ

DBInstanceStatus

このデータベースの現在の状態を指定します。

タイプ: 文字列

必須: いいえ

DbiResourceId

インスタンスの AWS リージョン一意でイミュータブルな識別子。この識別子は、インスタンスの AWS KMS キーにアクセスするたびに AWS CloudTrail ログエントリに記録されます。

タイプ: 文字列

必須: いいえ

DBSubnetGroup

サブネットグループ内の名前、説明、サブネットなど、インスタンスに関連付けられているサブネットグループに関する情報を指定します。

型: DBSubnetGroup オブジェクト

必須: いいえ

EnabledCloudwatchLogsExports.member.N

このインスタンスが CloudWatch Logs にエクスポートするように設定されている ログタイプのリスト。

タイプ: 文字列の配列

必須: いいえ

Endpoint

接続エンドポイントを指定します。

型: Endpoint オブジェクト

必須: いいえ

Engine

このインスタンスに使用されるデータベースエンジンの名前を入力します。

タイプ: 文字列

必須: いいえ

EngineVersion

データベースエンジンのバージョンを示します。

タイプ: 文字列

必須: いいえ

InstanceCreateTime

インスタンスが作成された日時を入力します。

型: タイムスタンプ

必須: いいえ

KmsKeyId

`StorageEncrypted` が の場合`true`、暗号化されたインスタンスの AWS KMS キー識別子。

タイプ: 文字列

必須: いいえ

LatestRestorableTime

特定時点への復元によりデータベースを復元できる最新の日時を指定します。

型: タイムスタンプ

必須: いいえ

PendingModifiedValues

インスタンスへの変更が保留中であることを指定します。この要素は、変更が保留中の場合にのみ含まれます。特定の変更は、サブエレメントによって識別されます。

型: PendingModifiedValues オブジェクト

必須: いいえ

PerformanceInsightsEnabled

Amazon RDS Performance Insights をその DB インスタンスで有効化する場合は `true`、それ以外の場合は `false` に設定します。

型: ブール値

必須: いいえ

PerformanceInsightsKMSKeyId

Performance Insights データを暗号化するための AWS KMS キー識別子。 AWS KMS キー ID は、 AWS KMS 暗号化 AWS KMS キーの Amazon リソースネーム (ARN)、 AWS KMS キー識別子、またはキーエイリアスです。

タイプ: 文字列

必須: いいえ

PreferredBackupWindow

`BackupRetentionPeriod` に応じた、自動バックアップが有効な場合に自動バックアップが作成される毎日の時間範囲を指定します。

タイプ: 文字列

必須: いいえ

PreferredMaintenanceWindow

週 1 回のシステムメンテナンスを実行できる時間範囲を世界標準時 (UTC) で指定します。

タイプ: 文字列

必須: いいえ

PromotionTier

既存のプライマリインスタンスの障害後に、Amazon DocumentDB レプリカをプライマリインスタンスに昇格される順序を指定する値。

タイプ: 整数

必須: いいえ

PubliclyAccessible

サポート外。Amazon DocumentDB は現在パブリックエンドポイントをサポートしていません。`PubliclyAccessible` の値は常に `false` です。

型: ブール値

必須: いいえ

StatusInfos.DBInstanceStatusInfo.N

リードレプリカのステータス。インスタンスがリードレプリカではない場合、これは空白です。

型: DBInstanceStatusInfo オブジェクトの配列

必須: いいえ

StorageEncrypted

インスタンスが暗号化されているかどうかを指定します。

型: ブール値

必須: いいえ

VpcSecurityGroups.VpcSecurityGroupMembership.N

インスタンスが属する VPC セキュリティグループ要素のリストを入力します。

型: VpcSecurityGroupMembership オブジェクトの配列

必須：いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBInstanceStatusInfo

サービス: Amazon DocumentDB (with MongoDB compatibility)

インスタンスのステータス情報のリストを入力します。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

Message

インスタンスにエラーがある場合のエラーの詳細。インスタンスがエラー状態にない場合、この値は空白です。

タイプ: 文字列

必須: いいえ

Normal

インスタンスが正常に動作している場合はブール値が `true`、インスタンスがエラー状態の場合は `false` です。

型: ブール値

必須: いいえ

Status

インスタンスのステータス。リードレプリカの `StatusType` の場合、値は `replicating`、エラー、`stopped`、または `terminated` となります。

タイプ: 文字列

必須: いいえ

StatusType

現在、この値は「`read replication`」です。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DBSubnetGroup

サービス: Amazon DocumentDB (with MongoDB compatibility)

サブネットグループに関する詳細情報。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

DBSubnetGroupArn

DB サブネットグループの Amazon リソースネーム (ARN)。

タイプ: 文字列

必須: いいえ

DBSubnetGroupDescription

サブネットグループの説明を入力します。

タイプ: 文字列

必須: いいえ

DBSubnetGroupName

サブネットグループの名前。

タイプ: 文字列

必須: いいえ

SubnetGroupStatus

サブネットグループのステータスを入力します。

タイプ: 文字列

必須: いいえ

Subnets.Subnet.N

サブネットグループ内の 1 つ以上のサブネットに関する詳細情報。

型: Subnet オブジェクトの配列

必須: いいえ

VpcId

サブネットグループの VPC (Virtual Private Cloud) ID を提供します。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Endpoint

サービス: Amazon DocumentDB (with MongoDB compatibility)

クラスターまたはインスタンスにアクセスするためのネットワーク情報。クライアントプログラムは、これらの Amazon DocumentDB リソースにアクセスするための有効なエンドポイントを指定する必要があります。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

Address

インスタンスの DNS アドレスが指定されます。

タイプ: 文字列

必須: いいえ

HostedZoneId

ホストゾーンを作成するときに Amazon Route 53 が割り当てる ID を指定します。

タイプ: 文字列

必須: いいえ

Port

データベースエンジンがリッスンするポートを指定します。

タイプ: 整数

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

EngineDefaults

サービス: Amazon DocumentDB (with MongoDB compatibility)

`DescribeEngineDefaultClusterParameters` オペレーションの呼び出しが成功した結果が含まれています。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

DBParameterGroupFamily

エンジンパラメータ情報を返すクラスターパラメータグループファミリーの名前。

タイプ: 文字列

必須: いいえ

Marker

以前のリクエストによって提供されたオプションのページ割りトークン。このパラメータを指定した場合、レスポンスには `MaxRecords` で指定された値まで、マーカーを超えるレコードのみが含まれます。

タイプ: 文字列

必須: いいえ

Parameters.Parameter.N

特定のクラスターパラメータグループファミリのパラメータ。

型: Parameter オブジェクトの配列

必須：いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Event

サービス: Amazon DocumentDB (with MongoDB compatibility)

イベントに関する詳細情報。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

Date

イベントの日付と時刻を指定します。

型: タイムスタンプ

必須: いいえ

EventCategories.EventCategory.N

イベントのカテゴリを指定します。

タイプ: 文字列の配列

必須: いいえ

Message

このイベントのテキストを入力します。

タイプ: 文字列

必須: いいえ

SourceArn

イベントの Amazon リソースネーム (ARN)

タイプ: 文字列

必須: いいえ

SourceIdentifier

イベントの発生元の識別子を入力します。

タイプ: 文字列

必須: いいえ

SourceType

このイベントのソースタイプを指定します。

タイプ: 文字列

有効な値: `db-instance | db-parameter-group | db-security-group | db-snapshot | db-cluster | db-cluster-snapshot`

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

EventCategoriesMap

サービス: Amazon DocumentDB (with MongoDB compatibility)

1 つ以上のイベントカテゴリ名が含むイベントソースタイプ。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

EventCategories.EventCategory.N

指定されたソースタイプのイベントカテゴリ。

タイプ: 文字列の配列

必須: いいえ

SourceType

返されたカテゴリに属しているソースタイプ。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

EventSubscription

サービス: Amazon DocumentDB (with MongoDB compatibility)

サブスクライブしたイベントの詳細情報。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

CustomerAwsId

Amazon DocumentDB イベント通知サブスクリプションに関連付けられている AWS 顧客アカウント。

タイプ: 文字列

必須: いいえ

CustSubscriptionId

Amazon DocumentDB イベント通知サブスクリプション ID。

タイプ: 文字列

必須: いいえ

Enabled

サブスクリプションが有効になっているかどうかを示すブール値。サブスクリプションが有効になっているかどうかを示す `true` の値。

型: ブール値

必須: いいえ

EventCategoriesList.EventCategory.N

Amazon DocumentDB イベント通知サブスクリプションのイベントカテゴリのリスト。

タイプ: 文字列の配列

必須: いいえ

EventSubscriptionArn

イベントサブスクリプションの Amazon リソースネーム (ARN)。

タイプ: 文字列

必須: いいえ

SnsTopicArn

Amazon DocumentDB イベント通知サブスクリプションのトピック ARN。

タイプ: 文字列

必須: いいえ

SourceIdsList.SourceId.N

Amazon DocumentDB イベント通知サブスクリプションのソース ID のリスト。

タイプ: 文字列の配列

必須: いいえ

SourceType

Amazon DocumentDB イベント通知サブスクリプションのソースタイプ。

タイプ: 文字列

必須: いいえ

Status

Amazon DocumentDB イベント通知サブスクリプションのステータス。

制約:

次のいずれかを指定できます: `creatingmodifyingdeletingactiveno-permissiontopic-not-exist`

`no-permission` ステータスは、Amazon DocumentDB が Amazon SNS トピックに投稿するアクセス許可を失ったことを示します。`topic-not-exist` ステータスは、サブスクリプションの作成後にトピックが削除されたことを示します。

タイプ: 文字列

必須: いいえ

SubscriptionCreationTime

Amazon DocumentDB イベント通知サブスクリプションが作成された時刻。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Filter

サービス: Amazon DocumentDB (with MongoDB compatibility)

より具体的な結果のリストを返すために使用される、フィルタ値の名前付きセット。フィルターを使用すると、ID などの特定の基準でリソースのセットを照合できます。

ワイルドカードがサポートされていません。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

Name

フィルターの名前。フィルター名では大文字と小文字が区別されます。

タイプ: 文字列

必須: はい

Values.Value.N

1 つまたは複数のフィルターの値。フィルタの値は大文字と小文字が区別されます。

タイプ: 文字列の配列

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

GlobalCluster

サービス: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB グローバルクラスターを表すデータ型。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

DatabaseName

新しいグローバルクラスター内のデフォルトのデータベース名。

タイプ: 文字列

必須: いいえ

DeletionProtection

新しいグローバルクラスターの削除保護設定。

型: ブール値

必須: いいえ

Engine

グローバルクラスターで使用される Amazon DocumentDB データベースエンジン。

タイプ: 文字列

必須: いいえ

EngineVersion

データベースエンジンのバージョンを示します。

タイプ: 文字列

必須: いいえ

GlobalClusterArn

グローバルクラスターの Amazon リソースネーム (ARN)。

タイプ: 文字列

必須: いいえ

GlobalClusterIdentifier

ユーザーが指定したグローバルクラスター識別子が含まれています。この識別子は、グローバルクラスターを識別する一意のキーです。

タイプ: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `[A-Za-z][0-9A-Za-z-:._]*`

必須: いいえ

GlobalClusterMembers.GlobalClusterMember.N

グローバルクラスタ内のセカンダリクラスターのクラスター ID のリスト。現在、1 つの項目に制限されています。

型: GlobalClusterMember オブジェクトの配列

必須: いいえ

GlobalClusterResourceId

グローバルデータベースクラスターの AWS リージョンリージョン固有のイミュータブルな識別子。この識別子は、クラスターの AWS KMS カスタマーマスターキー (CMK) にアクセスするたびに AWS CloudTrail ログエントリに記録されます。

タイプ: 文字列

必須: いいえ

Status

このグローバルクラスターの現在の状態を指定します。

タイプ: 文字列

必須: いいえ

StorageEncrypted

グローバルクラスターの記憶域暗号化設定。

型: ブール値

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

GlobalClusterMember

サービス: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB グローバルクラスターに関連付けられているプライマリおよびセカンダリークラスターに関する情報を含むデータ構造。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

DBClusterArn

各 Amazon DocumentDB クラスターの Amazon リソースネーム (ARN)。

タイプ: 文字列

必須: いいえ

IsWriter

Amazon DocumentDB クラスターが関連付けられている Amazon DocumentDB グローバルクラスターの主クラスタ (読み取り/書き込み機能を持つ) かどうか指定します。

型: ブール値

必須: いいえ

Readers.member.N

Amazon DocumentDB グローバルクラスターに関連付けられた各読み取り専用セカンダリクラスターの Amazon リソースネーム (ARN)。

タイプ: 文字列の配列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

OrderableDBInstanceOption

サービス: Amazon DocumentDB (with MongoDB compatibility)

インスタンスで使用できるオプション。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

AvailabilityZones.AvailabilityZone.N

インスタンスのアベイラビリティーゾーンのリスト。

型: AvailabilityZone オブジェクトの配列

必須: いいえ

DBInstanceClass

インスタンスのインスタンスクラス。

タイプ: 文字列

必須: いいえ

Engine

インスタンスのエンジンタイプ。

タイプ: 文字列

必須: いいえ

EngineVersion

インスタンスのエンジンバージョン。

タイプ: 文字列

必須: いいえ

LicenseModel

インスタンスのライセンスモデル。

タイプ: 文字列

必須: いいえ

StorageType

その DB クラスターに関連付けるストレージタイプ。

タイプ: 文字列

必須: いいえ

Vpc

インスタンスが VPC (Virtual Private Cloud) にあるかどうかを示します。

型: ブール値

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Parameter

サービス: Amazon DocumentDB (with MongoDB compatibility)

個々のパラメータに関する詳細情報。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

AllowedValues

パラメータの有効な値の範囲を指定します。

タイプ: 文字列

必須: いいえ

ApplyMethod

パラメータの更新を適用するタイミングを指定します。

型: 文字列

有効な値：`immediate | pending-reboot`

必須：いいえ

ApplyType

エンジン固有のパラメータタイプを指定します。

タイプ: 文字列

必須: いいえ

DataType

パラメータの有効なデータ型を指定します。

タイプ: 文字列

必須: いいえ

Description

パラメータの説明を入力します。

タイプ: 文字列

必須: いいえ

IsModifiable

パラメータを変更できるか (`true`) 、できない (`false`) かを示します。いくつかのパラメータには、それらの変更を妨げるセキュリティ上または運用上の影響があります。

型: ブール値

必須: いいえ

MinimumEngineVersion

パラメータを適用できる最も古いエンジンバージョン。

タイプ: 文字列

必須: いいえ

ParameterName

パラメータの名前を指定します。

タイプ: 文字列

必須: いいえ

ParameterValue

パラメータの値を指定します。CSV 形式のクラスターパラメータの 1 `AllowedValues` つ以上である必要があります。

次の値を指定できます:

- `enabled`: クラスターは TLS バージョン 1.0 から 1.3 を使用した安全な接続を受け入れます。
- `disabled`: クラスターは TLS を使用した安全な接続を受け入れません。
- `fips-140-3`: クラスターは、連邦情報処理規格 (FIPS) 発行 140-3 の要件に従ってのみ安全な接続を受け入れます。ca-central-1、us-west-2、us-east-1、us-east-2、us-gov-east-1、us-gov-

west-1 の各リージョンで Amazon DocumentDB 5.0 (エンジンバージョン 3.0.3727) クラスター以降でのみサポートされています。

- `tls1.2+`: クラスターは TLS バージョン 1.2 以降を使用した安全な接続を受け入れます。Amazon DocumentDB 4.0 (エンジンバージョン 2.0.10980) および Amazon DocumentDB 5.0 (エンジンバージョン 3.0.11051) 以降でのみサポートされています。
- `tls1.3+`: クラスターは TLS バージョン 1.3 以降を使用した安全な接続を受け入れます。Amazon DocumentDB 4.0 (エンジンバージョン 2.0.10980) および Amazon DocumentDB 5.0 (エンジンバージョン 3.0.11051) 以降でのみサポートされています。

タイプ: 文字列

必須: いいえ

Source

パラメータ値のソースを示します。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

PendingCloudwatchLogsExports

サービス: Amazon DocumentDB (with MongoDB compatibility)

設定がまだ保留中のログタイプのリスト。これらのログタイプは有効化中または無効化中です。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

LogTypesToDisable.member.N

有効化中のログタイプ。有効にすると、これらのログタイプは Amazon CloudWatch Logs にエクスポートされます。

タイプ: 文字列の配列

必須: いいえ

LogTypesToEnable.member.N

無効化中のログタイプ。無効化されたログタイプは CloudWatch Logs にエクスポートされません。

タイプ: 文字列の配列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

PendingMaintenanceAction

サービス: Amazon DocumentDB (with MongoDB compatibility)

リソースの保留中のメンテナンスアクションに関する情報を入力します。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

Action

リソースに対して使用可能な保留中のメンテナンスアクションのタイプ。

タイプ: 文字列

必須: いいえ

AutoAppliedAfterDate

アクションが適用されたときのメンテナンスウィンドウの日付。メンテナンスアクションは、この日以降の最初のメンテナンスウィンドウ中にリソースに適用されます。この日付を指定した場合、`next-maintenance` オプトインリクエストは無視されます。

型: タイムスタンプ

必須: いいえ

CurrentApplyDate

保留中のメンテナンスアクションがリソースに適用される有効日。

型: タイムスタンプ

必須: いいえ

Description

メンテナンスアクションについての詳細を提供する説明。

タイプ: 文字列

必須: いいえ

ForcedApplyDate

アクションが自動的に適用されたときのメンテナンスウィンドウの日付。メンテナンスアクションは、リソースのメンテナンスウィンドウに関係なく、この日にリソースに適用されます。この日付を指定した場合、`immediate` オプトインリクエストは無視されます。

型: タイムスタンプ

必須: いいえ

OptInStatus

リソースに対して受信されたオプトインリクエストのタイプを示します。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

PendingModifiedValues

サービス: Amazon DocumentDB (with MongoDB compatibility)

インスタンスの 1 つ以上の変更された設定。これらの変更された設定はリクエストされていますが、まだ適用されていません。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

AllocatedStorage

適用される、または現在適用されているインスタンスの新しい `AllocatedStorage` サイズが含まれています。

タイプ: 整数

必須: いいえ

BackupRetentionPeriod

自動バックアップを保持する保留日数を指定します。

タイプ: 整数

必須: いいえ

CACertificateIdentifier

DB インスタンスの 認定権限 (CA) 証明書の識別子を指定します。

タイプ: 文字列

必須: いいえ

DBInstanceClass

適用される、または現在適用されているインスタンスの新しい `DBInstanceClass` が含まれています。

タイプ: 文字列

必須: いいえ

DBInstanceIdentifier

適用される、または現在適用されているインスタンスの新しい DBInstanceIdentifier が含まれています。

タイプ: 文字列

必須: いいえ

DBSubnetGroupName

インスタンスの新しいサブネットグループ。

タイプ: 文字列

必須: いいえ

EngineVersion

データベースエンジンのバージョンを示します。

タイプ: 文字列

必須: いいえ

Iops

適用される、または現在適用されているインスタンスの新しいプロビジョンド IOPS 値を指定します。

タイプ: 整数

必須: いいえ

LicenseModel

インスタンスのライセンスモデル。

有効な値: license-included、bring-your-own-license、general-public-license

タイプ: 文字列

必須: いいえ

MasterUserPassword

インスタンスのマスター認証情報の保留中または現在進行中の変更が含まれています。

タイプ: 文字列

必須: いいえ

MultiAZ

Single-AZ インスタンスをマルチ AZ 配置に変更することを示します。

型: ブール値

必須: いいえ

PendingCloudwatchLogsExports

設定がまだ保留中のログタイプのリスト。これらのログタイプは有効化中または無効化中です。

型: PendingCloudwatchLogsExports オブジェクト

必須: いいえ

Port

インスタンスの保留ポートを指定します。

タイプ: 整数

必須: いいえ

StorageType

インスタンスに関連付けられるストレージタイプを指定します。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

ResourcePendingMaintenanceActions

サービス: Amazon DocumentDB (with MongoDB compatibility)

ApplyPendingMaintenanceAction の出力を表します。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

PendingMaintenanceActionDetails.PendingMaintenanceAction.N

リソースの保留中のメンテナンスアクションに関する詳細を提供するリスト。

型: PendingMaintenanceAction オブジェクトの配列

必須: いいえ

ResourceIdentifier

保留中のアクションが適用されるリソースの Amazon リソースネーム (ARN)。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ServerlessV2FeaturesSupport

サービス: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB エンジンのバージョンが異なる Amazon DocumentDB Serverless プロパティまたは制限を指定します。新しいクラスターまたはアップグレードされたクラスターで使用する Amazon DocumentDB バージョンを決定するときに、この属性の値をテストできます。また、既存のクラスターのバージョンを取得し、それらの機能を使用する前に、そのバージョンが特定の Amazon DocumentDB Serverless 機能をサポートしているかどうかを確認することもできます。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

MaxCapacity

Amazon DocumentDB Serverless クラスター内のインスタンスの Amazon DocumentDB キャパシティユニット (DCUs) の最大数。DCU 値は、32、32.5、33 などの半ステップ単位で指定できます。

型: 倍精度浮動小数点数

必須: いいえ

MinCapacity

Amazon DocumentDB Serverless クラスター内のインスタンスの Amazon DocumentDB キャパシティユニット (DCUs) の最小数。DCU 値は、8、8.5、9 などの半ステップ単位で指定できます。

型: 倍精度浮動小数点数

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ServerlessV2ScalingConfiguration

サービス: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB Serverless クラスターのスケーリング設定を設定します。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

MaxCapacity

Amazon DocumentDB Serverless クラスター内のインスタンスの Amazon DocumentDB キャパシティユニット (DCUs) の最大数。DCU 値は、32、32.5、33 などの半ステップ単位で指定できます。

型: 倍精度浮動小数点数

必須: いいえ

MinCapacity

Amazon DocumentDB Serverless クラスター内のインスタンスの Amazon DocumentDB キャパシティユニット (DCUs) の最小数。DCU 値は、8、8.5、9 などの半ステップ単位で指定できます。

型: 倍精度浮動小数点数

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ServerlessV2ScalingConfigurationInfo

サービス: Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB Serverless クラスターのスケーリング設定を取得します。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

MaxCapacity

Amazon DocumentDB Serverless クラスター内のインスタンスの Amazon DocumentDB キャパシティユニット (DCUs) の最大数。DCU 値は、32、32.5、33 などの半ステップ単位で指定できます。

型: 倍精度浮動小数点数

必須: いいえ

MinCapacity

Amazon DocumentDB Serverless クラスター内のインスタンスの Amazon DocumentDB キャパシティユニット (DCUs) の最小数。DCU 値は、8、8.5、9 などの半ステップ単位で指定できます。

型: 倍精度浮動小数点数

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Subnet

サービス: Amazon DocumentDB (with MongoDB compatibility)

サブネットグループに関する詳細情報。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

SubnetAvailabilityZone

サブネットのアベイラビリティーゾーンを指定します。

型: AvailabilityZone オブジェクト

必須: いいえ

SubnetIdentifier

サブネットの識別子を指定します。

タイプ: 文字列

必須: いいえ

SubnetStatus

サブネットのステータスを指定します。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

Tag

サービス: Amazon DocumentDB (with MongoDB compatibility)

キーと値のペアで構成される Amazon DocumentDB リソースに割り当てられるメタデータ。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

Key

タグに必要な名前。文字列値は、1 ～ 128 文字の Unicode 文字です。「`aws:`」または「`rds:`」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、「_」、「.」、「/」、「=」、「+」、「-」 (Java 正規表現: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)") のみ使用できます。

タイプ: 文字列

必須: いいえ

Value

タグの値 (オプション)。文字列値は、1 ～ 256 文字の Unicode 文字です。「`aws:`」または「`rds:`」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、「_」、「.」、「/」、「=」、「+」、「-」 (Java 正規表現: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)") のみ使用できます。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

UpgradeTarget

サービス: Amazon DocumentDB (with MongoDB compatibility)

インスタンスをアップグレードできるデータベースエンジンのバージョン。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

AutoUpgrade

`AutoMinorVersionUpgrade` が `true` に設定されているソース DB インスタンスにターゲットバージョンが適用されるかどうかを示す値。

型: ブール値

必須: いいえ

Description

インスタンスをアップグレードできるデータベースエンジンのバージョン。

タイプ: 文字列

必須: いいえ

Engine

アップグレードターゲットデータベースエンジンの名前。

タイプ: 文字列

必須: いいえ

EngineVersion

アップグレードターゲットのデータベースエンジンのバージョン番号。

タイプ: 文字列

必須: いいえ

IsMajorVersionUpgrade

データベースエンジンがメジャーバージョンにアップグレードされているかどうかを示す値。

型: ブール値

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

VpcSecurityGroupMembership

サービス: Amazon DocumentDB (with MongoDB compatibility)

VPC (Virtual Private Cloud) セキュリティグループメンバーシップに関するクエリのレスポンス要素として使用されます。

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

Status

VPC セキュリティグループのステータス。

タイプ: 文字列

必須: いいえ

VpcSecurityGroupId

VPC セキュリティグループの名前。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Amazon DocumentDB Elastic Clusters

Amazon DocumentDB Elastic クラスターでは、以下のデータ型がサポートされています。

- Cluster
- ClusterInList
- ClusterSnapshot
- ClusterSnapshotInList
- PendingMaintenanceActionDetails
- ResourcePendingMaintenanceAction
- Shard
- ValidationExceptionField

Cluster

サービス: Amazon DocumentDB Elastic Clusters

特定のエラスティッククラスターに関する情報を返します。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

adminUserName

エラスティッククラスター管理者の名前。

タイプ: 文字列

必須: はい

authType

エラスティッククラスターの認証タイプ。

タイプ: 文字列

有効な値: `PLAIN_TEXT | SECRET_ARN`

必須: はい

clusterArn

エラスティッククラスターの ARN 識別子。

タイプ: 文字列

必須: はい

clusterEndpoint

エラスティッククラスターへの接続に使用されている URL。

タイプ: 文字列

必須: はい

clusterName

エラスティッククラスターの名前。

タイプ: 文字列

必須: はい

createTime

エラスティッククラスターが作成された協定世界時 (UTC) 時刻。

タイプ: 文字列

必須: はい

kmsKeyId

エラスティッククラスターの暗号化に使用する KMS キー識別子。

タイプ: 文字列

必須: はい

preferredMaintenanceWindow

週 1 回のシステムメンテナンスを実行できる時間帯 (世界標準時 (UTC))。

形式: `ddd:hh24:mi-ddd:hh24:mi`

タイプ: 文字列

必須: はい

shardCapacity

各エラスティッククラスターシャードに割り当てられている vCPU の数。最大数は 64 です。使用できる値は、2、4、8、16、32、64 です。

タイプ: 整数

必須: はい

shardCount

エラスティッククラスターに割り当てられているシャードの数。最大数は 32 です。

タイプ: 整数

必須: はい

status

エラスティッククラスターのステータス。

タイプ: 文字列

有効な値: `CREATING | ACTIVE | DELETING | UPDATING | VPC_ENDPOINT_LIMIT_EXCEEDED | IP_ADDRESS_LIMIT_EXCEEDED | INVALID_SECURITY_GROUP_ID | INVALID_SUBNET_ID | INACCESSIBLE_ENCRYPTION_CREDS | INACCESSIBLE_SECRET_ARN | INACCESSIBLE_VPC_ENDPOINT | INCOMPATIBLE_NETWORK | MERGING | MODIFYING | SPLITTING | COPYING | STARTING | STOPPING | STOPPED | MAINTENANCE | INACCESSIBLE_ENCRYPTION_CREDENTIALS_RECOVERABLE`

必須: はい

subnetIds

エラスティッククラスターの Amazon EC2 サブネット ID。

タイプ: 文字列の配列

必須: はい

vpcSecurityGroupIds

このクラスターに関連付ける EC2 VPC セキュリティグループのリスト。

タイプ: 文字列の配列

必須: はい

backupRetentionPeriod

自動スナップショットが保持される日数。

タイプ: 整数

必須: いいえ

preferredBackupWindow

`backupRetentionPeriod` に応じて、自動バックアップが有効な場合に毎日自動バックアップの作成を実行する時間範囲を指定します。

タイプ: 文字列

必須: いいえ

shardInstanceCount

クラスター内のすべてのシャードに適用されるレプリカインスタンスの数。`shardInstanceCount` 値 が 1 の場合、ライターインスタンスが 1 つあり、それ以外の追加のインスタンスはすべてレプリカであり、読み取りや可用性の向上に使用できます。

タイプ: 整数

必須: いいえ

shards

クラスター内のシャードの数

型: Shard オブジェクトの配列

必須：いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ClusterInList

サービス: Amazon DocumentDB Elastic Clusters

Amazon DocumentDB エラスティッククラスターのリスト。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

clusterArn

エラスティッククラスターの ARN 識別子。

タイプ: 文字列

必須: はい

clusterName

エラスティッククラスターの名前。

タイプ: 文字列

必須: はい

status

エラスティッククラスターのステータス。

タイプ: 文字列

有効な値: `CREATING | ACTIVE | DELETING | UPDATING | VPC_ENDPOINT_LIMIT_EXCEEDED | IP_ADDRESS_LIMIT_EXCEEDED | INVALID_SECURITY_GROUP_ID | INVALID_SUBNET_ID | INACCESSIBLE_ENCRYPTION_CREDS | INACCESSIBLE_SECRET_ARN | INACCESSIBLE_VPC_ENDPOINT | INCOMPATIBLE_NETWORK | MERGING | MODIFYING | SPLITTING | COPYING | STARTING | STOPPING | STOPPED | MAINTENANCE | INACCESSIBLE_ENCRYPTION_CREDENTIALS_RECOVERABLE`

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ClusterSnapshot

サービス: Amazon DocumentDB Elastic Clusters

特定のエラスティッククラスタースナップショットに関する情報を返します。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

adminUserName

エラスティッククラスター管理者の名前。

タイプ: 文字列

必須: はい

clusterArn

エラスティッククラスターの ARN 識別子。

タイプ: 文字列

必須: はい

clusterCreationTime

エラスティッククラスターが作成された協定世界時 (UTC) 時刻。

タイプ: 文字列

必須: はい

kmsKeyId

KMS キー識別子は、KMS 暗号化キーの Amazon リソースネーム (ARN) です。KMS 暗号化キーを所有する Amazon アカウントと同じアカウントを使用して DB クラスターを作成する場合、KMS 暗号化キーの ARN の代わりに KMS キーのエイリアスを使用できます。ここで暗号化キーが指定されていない場合、Amazon DocumentDB は KMS がアカウント用に作成したデフォルトの暗号化キーを使用します。各アカウントは Amazon のリージョンごとにデフォルトの暗号化キーを持ちます。

タイプ: 文字列

必須: はい

snapshotArn

エラスティッククラスタースナップショットの ARN 識別子。

タイプ: 文字列

必須: はい

snapshotCreationTime

エラスティッククラスターが作成された協定世界時 (UTC) 時刻。

タイプ: 文字列

必須: はい

snapshotName

エラスティッククラスタースナップショットの名前。

タイプ: 文字列

必須: はい

status

エラスティッククラスタースナップショットのステータス。

タイプ: 文字列

有効な値: `CREATING | ACTIVE | DELETING | UPDATING | VPC_ENDPOINT_LIMIT_EXCEEDED | IP_ADDRESS_LIMIT_EXCEEDED | INVALID_SECURITY_GROUP_ID | INVALID_SUBNET_ID | INACCESSIBLE_ENCRYPTION_CREDS | INACCESSIBLE_SECRET_ARN | INACCESSIBLE_VPC_ENDPOINT | INCOMPATIBLE_NETWORK | MERGING | MODIFYING | SPLITTING | COPYING | STARTING | STOPPING | STOPPED | MAINTENANCE | INACCESSIBLE_ENCRYPTION_CREDENTIALS_RECOVERABLE`

必須: はい

subnetIds

エラスティッククラスターの Amazon EC2 サブネット ID。

タイプ: 文字列の配列

必須: はい

vpcSecurityGroupIds

このエラスティッククラスターに関連付ける EC2 VPC セキュリティグループのリスト。

タイプ: 文字列の配列

必須: はい

snapshotType

返されるクラススナップショットのタイプ。次のいずれかの値を指定できます。

- `automated` - Amazon DocumentDB が AWS アカウントに対して自動的に作成したすべてのクラススナップショットを返します。
- `manual` - AWS アカウント用に手動で作成したすべてのクラススナップショットを返します。

タイプ: 文字列

有効な値: `MANUAL | AUTOMATED`

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ClusterSnapshotInList

サービス: Amazon DocumentDB Elastic Clusters

エラスティッククラスタースナップショットのリスト。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

clusterArn

エラスティッククラスターの ARN 識別子。

タイプ: 文字列

必須: はい

snapshotArn

エラスティッククラスタースナップショットの ARN 識別子。

タイプ: 文字列

必須: はい

snapshotCreationTime

エラスティッククラスターが作成された協定世界時 (UTC) 時刻。

タイプ: 文字列

必須: はい

snapshotName

エラスティッククラスタースナップショットの名前。

タイプ: 文字列

必須: はい

status

エラスティッククラスタースナップショットのステータス。

タイプ: 文字列

有効な値: `CREATING | ACTIVE | DELETING | UPDATING | VPC_ENDPOINT_LIMIT_EXCEEDED | IP_ADDRESS_LIMIT_EXCEEDED | INVALID_SECURITY_GROUP_ID | INVALID_SUBNET_ID | INACCESSIBLE_ENCRYPTION_CREDS | INACCESSIBLE_SECRET_ARN | INACCESSIBLE_VPC_ENDPOINT | INCOMPATIBLE_NETWORK | MERGING | MODIFYING | SPLITTING | COPYING | STARTING | STOPPING | STOPPED | MAINTENANCE | INACCESSIBLE_ENCRYPTION_CREDENTIALS_RECOVERABLE`

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

PendingMaintenanceActionDetails

サービス: Amazon DocumentDB Elastic Clusters

保留中のメンテナンスアクションの詳細を取得します。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

action

保留中のメンテナンスアクションの特定のアクションを表示します。

タイプ: 文字列

必須: はい

autoAppliedAfterDate

アクションが適用されるメンテナンスウィンドウの日付を表示します。メンテナンスアクションは、この日以降の最初のメンテナンスウィンドウ中にリソースに適用されます。この日付を指定すると、すべてのNEXT_MAINTENANCEoptInTypeリクエストは無視されます。

タイプ: 文字列

必須: いいえ

currentApplyDate

保留中のメンテナンスアクションがリソースに適用される有効日を表示します。

タイプ: 文字列

必須: いいえ

description

メンテナンスアクションの詳細を説明する説明を表示します。

タイプ: 文字列

必須: いいえ

forcedApplyDate

メンテナンスアクションが自動的に適用される日付を表示します。メンテナンスアクションは、リソースのメンテナンスウィンドウに関係なく、この日にリソースに適用されます。この日付を指定すると、すべてのIMMEDIATEoptInTypeリクエストは無視されます。

タイプ: 文字列

必須: いいえ

optInStatus

リソースに対して受信したoptInTypeリクエストのタイプを表示します。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ResourcePendingMaintenanceAction

サービス: Amazon DocumentDB Elastic Clusters

リソースの保留中のメンテナンスアクションに関する情報を入力します。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

pendingMaintenanceActionDetails

リソースの保留中のメンテナンスアクションに関する情報を入力します。

型: PendingMaintenanceActionDetails オブジェクトの配列

必須: いいえ

resourceArn

保留中のメンテナンスアクションが適用されるリソースの Amazon DocumentDB Amazon リソースネーム (ARN)。

タイプ: 文字列

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Shard

サービス: Amazon DocumentDB Elastic Clusters

シャードの名前

内容

> **Note**
>
> 次のリストで必須のパラメーターを説明します。

createTime

そのシャードが作成された協定世界時 (UTC) 時刻。

タイプ: 文字列

必須: はい

shardId

シャードの ID

タイプ: 文字列

必須: はい

status

そのシャードの現在のステータス。

タイプ: 文字列

有効な値: `CREATING | ACTIVE | DELETING | UPDATING | VPC_ENDPOINT_LIMIT_EXCEEDED | IP_ADDRESS_LIMIT_EXCEEDED | INVALID_SECURITY_GROUP_ID | INVALID_SUBNET_ID | INACCESSIBLE_ENCRYPTION_CREDS | INACCESSIBLE_SECRET_ARN | INACCESSIBLE_VPC_ENDPOINT | INCOMPATIBLE_NETWORK | MERGING | MODIFYING | SPLITTING | COPYING | STARTING | STOPPING | STOPPED | MAINTENANCE | INACCESSIBLE_ENCRYPTION_CREDENTIALS_RECOVERABLE`

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ValidationExceptionField

サービス: Amazon DocumentDB Elastic Clusters

特定の検証例外が発生した具体的なフィールド。

内容

> **Note**
> 次のリストで必須のパラメーターを説明します。

message

このフィールドの検証例外を説明するエラーメッセージ。

タイプ: 文字列

必須: はい

name

検証例外が発生したフィールドの名前。

タイプ: 文字列

必須: はい

以下の資料も参照してください。

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

共通エラー

このセクションでは、すべての AWS サービスの API アクションに共通するエラーを一覧表示します。このサービスの API アクションに固有のエラーについては、その API アクションのトピックを参照してください。

AccessDeniedException

このアクションを実行する十分なアクセス権限がありません。

HTTP ステータスコード: 400

IncompleteSignature

リクエスト署名が AWS 標準に準拠していません。

HTTP ステータスコード: 400

InternalFailure

リクエストの処理が、不明なエラー、例外、または障害により実行できませんでした。

HTTP ステータスコード: 500

InvalidAction

リクエストされたアクション、またはオペレーションは無効です。アクションが正しく入力されていることを確認してください。

HTTP ステータスコード: 400

InvalidClientTokenId

指定された X.509 証明書または AWS アクセスキー ID がレコードに存在しません。

HTTP ステータスコード: 403

NotAuthorized

このアクションを実行するためのアクセス許可がありません。

HTTP ステータスコード: 400

OptInRequired

AWS アクセスキー ID には、サービスのサブスクリプションが必要です。

HTTP ステータスコード: 403

RequestExpired

リクエストの日付スタンプの 15 分以上後またはリクエストの有効期限 (署名付き URL の場合など) の 15 分以上後に、リクエストが到着しました。または、リクエストの日付スタンプが現在より 15 分以上先です。

HTTP ステータスコード: 400

ServiceUnavailable

サーバーの一時的な障害により、リクエストは失敗しました。

HTTP ステータスコード: 503

ThrottlingException

リクエストのスロットリングにより、リクエストが拒否されました。

HTTP ステータスコード: 400

ValidationError

入力が AWS サービスで指定された制約を満たしていません。

HTTP ステータスコード: 400

共通パラメータ

次のリストには、すべてのアクションが署名バージョン 4 リクエストにクエリ文字列で署名するために使用するパラメータを示します。アクション固有のパラメータは、アクションのトピックに示されています。署名バージョン 4 の詳細については、IAM ユーザーガイドの AWS API リクエストの署名を参照してください。

Action

実行するアクション。

型: 文字列

必須: はい

Version

リクエストが想定している API バージョンである、YYYY-MM-DD 形式で表示されます。

型: 文字列

必須: はい

X-Amz-Algorithm

リクエストの署名を作成するのに使用したハッシュアルゴリズム。

条件: HTTP 認可ヘッダーではなくクエリ文字列に認証情報を含める場合は、このパラメータを指定します。

型: 文字列

有効な値: `AWS4-HMAC-SHA256`

必須: 条件による

X-Amz-Credential

認証情報スコープの値で、アクセスキー、日付、対象とするリージョン、リクエストしているサービス、および終了文字列 ("aws4_request") を含む文字列です。値は次の形式で表現されます。[access_key/YYYYYYYYMMDD/リージョン/サービス/aws4_request]

詳細については、「IAM ユーザーガイド」の「署名付き AWS API リクエストの作成」を参照してください。

条件: HTTP 認可ヘッダーではなくクエリ文字列に認証情報を含める場合は、このパラメータを指定します。

型: 文字列

必須: 条件による

X-Amz-Date

署名を作成するときに使用する日付です。形式は ISO 8601 基本形式の YYYYMMDD'T'HHMMSS'Z' でなければなりません。例えば、日付 `20120325T120000Z` は、有効な X-Amz-Date の値です。

条件: X-Amz-Date はすべてのリクエストに対してオプションです。署名リクエストで使用する日付よりも優先される日付として使用できます。ISO 8601 ベーシック形式で日付ヘッダーが指定されている場合、X-Amz-Date は必要ありません。X-Amz-Date を使用すると、常に Date ヘッダーの値よりも優先されます。詳細については、「IAM ユーザーガイド」の AWS「API リクエスト署名の要素」を参照してください。

タイプ: 文字列

必須: 条件による

X-Amz-Security-Token

AWS Security Token Service () の呼び出しによって取得された一時的なセキュリティトークン AWS STS。 AWS STSの一時的なセキュリティ認証情報をサポートするサービスのリストについては、「IAM ユーザーガイド」の「IAM と連携するAWS のサービス」を参照してください。

条件: の一時的なセキュリティ認証情報を使用している場合は AWS STS、セキュリティトークンを含める必要があります。

タイプ: 文字列

必須: 条件による

X-Amz-Signature

署名する文字列と派生署名キーから計算された 16 進符号化署名を指定します。

条件: HTTP 認可ヘッダーではなくクエリ文字列に認証情報を含める場合は、このパラメータを指定します。

型: 文字列

必須: 条件による

X-Amz-SignedHeaders

正規リクエストの一部として含まれていたすべての HTTP ヘッダーを指定します。署名付きヘッダーの指定の詳細については、IAM ユーザーガイドの「署名付き AWS API リクエストの作成」を参照してください。

条件: HTTP 認可ヘッダーではなくクエリ文字列に認証情報を含める場合は、このパラメータを指定します。

型: 文字列

必須: 条件による

リリースノート

リリースノートでは、リリース日までの Amazon DocumentDB の機能、改善点、バグフィックスを解説しています。リリースノートには、Amazon DocumentDB エンジンのすべてのバージョンの更新情報が更新が行われるごとに記載されています。

Amazon DocumentDB エンジンパッチの最新バージョンは、次のコマンドを実行して確認できます。

```
db.runCommand({getEngineVersion: 1})
```

クラスターがエンジンの最新バージョンでない場合は、エンジンをアップグレードする保留中のメンテナンスがある可能性があります。詳細については、「 デベロッパーガイド」の「Amazon DocumentDB のメンテナンス」を参照してください。

[データベースの新機能] ページで新しい Amazon DocumentDB 機能をフィルタリングできます。[製品] フィールドは、[Amazon DocumentDB]を選択します。その後、**elastic clusters** や **vector search** などのキーワードを使用して検索します。

トピック

- 2025 年 7 月 30 日
- 2025 年 7 月 29 日
- 2025 年 6 月 18 日
- 2025 年 5 月 8 日
- 2025 年 4 月 2 日
- 2025 年 3 月 24 日
- 2025 年 2 月 6 日
- 2025 年 1月 28 日
- 2025年 1 月 15 日
- 2024 年 12 月 18 日
- 2024 年 11 月 12 日
- 2024 年 11 月 6 日
- 2024 年 11 月 1 日
- 2024 年 10 月 22 日

- 2024 年 9 月 18 日
- 2024 年 9 月 17 日
- 2024 年 8 月 22 日
- 2024 年 8 月 20 日
- 2024 年 8 月 8 日
- 2024 年 7 月 23 日
- 2024 年 7 月 22 日
- 2024 年 7 月 9 日
- 2024 年 7 月 8 日
- 2024 年 6 月 25 日
- 2024 年 5 月 29 日
- 2024 年 4 月 3 日
- 2024 年 2 月 22 日
- 2024 年 1 月 30 日
- 2024 年 1 月 10 日
- 2023 年 12 月 20 日
- 2023 年 12 月 13 日
- 2023 年 11 月 29 日
- 2023 年 11 月 21 日
- 2023 年 11 月 17 日
- 2023 年 11 月 6 日
- 2023 年 9 月 25 日
- 2023 年 9 月 20 日
- 2023 年 9 月 15 日
- 2023 年 9 月 11 日
- 2023 年 8 月 3 日
- 2023 年 7 月 13 日
- 2023 年 6 月 7 日
- 2023 年 5 月 10 日
- 2023 年 4 月 4 日

- 2023 年 3 月 22 日
- 2023 年 3 月 1 日
- 2023 年 2 月 27 日
- 2023 年 2 月 2 日
- 2022 年 11 月 30 日
- 2022 年 8 月 9 日
- 2022 年 7 月 25 日
- 2022 年 6 月 27 日
- 2022 年 4 月 29 日
- 2022 年 4 月 7 日
- 2022 年 3 月 16 日
- 2022 年 2 月 8 日
- 2022 年 1 月 24 日
- 2022 年 1 月 21 日
- 2021 年 10 月 25 日
- 2021 年 6 月 24 日
- 2021 年 5 月 4 日
- 2021 年 1 月 15 日
- 2020 年 11 月 9 日
- 2020 年 10 月 30 日
- 2020 年 9 月 22 日
- 2020 年 7 月 10 日
- 2020 年 6 月 30 日

2025 年 7 月 30 日

新機能

Amazon DocumentDB 5.0

Amazon DocumentDB インスタンスベースのクラスターは、オンデマンドの自動スケーリング設定である Amazon DocumentDB Serverless をサポートするようになりました。

詳細については、Amazon DocumentDB サーバーレスの使用「」と「新しいブログ記事」を参照してください。

Amazon DocumentDB Serverless は、エンジンパッチバージョン 3.0.14228 以降から使用できます。

2025 年 7 月 29 日

> **Note**
>
> 以下の Amazon DocumentDB エンジンパッチは、今後数週間ですべての Amazon DocumentDB リージョンに配信される予定です。このエンジンパッチがお客様のリージョンで利用可能になると、 の AWS ヘルスダッシュボード (AHD) を介して、 AWS Management Console および AWS アカウントのルートユーザーの E メールアドレスに E メールでサービスパッチ通知が送信されます。
> このエンジンパッチには、以下の新機能とバグ修正が含まれています。エンジンパッチがすべてのリージョンで利用可能になると、以下のリストと関連する関連ドキュメントが更新され、追加の機能発表が含まれる場合があることに注意してください。

新機能

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.15902)

- `$regexFindAll` 集計演算子のサポートを追加しました。
- インデックスの肥大化とコレクションベースの肥大化メトリクスの起動
- ガベージコレクション`AvailableMVCCIds`とエイジングコレクションに関するクラスターの全体的な状態に関する情報`LongestRunningGCProcess`を提供する CloudWatch メトリクスと を追加しました。
- `serverStatus` コマンドに「nvme_cache」フィールドが含まれるようになりました。このフィールドには、NVMe-backed インスタンスで NVMe キャッシュに書き込まれたページと書き込まれていないページの数が表示されます。
- `$group` および のフィールド数の制限を削除 `$project`
- 次の文字列演算子のサポート: `$trim`、`$rtrim`、、`$ltrim$regexFindAll$replaceOne`、、および `$replaceAll`
- 次の配列演算子のサポート: `$first`および `$last`

- 次のステージ演算子のサポート: `$collStats`

Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.11747)

- インデックスの肥大化とコレクションベースの肥大化メトリクスの起動
- ガベージコレクション`AvailableMVCCIds`とエイジングコレクションに関するクラスターの全体的な状態に関する情報`LongestRunningGCProcess`を提供する CloudWatch メトリクスと を追加しました。
- 次の文字列演算子のサポート: `$trim`、`$rtrim`、および `$ltrim`
- 次のステージ演算子のサポート: `$collStats`

バグの修正とその他の変更

true/false のブール値または 0/1 の数値を含む`newRoot`フィールドが出力ドキュメントにない`$replaceRoot`という動作のバグを修正しました。

2025 年 6 月 18 日

新機能

Amazon DocumentDB がイスラエル (テルアビブ) リージョンで利用可能になりました。

2025 年 5 月 8 日

新機能

Amazon DocumentDB が欧州 (ストックホルム) リージョンで利用可能になりました。詳細については、このブログ記事を参照してください。

2025 年 4 月 2 日

バグの修正とその他の変更

バックグラウンドインデックスを使用したベクトルインデックス作成のバグを修正しました。

2025 年 3 月 24 日

新機能

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.11800)

変更ストリームpostBatchResumeTokenで のサポートを追加しました。

詳細については、「postBatchResumeToken を用いた変更ストリームの再開」を参照してください。

2025 年 2 月 6 日

新機能

Amazon DocumentDB が と統合されました AWS Toolkit for Visual Studio Code。詳細については、「 ユーザーガイド」の「Amazon Amazon DocumentDB の使用」および「Amazon DocumentDB の使用」を参照してください。 AWS Toolkit for Visual Studio Code

2025 年 1月 28 日

新機能

ワンクリック接続のために、Amazon DocumentDB はインスタンスベースおよびエラスティッククラスター AWS CloudShell 用に と統合されました。

詳細については、以下のいずれかまたはすべてを参照してください。

- Amazon DocumentDB の使用開始
- Amazon DocumentDB エラスティッククラスターの開始方法
- 最新情報ブログ記事
- テクニカルハウツーブログ記事

2025年 1 月 15 日

新機能

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.11800)

Amazon DocumentDB は、ライターインスタンスの再起動によって読み取り可用性を維持するようになりました。リーダーインスタンスは、ライターインスタンスの再起動中に読み取りリクエストを引き続き処理するようになりました。

バグの修正とその他の変更

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.11800)

- 一括挿入の特殊なケースを処理するkillOpコマンドを修正しました。
- Amazon DocumentDB インスタンスのネットワーク I/O 関連のメモリ使用量が改善されました。
- `$text` フィルタークエリの `count` コマンドを修正しました。

Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.11096)

- 一括挿入の特殊なケースを処理するkillOpコマンドを修正しました。
- Amazon DocumentDB インスタンスのネットワーク I/O 関連のメモリ使用量が改善されました。

2024 年 12 月 18 日

新機能

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.5721)

Amazon DocumentDB インスタンスベースのクラスターが NVMe-backed R6GD インスタンスをサポートするようになりました。

詳細については、NVMe-backed インスタンス「」および「新しいブログ記事」を参照してください。

2024 年 11 月 12 日

新機能

Amazon DocumentDB エラスティッククラスターがバックグラウンドインデックスをサポートするようになりました。

2024 年 11 月 6 日

新機能

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.11051)

- 最小 TLS バージョン`tls1.2+`と `tls` `tls1.3+`クラスターパラメータのサポートを追加しました。
- フィールド名の文字 dollar($) と dot(.) のサポートを有効にしました。機能の違いについては、「」を参照してくださいフィールド名のドル ($) とドット (.)。

Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.10980)

- 最小 TLS バージョン`tls1.2+`と `tls` `tls1.3+`クラスターパラメータのサポートを追加しました。

2024 年 11 月 1 日

新機能

Amazon DocumentDB で Elastic クラスターのメンテナンスアクションがサポートされるようになりました。詳細については、「Amazon DocumentDB エラスティッククラスターのメンテナンス」を参照してください。

2024 年 10 月 22 日

新機能

Amazon DocumentDB は、ストレージネットワークスループットメトリクス `StorageNetworkReceiveThroughput`、`StorageNetworkTransmitThroughput`、および をサポートするようになりました`StorageNetworkThroughput`。詳細については、「CloudWatch メトリクスで Amazon DocumentDB インスタンスの使用状況を評価する」を参照してください。

2024 年 9 月 18 日

新機能

Amazon DocumentDB が アフリカ (ケープタウン) リージョンで利用可能になりました。詳細については、このブログ記事を参照してください。

Amazon DocumentDB が欧州 (スペイン) リージョンで利用可能になりました。詳細については、このブログ記事を参照してください。

2024 年 9 月 17 日

新機能

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.10696)

Amazon DocumentDB は、`runCommand` の `reIndex` でインデックスの再構築をサポートするようになりました。詳細については、「を使用したインデックスのメンテナンス `reIndex`」を参照してください。

> **Note**
>
> `reIndex` は Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.10696 以降) でのみサポートされています。

バグの修正とその他の変更

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.10696) および Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.10898)

- `$setOnInsert` は、更新オペレーションの結果として挿入中に `_id` フィールドをサポートするようになりました。
- Amazon DocumentDB 5.0 へのメジャーバージョンアップグレード後にストレージ容量を再利用できない問題を修正しました。

2024 年 8 月 22 日

新機能

Amazon DocumentDB 5.0 (すべてのエンジンパッチバージョン) および Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.5704)

Amazon DocumentDB グローバルクラスターは、クラスターのスイッチオーバーとマネージドクラスターのフェイルオーバーをサポートするようになりました。詳細については、「Amazon DocumentDB グローバルクラスターのスイッチオーバーの実行」および「Amazon DocumentDB グローバルクラスターのマネージドフェイルオーバーの実行」を参照してください。

> **Note**
>
> グローバルクラスターのスイッチオーバーとフェイルオーバーは、Amazon DocumentDB 4.0 および 5.0 でのみサポートされています。

2024 年 8 月 20 日

新機能

Amazon DocumentDB 3.6 (最小エンジンパッチバージョン 1.0.208662) では、TLS CA 証明書の更新にシステムの再起動が不要になりました。詳細については、「Amazon DocumentDB TLS 証明書の更新」を参照してください。

2024 年 8 月 8 日

新機能

Amazon DocumentDB Elastic クラスターが、アジアパシフィック (香港)、カナダ (中部)、欧州 (パリ) の各リージョンで利用可能になりました。詳細については、「Elastic クラスターのリージョン可用性」を参照してください。

2024 年 7 月 23 日

新機能

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.8126) および Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.10709)

- 長いインデックス名 (最大 255 文字) のサポートが追加されました。詳細については、命名に関する制約を参照してください。
- 最大接続限度を 2 倍に引き上げました。

インスタンスタイプ	元の限度	新しい制限
t3.medium	500	1,000
t4g.medium	500	1,000
r5.large	1700	3400
r5.xlarge	3500	7000
r5.2xlarge	7100	14200
r5.4xlarge	14200	28400
r5.8xlarge	28400	60000
r5.12xlarge	30000	60000
r5.16xlarge	30000	60000
r5.24xlarge	30000	60000
r6g.large	1700	3400
r6g.xlarge	3500	7000
r6g.2xlarge	7100	14200
r6g.4xlarge	14200	28400

インスタンスタイプ	元の限度	新しい制限
r6g.8xlarge	28400	60000
r6g.12xlarge	30000	60000
r6g.16xlarge	30000	60000

詳細については、「インスタンス制限」を参照してください。

バグの修正とその他の変更

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.8126)

リーダーの変更ストリームで `CurrentTime` と `ResumeToken` を同期させるロジックを強化しました。

2024 年 7 月 22 日

新機能

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.6742)

- DML 監査フィルタリングのサポートを追加しました。フィルター条件を設定することにより、すべての DML クエリをログに記録する代わりに特定の要件に基づいて DML 監査ログをフィルタリングできるようになりました。詳細については、「DML 監査イベントのフィルタリング」を参照してください。
- ドキュメント圧縮のサポートを以下に追加しました。
  - 最小圧縮しきい値の設定
  - 既存のコレクションで圧縮を有効にする (新しいドキュメントに適用)
  - クラスターレベルでデフォルトの圧縮設定を許可する

  詳細については、「コレクションレベルのドキュメント圧縮の管理」を参照してください。
- リーダーインスタンスで変更ストリームを使用するためのサポートを追加しました。詳細については、「セカンダリインスタンスでの変更ストリームの使用」を参照してください。

Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.10593)

- DML 監査フィルタリングのサポートを追加しました。フィルター条件を設定することにより、すべての DML クエリをログに記録する代わりに特定の要件に基づいて DML 監査ログをフィルタリングできるようになりました。詳細については、「DML 監査イベントのフィルタリング」を参照してください。

Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.208662)

Amazon DocumentDB MVU の db.r5.* および db.r6.* インスタンスのインデックス制限を排除しました。詳細については、「MVU 前提条件と制限事項」を参照してください。

バグの修正とその他の変更

Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.208662)

Amazon DocumentDB が、-NaN を有効な JSON トークンとして認識するようになりました。

2024 年 7 月 9 日

新機能

Amazon DocumentDB 4.0 (最小エンジンパッチバージョン 2.0.10179) および 5.0 (最小エンジンパッチバージョン 3.0.4780) では、TLS CA 証明書の更新にシステムの再起動が不要になりました。詳細については、「Amazon DocumentDB TLS 証明書の更新」を参照してください。

2024 年 7 月 8 日

新機能

Amazon DocumentDB Elastic クラスターが欧州 (ミラノ) リージョンで利用可能になりました。詳細については、「Elastic クラスターのリージョン可用性」を参照してください。

2024 年 6 月 25 日

新機能

IAM ARN AWS による認証は、サポートされているすべてのリージョンの Amazon DocumentDB インスタンスベースの 5.0 クラスターで使用できます。 ARNs 詳細については、「IAM ID を使用した認証」を参照してください。

2024 年 5 月 29 日

新機能

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.6742)

- `$regexMatch` および `$regexFind` 演算子のサポートを追加しました。
- 大きな整数に対処する際に、監査ログの完全精度を確保するサポートが追加されました。監査ログですべての数値の正確な数値表現を維持できるようになり、精度の低下が防止されます。

Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.10593)

- 大きな整数に対処する際に、監査ログの完全精度を確保するサポートが追加されました。監査ログですべての数値の正確な数値表現を維持できるようになり、精度の低下が防止されます。

2024 年 4 月 3 日

Amazon DocumentDB が、中東 (UAE) リージョンで利用可能になりました。詳細については、このブログ記事を参照してください。

新機能

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.5721)

- `bypassDocumentValidation` のサポートと、`$jsonSchema` の詳細なエラーメッセージのサポートをを追加しました。`bypassDocumentValidation` の詳細については、「bypassDocumentValidation」を参照してください。
- `$expr` のサポートが追加されました。

- $lookup で非相関結合のサポートが追加されました。
- $out 集約ステージで検証ルールを保持するサポートを追加しました。

Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.10392)

- bypassDocumentValidation のサポートを $jsonSchema に追加しました。bypassDocumentValidation の詳細については、「bypassDocumentValidation」を参照してください。
- $expr のサポートが追加されました。
- $lookup で非相関結合のサポートが追加されました。
- $out 集約ステージで検証ルールを保持するサポートを追加しました。

バグの修正とその他の変更

- mongo シェルバージョン 1.7 以降で db.coll.stats() を呼び出すときに発生していたエラーを修正しました。
- 同じ集約パイプラインの一部として $regex を含む変更ストリームクエリのメモリリークの問題を修正しました。

2024 年 2 月 22 日

新機能

Amazon DocumentDB Elastic クラスター

Amazon DocumentDB Elastic クラスターで、以下の操作がサポートされるようになりました。

- 読み取り可能なセカンダリシャードインスタンスレプリカ - 詳細については、「ステップ 1: エラスティッククラスターを作成する」のステップ 5b を参照してください。
- クラスターの開始/停止 - 詳細については、「Amazon DocumentDB Elastic クラスターの停止と開始」を参照してください。
- 設定可能なシャードインスタンス - 詳細については、「ステップ 1: エラスティッククラスターを作成する」のステップ 5b を参照してください。
- スナップショットの自動バックアップ - 詳細については、「Elastic クラスタースナップショットの自動バックアップの管理」を参照してください。

- スナップショットのコピー - 詳細については、「Elastic クラススナップショットのコピー」を参照してください。

2024 年 1 月 30 日

新機能

Amazon DocumentDB Elastic クラスター

Amazon DocumentDB Elastic クラスターが、次のリージョンで利用可能になりました。

- アジアパシフィック (ムンバイ)
- アジアパシフィック (ソウル)
- 南米 (サンパウロ)
- 欧州 (ロンドン)

詳細については、「エラスティッククラスターのリージョンとバージョンの可用性」を参照してください。

Amazon DocumentDB グローバルクラスター

グローバルクラスターが、 AWS GovCloud (米国東部) と AWS GovCloud (米国西部) の両方の AWS GovCloud (US) リージョンで利用可能になりました。

2024 年 1 月 10 日

新機能

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.4574、3.0.4780、3.0.4960)

- HNSW ベクトルインデックスのサポートを追加しました。詳細については、「Amazon DocumentDB のベクトル検索」を参照してください。
- 部分インデックスのサポートを追加しました。詳細については、「部分インデックス」を参照してください。
- `currentOp` コマンド内のコレクションでの GC ランタイムのサポートを追加しました。

- Amazon DocumentDB でのネイティブテキスト検索のテキストインデックスサポートを追加しました。詳細については、「Amazon DocumentDB でのテキスト検索の実行」を参照してください。
- `$jsonSchema` スキーマキーワード `type`、`allOf`、`oneOf`、`anyOf`、`not`、`maxItems`、`minItems`、`maxProperties`、`minProperties`、のサポートが追加されました。

  詳細については、「JSON スキーマ検証の使用」を参照してください。
- 算術演算子 `$ceil`、`$floor`、`$ln`、`$log`、`$log10`、`$sqrt`、`$exp` のサポートが追加されました。

  詳細については、「算術演算子」を参照してください。
- 条件式演算子 `$switch` のサポートを追加しました。
- 並列 `IVFFLAT` ベクトルインデックスビルドのサポートが追加されました。開発者ガイドからパラレル `IVFFLAT` ベクトルインデックスビルドの制限を削除してドキュメントを更新しました。

Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.10124、2.0.10179、2.0.10221)

- `currentOp` コマンド内のコレクションでの GC ランタイムのサポートを追加しました。
- `$jsonSchema` スキーマキーワード `type`、`allOf`、`oneOf`、`anyOf`、`not`、`maxItems`、`minItems`、`maxProperties`、`minProperties`、のサポートが追加されました。

  詳細については、「JSON スキーマ検証の使用」を参照してください。
- 算術演算子 `$ceil`、`$floor`、`$ln`、`$log`、`$log10`、`$sqrt`、`$exp` のサポートが追加されました。

  詳細については、「算術演算子」を参照してください。
- 条件式演算子 `$switch` のサポートを追加しました。

バグの修正とその他の変更

- `db.runCommand("dbstats")` の呼び出しで大文字と小文字を区別しない機能を復活させました。3.0.4960 または 2.0.10221 より前のバージョンのエンジンパッチを使用している Amazon DocumentDB 5.0 および 4.0 のお客様は、これらの最新のエンジンパッチを適用する必要があります。

- mongo シェルバージョン 1.7 以降で `db.coll.stats()` を呼び出すときに発生していたエラーを修正しました。開発者ガイドから mongo シェルの `db.coll.stats()` トラブルシューティングのヒントを削除してドキュメントを更新しました。

2023 年 12 月 20 日

その他の変更

Amazon DocumentDB 3.6 および 4.0 でのインプレースメジャーバージョンアップグレードのサポートを追加しました。詳細については、「Amazon DocumentDB のインプレースメジャーバージョンアップグレード」を参照してください。

2023 年 12 月 13 日

新機能

ワンクリック EC2 接続のサポートを追加しました。詳細については、「Amazon EC2 を使用した接続」を参照してください。

2023 年 11 月 29 日

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.3727)

新機能

ベクトル検索のサポートを追加しました。詳細については、こちらのブログ記事と、「Amazon DocumentDB 開発者ガイド」の「Amazon DocumentDB のベクトル検索」を参照してください。

2023 年 11 月 21 日

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.3727)

新機能

I/O 最適化ストレージのサポートを追加しました。詳細については、「Amazon DocumentDB 開発者ガイド」の「Amazon DocumentDB クラスターストレージ設定」を参照してください。

SageMaker Canvas とのノーコード機械学習の統合を追加しました。詳細については、「Amazon DocumentDB 開発者ガイド」の「Amazon SageMaker AI Canvas によるノーコード機械学習」を参照してください。

2023 年 11 月 17 日

新機能

Amazon DocumentDB が、 AWS GovCloud (米国東部) リージョンで利用可能になりました。詳細については、このブログ記事を参照してください。

バグの修正とその他の変更

Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.208570)

ユーザー定義のローカル変数名で、`$let` や `$filter` などの射影演算子の「_」(アンダースコア) をサポートするようになりました。

2023 年 11 月 6 日

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.3727) および 4.0 (エンジンパッチバージョン 2.0.9876)

新機能

- `$jsonSchema` スキーマキーワード `maxLength`、`minLength`、`maximum`、`minimum`、`exclusiveMaximum`、`exclusiveMinimum`、`items` のサポートが追加されました。

  JSON スキーマ検証は、インスタンスベースのクラスターでのみサポートされることに注意してください。
- `$convert` 集約パイプライン演算子とその省略形の派生演算子 `$toBool`、`$toInt`、`$toLong`、`$toDouble`、`$toString`、`$toDecimal`、`$toObjectId`、`$toDate` のサポートが追加されました。
- 集合式演算子 `$setDifference`、`$anyElementTrue`、`$allElementTrue` のサポートが追加されました。

バグの修正とその他の変更

-NaN から NaN へのチェンジストリームの更新が表示されない問題を修正しました。

2023 年 9 月 25 日

新機能

Amazon DocumentDB が、アジアパシフィック (香港) リージョンで使用できるようになりました。詳細については、このブログ記事を参照してください。

2023 年 9 月 20 日

新機能

Amazon DocumentDB 3.6 および 4.0 でのインプレースメジャーバージョンアップグレードのサポートを追加しました。詳細については、「Amazon DocumentDB のインプレースメジャーバージョンアップグレード」を参照してください。

2023 年 9 月 15 日

新機能

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.3140) および 4.0 (エンジンパッチバージョン 2.0.9686)

インスタンスベースのクラスターでのみ $jsonSchema スキーマ検証のサポートが追加されました。詳細については、「JSON スキーマ検証の使用」を参照してください。

2023 年 9 月 11 日

新機能

Amazon DocumentDB がアジアパシフィック (ハイデラバード) リージョンで利用可能になりました。詳細については、このブログ記事を参照してください。

2023 年 8 月 3 日

新機能

Amazon DocumentDB Elastic クラスター

- Amazon DocumentDB Elastic クラスターで、以下の操作がサポートされるようになりました。
  - `top`
  - `collStats`
  - `hint`
  - `dataSize`

  サポートされているコマンドと操作の一覧については、「Amazon DocumentDB でサポートされている MongoDB API、オペレーション、およびデータ型」を参照してください。
- 有効期限 (TTL) インデックスがサポートされるようになりました。
- インデックス `hints` がインデックス式でサポートされるようになりました。

2023 年 7 月 13 日

新機能

Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.1948)

- ドキュメント圧縮のサポートが追加されました。
- 並列インデックスビルドのサポートが追加されました。
- インデックスビルドステータスのサポートが追加されました。

Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.9259)

- 並列インデックスビルドのサポートが追加されました。

バグの修正とその他の変更

[Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.1948)]

- ユーザーがシステムコレクションにアクセスできない場合の Amazon DocumentDB Elasticクラスターの `createCollection` の認証問題を修正しました。
- セカンダリリージョンのインスタンスが同じプライマリリージョンのインスタンス名を使用できないという問題が修正されました。

Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.9259)

- 監査ログへの内部モニタリングクエリの追加を停止しました。

2023 年 6 月 7 日

バグの修正とその他の変更

Amazon DocumentDB 5.0

- r5 および t3.medium インスタンスが Amazon DocumentDB 5.0 でサポートされるようになりました。
- `engineVersion` オプションデフォルトは AWS SDK、 AWS CLI、および `5.0.0` です AWS CloudFormation。

2023 年 5 月 10 日

バグの修正とその他の変更

[Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.1361)]

- createIndex コマンドで `ignoreunknownindexoptions` のサポートが追加されました。
- 監査ログへの内部モニタリングクエリの追加を停止しました。
- ユーザー定義のローカル変数名で、`$let` や `$filter` などの射影演算子の「_」(アンダースコア) をサポートするようになりました。

2023 年 4 月 4 日

バグの修正とその他の変更

[Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.8934)]

- 進行中のワークロード中に DML 監査が有効になっている場合の問題を修正しました。
- ヒント付きの集計コマンドに文字列値が渡された場合の DML 監査の問題を修正しました。
- readwriteanydatabase ロールを持つユーザーが authorizedCollections オプションと nameOnly オプションの両方が true に設定されていると、`listCollections` コマンドが機能しないという問題が修正されました。
- フィールド名の数字文字列を適切に解析するように問題を修正しました。
- 実行時間の長いカーソルがガベージコレクションに影響する場合、そのカーソルはキャンセルされます。
- ユーザー定義のローカル変数名で、`$let` や `$filter` などの射影演算子の「_」(アンダースコア) をサポートするようになりました。

2023 年 3 月 22 日

新機能

Amazon DocumentDB Elastic クラスターが、アジアパシフィック (シンガポール)、アジアパシフィック (シドニー)、アジアパシフィック (東京) の各リージョンで利用可能になりました。詳細については、「エラスティッククラスターのリージョンとバージョンの可用性」を参照してください。

2023 年 3 月 1 日

新機能

[Amazon DocumentDB 5.0 (エンジンパッチバージョン 3.0.775)]

- Amazon DocumentDB 5.0 の追加
  - MongoDB 5.0 との互換 (MongoDB 5.0 API ドライバーのサポート)
  - クライアント側のフィールドレベル暗号化 (FLE) のサポート。Amazon DocumentDB クラスターにデータを書き込む前に、クライアント側でフィールドを暗号化できるようになりました。詳細については、「クライアント側のフィールドレベル暗号化」を参照してください。
  - 新しい集約演算子: `$dateAdd`、`$dateSubtract`
- すべてのインスタンスベースの Amazon DocumentDB クラスターとシャードベースのエラスティッククラスターのストレージ制限を 128 TiB に増大。
- Amazon DocumentDB 5.0 では、最初のネストレベルで `$elemMatch` 演算子によるインデックススキャンがサポートされるようになりました。クエリに 1 つのレベルの `$elemMatch` フィルター

しかなく、ネストされた $elemMatch クエリがインデックススキャンをサポートしていない場合、インデックススキャンがサポートされます。

インデックススキャンをサポートするクエリシェイプは以下のとおりです。

```
db.foo.find( { "a": {$elemMatch: { "b": "xyz", "c": "abc"} } })
```

インデックススキャンをサポートしないクエリシェイプは以下のとおりです。

```
db.foo.find( { "a": {$elemMatch: { "b": {$elemMatch: { "d": "xyz", "e":
 "abc"} }} } })
```

2023 年 2 月 27 日

バグの修正とその他の変更

Amazon DocumentDB 4.0

のサポートが追加されました AWS Lambda。詳細については、「変更ストリーム AWS Lambda でを使用する」を参照してください。

2023 年 2 月 2 日

バグの修正とその他の変更

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.208432)]

- 進行中のワークロード中に DML 監査が有効になっている場合の問題を修正しました。
- ヒント付きの集計コマンドに文字列値が渡された場合の DML 監査の問題を修正しました。
- readwriteanydatabase ロールを持つユーザーが authorizedCollections オプションと nameOnly オプションの両方が true に設定されていると、`listCollections` コマンドが機能しないという問題が修正されました。
- フィールド名の数字文字列を適切に解析するように問題を修正しました。
- 実行時間の長いカーソルがガベージコレクションに影響する場合、そのカーソルはキャンセルされます。

2022 年 11 月 30 日

新機能

Amazon DocumentDB Elastic クラスター

Amazon DocumentDB Elastic クラスターは、ユーザーが MongoDB シャーディング API を利用してクラスターをスケールアウトできるようにする新しいタイプの Amazon DocumentDB クラスターです。Elastic クラスターは、データと計算を基盤となる複数のコンピューティングインスタンスとボリュームに分散させることで、ペタバイトのストレージ容量で事実上あらゆる数の読み取りと書き込みを処理します。詳細については、「Amazon DocumentDB Elastic クラスターの使用」を参照してください。

2022 年 8 月 9 日

新機能

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.208152) および 4.0]

- Decimal128 データ型のサポートが追加されました。Decimal128 は、DocumentDB が利用可能なすべてのリージョンでサポートされている BSON データ型です。

  詳細については、「データ型」を参照してください。
- Amazon CloudWatch Logs を用いてデータベースイベントを監査するためのサポートを追加しました。Amazon DocumentDB で、データ操作言語 (DML) イベントとデータ定義言語 (DDL) イベントを Amazon CloudWatch Logs に記録できるようになりました。

  詳細については、このブログ記事を参照してください。

バグの修正とその他の変更

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.208152) および 4.0]

- 自分のパスワードを、`changeOwnPassword` 権限のある独自のパスワードで変更できるようになりました。

2022 年 7 月 25 日

新機能

Amazon DocumentDB 4.0

同じ DocumentDB クラスターボリュームを使用し、元のクラスターと同じデータを持つクローンを作成できるようになったため、クラスターをより速く作成できるようになりました。詳細については、「Amazon DocumentDB クラスターの管理」を参照してください。

2022 年 6 月 27 日

新機能

[Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.7509)]

Amazon DocumentDB は、使用パターンに基づいてデータベースのサイズを動的に変更します。データを追加するとスペースは最大 64 テビバイト (TiB) まで増加し、データを削除すると割り当てられるスペースが減少します。

2022 年 4 月 29 日

新機能

Amazon DocumentDB が中国 (北京) リージョンで利用可能になりました。詳細については、このブログ記事を参照してください。

2022 年 4 月 7 日

新機能

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.207836 と 1.0.208015) および 4.0 (エンジンパッチバージョン 2.0.6142 と 2.0.6948)]

Amazon DocumentDB Performance Insights は現在プレビュー段階にあります。7 日間のパフォーマンス履歴を追加費用なしでローリングウィンドウに保存できるようになりました。詳細については、「Performance Insights を使用したモニタリング」を参照してください。

2022 年 3 月 16 日

新機能

Amazon DocumentDB が欧州 (ミラノ) リージョンで利用可能になりました。詳細については、このブログ記事を参照してください。

2022 年 2 月 8 日

新機能

Amazon DocumentDB R6g と T4g インスタンスは、アジアパシフィック、南米、欧州で利用できるようになりました。詳細については、このブログ記事を参照してください。

2022 年 1 月 24 日

新機能

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.207684) および 4.0 (エンジンパッチバージョン 2.0.5170)]

- Amazon DocumentDB で無料トライアルが利用可能になりました。詳細については、「Amazon DocumentDB の無料トライアル」ページを参照してください。
- 次の API を含む拡張機能を地理空間クエリで使用できるようになりました。
  - `$geoWithin`
  - `$geoIntersects`
- 以下の MongoDB 演算子のサポートを追加しました。
  - `$mergeObjects`
  - `$reduce`

  詳細については、「Amazon DocumentDB による地理空間データのクエリ」を参照してください。

2022 年 1 月 21 日

新機能

[Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.5706)]

- Amazon DocumentDB Graviton2 (r6g.large、r6g.2xlarge、r6g.4xlarge、r6g.8xlarge、r6g.12xlarge、r6g.16xlarge、t4g.medium) インスタンスがサポートされるようになりました。

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.207781) および 4.0 (エンジンパッチバージョン 2.0.5706)]

- 以下の MongoDB API のサポートを追加しました。
  - `$reduce`
  - `$mergeObjects`
  - `$geoWithin`
  - `$geoIntersects`

2021 年 10 月 25 日

新機能

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.207780) および 4.0 (エンジンパッチバージョン 2.0.5704)]

- 以下の MongoDB API のサポートを追加しました。
  - `$literal`
  - `$map`
  - `$$ROOT`
- GeoSpatial クエリ機能をサポートしました。詳しくはこちらの ブログ投稿 をご覧ください
- ユーザー定義ロールによるアクセスコントロールに対応しています。詳しくはこちらの ブログ投稿 をご覧ください
- Amazon DocumentDB JDBC ドライバにより、Tableau などの BI ツールや SQL Workbench などのクエリツールからの接続を有効にします。

バグの修正とその他の変更

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.207780) および 4.0 (エンジンパッチバージョン 2.0.5704)]

- `$natural` と共に明示的な `.sort()` が存在する場合、`$natural` が正しくソートされるようにバグ修正
- 変更ストリームが `$redact` で動作するようにバグ修正
- `$ifNull` が空の配列でも動作するようにバグ修正
- 現在ログインしているユーザーを削除したり、進行中のアクティビティに対するユーザーの権限を取り消すと、リソースが過剰に消費され、サーバーがクラッシュする不具合を修正
- `listDatabase` と `listCollection`の権限チェックのバグ修正
- マルチキーエレメントのデデュープロジックのバグ修正

2021 年 6 月 24 日

新機能

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.207117) および 4.0 (エンジンパッチバージョン 2.0.3371)]

- r5.8xlarge と r5.16xlarge インスタンスがサポートされるようになりました。詳細については、ブログ記事「Amazon DocumentDB で r5.8xlarge と r5.16xlarge インスタンスがサポートされるようになりました」を参照してください。
- グローバルクラスターがサポートされるようになりました。これにより、リージョン全体の障害からのディザスタリカバリが可能になり、最も近い Amazon DocumentDB クラスターからの読み取りを許可することで低レイテンシーのグローバル読み取りが可能になります。グローバルクラスターは現在、南米 (サンパウロ)、欧州 (ミラノ)、中国 (北京)、中国 (寧夏) の各リージョンではサポートされていません。

2021 年 5 月 4 日

新機能

すべての新機能はこちらの ブログ投稿 でご確認ください。

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.207117) および 4.0 (エンジンパッチバージョン 2.0.3371)]

- `renameCollection`
- `$zip`
- `$indexOfArray`
- `$reverseArray`
- `$natural`
- `$hint` アップデートのサポート
- `distinct` のインデックススキャン

バグの修正とその他の変更

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.207117) および 4.0 (エンジンパッチバージョン 2.0.3371)]

- `$in` クエリのメモリ使用量を削減しました
- マルチキーインデックスのメモリリークを修正しました
- `$out` の説明プランとプロファイラーの出力を修正しました
- 信頼性を向上させるため、内部モニタリングシステムからの操作にタイムアウトを追加しました
- マルチキーインデックスに渡されるクエリ述語に影響する不具合を修正しました。

2021 年 1 月 15 日

新機能

[Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.722)]

- なし

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.206295)]

- `$lookup` の集約ステージでインデックスを使用できるようになりました
- 予測を含む `find( )` クエリは、インデックス (対象クエリ) からの指示で配信できます

- hint() と findAndModify の併用が可能です
- $addToSet 演算子のパフォーマンス最適化
- インデックス全体のサイズを縮小するための改善
- 新しい集約演算子:
  $ifNull、$replaceRoot、$setIsSubset、$setInstersection、$setUnion、$setEquals
- また、ユーザーは KillCursor ロールを必要とせずに自分でカーソルを終了することもできます

2020 年 11 月 9 日

新機能

すべての新機能はこちらの ブログ投稿 でご確認ください。

[Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.722)]

- MongoDB 4.0 の互換
- ACID トランザクション
- cluster(client.watch() または mongo.watch()) およびデータベースレベルの
  (db.watch()) 変更ストリームのサポート
- startAtOperationTime を使用して変更ストリームを開始または再開できます
- 変更ストリームの保存期間を 7 日間 (以前は 24 時間) に延長します
- AWS DMS Amazon DocumentDB 4.0 の ターゲット
- CloudWatch メトリクス:
  TransactionsOpen、TransactionsOpenMax、TransactionsAborted、TransactionsStarted、
- currentOp、ServerStatus、profiler のトランザクション用の新しいフィールド。
- $lookup の集約ステージでインデックスを使用できるようになりました
- 予測を含む find() クエリは、インデックス (対象クエリ) からの指示で配信できます
- hint() と findAndModify の併用が可能です
- $addToSet 演算子のパフォーマンス最適化
- インデックス全体のサイズを小さくするための改善。
- 新しい集約演算子:
  $ifNull、$replaceRoot、$setIsSubset、$setInstersection、$setUnion、$setEquals

- ListCollection と ListDatabase コマンドで、オプションとして authorizedCollections と authorizedDatabases パラメータを使用することができるようになり、それぞれ listCollections と listDatabase ロールを必要とすることなく、ユーザーがアクセス許可を持つコレクションとデータベースを一覧表示できるようになりました。
- また、ユーザーは KillCursor ロールを必要とせずに自分でカーソルを終了することもできます
- サブドキュメントの数値タイプの比較が、第 1 レベルのドキュメントの数値タイプの比較と一致するようになりました。Amazon DocumentDB 4.0 での動作が MongoDB と互換になりました。

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.206295)]

- なし

バグの修正とその他の変更

[Amazon DocumentDB 4.0 (エンジンパッチバージョン 2.0.722)]

- 位置演算子 $ を使用する場合、$setOnInsert は更新できなくなりました。Amazon DocumentDB 4.0 での動作が MongoDB と互換になりました。
- $createCollection とセット autoIndexId の問題を修正しました
- ネストされたドキュメントの投影
- ワーキングメモリーのデフォルト設定をインスタンスのメモリサイズに合わせてスケールするように変更しました
- ガベージコレクションの改善
- パス内の空のキーでのルックアップ、mongo との動作の違い
- タイムゾーンの動作の dateToString バグを修正
- ソート順を尊重するように $push (集計) を修正
- $currentOp の集計のバグを修正
- セカンダリのreadPreference の問題を修正しました
- $createIndex がコマンド発行時のデータベースと同じであることを検証する際の問題を修正しました
- minKey、maxKey 検索に失敗したときの一貫性のない動作を修正しました
- $size 演算子が複合配列で動作しないという問題が修正されました
- 正規表現による $in の否定に関する問題が修正されました

- ビューに対する $distinct コマンド実行の問題を修正しました
- 集約コマンドと検索コマンドで、欠落しているフィールドを異なる方法でソートする問題を修正しました
- 正規表現への $eq がタイプをチェックしない問題を修正しました
- タイムスタンプの順序位置動作の $currentDate バグを修正しました
- $currentDate のミリ秒単位の精度が修正されました

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.206295)]

- なし

2020 年 10 月 30 日

新機能

すべての新機能はこちらの ブログ投稿 でご確認ください。

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.206295)]

- クラスターレベル (client.watch() または mongo.watch())、およびデータベース (db.watch()) で変更ストリームカーソルを開く機能が追加されました。
- 変更ストリームの保存期間を 7 日間 (以前は 24 時間) に延長する機能

バグの修正とその他の変更

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.206295)]

- 一般的なケースでのさまざまなパフォーマンスの改善
- 対象を絞ったセキュリティの改善
- 複合インデックスの 2 番目のフィールドのスキップソートに関する問題を修正しました
- マルチキーインデックス (複合インデックスではない) の単一フィールドで等しくなるように標準インデックスを有効にする
- 認証競合状態を修正しました
- まれにガベージコレクションがクラッシュする原因となっていた問題を修正しました

- RBAC のセキュリティ改善
- `databaseConnectionsMax` メトリクスの追加
- `r5.24xlarge` インスタンス上の特定ワークロードのパフォーマンスの改善

2020 年 9 月 22 日

新機能

すべての新機能はこちらの ブログ投稿 でご確認ください。

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.206295)]

- `$out` 集約段階
- インスタンスあたりの接続数とカーソルの最大数を 10 倍まで増やしました

バグの修正とその他の変更

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.206295)]

- なし

2020 年 7 月 10 日

新機能

すべての新機能はこちらの ブログ投稿 でご確認ください。

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.206295)]

- クロスリージョンスナップショットのコピー

バグの修正とその他の変更

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.206295)]

- なし

2020 年 6 月 30 日

新機能

すべての新機能はこちらの ブログ投稿 でご確認ください。

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.206295)]

- t3 medium インスタンス

バグの修正とその他の変更

[Amazon DocumentDB 3.6 (エンジンパッチバージョン 1.0.206295)]

- t3 インスタンスのアイドルメモリ再利用
- 認証の改善
- SASL 認証のパフォーマンスの改善
- 可能な最大操作数を超えた場合の `currentOp` 問題を修正しました
- 一括更新と削除の `killOps` 問題を修正しました
- `$match` による `$sample` パフォーマンスの改善
- 編集段階での cond ケースでの `$$` サポートが修正されました
- 繰り返し発生するさまざまなクラッシュの根本原因を修正しました
- TTL スイープが改善され、IO とレイテンシーが削減されました
- `$unwind` のメモリ使用率が最適化されました
- コレクション統計とドロップインデックスの競合状態を修正しました
- 同時インデックス構築中の競合状態を修正しました
- インデックスの `hash_search` でまれに発生するクラッシュを修正しました

Amazon DocumentDB 開発者ガイドのドキュメント履歴

- API バージョン: 2014-10-31
- ドキュメント最新更新日: 2024 年 7 月 23 日

以下の表は、Amazon DocumentDB デベロッパーガイド の今回のリリースの内容をまとめたものです。

変更	説明	日付
AWS マネージドポリシーの更新 - ポリシーの変更	Amazon DocumentDB は、Elastic クラスターのフルアクセスポリシーを更新します。	2025 年 2 月 11 日
AWS マネージドポリシーの更新 - ポリシーの変更	Amazon DocumentDB は、Elastic クラスターのフルアクセスポリシーを更新します。	2024 年 2 月 21 日
AWS マネージドポリシーの更新 - ポリシーの変更	Amazon DocumentDB は、Elastic クラスターの読み取り専用ポリシーとフルアクセスポリシーを更新します。	2023 年 6 月 21 日
AWS マネージドポリシーの更新 - 新しいポリシー	Amazon DocumentDB に、Elastic クラスター用の新しい読み取り専用ポリシーが追加されました。	2023 年 6 月 8 日
AWS マネージドポリシーの更新 - 新しいポリシー	Amazon DocumentDB に、Elastic クラスター用の新しいフルアクセスポリシーが追加されます。	2023 年 6 月 5 日

MongoDB 5.0 互換性	Amazon DocumentDB は、MongoDB のバージョン 5.0 と互換性があります。	2023 年 3 月 1 日
ポリシーの更新	Amazon DocumentDB Elastic クラスター機能をサポートするために、AmazonDocDBConsoleFullAccess ポリシーが更新され、AmazonDocDB-ElasticServiceRolePolicy が新たに追加されました。	2022 年 11 月 30 日
Elastic クラスター	Amazon DocumentDB の分散ストレージシステム全体のデータのハッシュベースのパーティショニング (シャーディング) をサポートする新しい Elastic クラスター機能を追加しました。	2022 年 11 月 30 日
グローバルクラスター	グローバルクラスターの使用方法に関してのドキュメントを追加しました。	2021 年 6 月 2 日
イベントサブスクリプション	イベントサブスクリプションのドキュメントを追加しました。	2021 年 3 月 26 日
Version 3.6 アップグレード	ロールベースのアクセス制御、集約オペレータ、およびパフォーマンスにおけるバージョン 3.6 への文書化された改善。	2021 年 1 月 15 日
MongoDB 4.0 互換性	Amazon DocumentDB は今、MongoDB のバージョン 4.0 と互換性があります。	2020 年 11 月 9 日

入門ガイド	Amazon EC2, Robo3T AWS Cloud9、または Studio3T を使用して Amazon DocumentDB を開始するための新しい入門ガイド。	2020 年 8 月 15 日
Additional Availability Zones supported	Amazon DocumentDB は、アジアパシフィック (ソウル)(ap-northeast-2) の追加アベイラビリティーゾーンのサポートを追加しました。	2020 年 7 月 14 日
リージョン間のスナップショットのコピーのためのサポートを追加しました。	Amazon DocumentDB は、AWS リージョンを通してクラスタースナップショットをコピーするサポートを追加しました。詳細については、リージョン間のスナップショットのコピー を参照してください。	2020 年 7 月 10 日
T3 インスタンスクラスのサポートを追加しました。	Amazon DocumentDB をサポートするすべてのリージョンにおいて、T3 インスタンスタイプのためのサポートを追加しました。詳細については、「リージョン別にサポートされているインスタンスクラス」および「インスタンスクラスの仕様」を参照してください。	2020 年 6 月 30 日
のサポートを追加しました AWS GovCloud (US)。	Amazon DocumentDB が AWS GovCloud (US) リージョン (us-gov-west-1) で利用可能になりました。	2020 年 6 月 29 日

Added 16 new CloudWatch metrics.	Amazon DocumentDB は 16 の新しい Amazon CloudWatch メトリクスのサポートを追加しました。詳細については、「CloudWatch による Amazon DocumentDB のモニタリング」を参照してください。	2020 年 6 月 23 日
null 文字と $regex 演算子のサポートが追加されました。	Amazon DocumentDB に、文字列内の null 文字のサポートと $regex のインデックスを使用する機能が追加されました。Amazon DocumentDB でサポートされている MongoDB API と集約パイプライン機能については、「MongoDB との機能的な違い」を参照してください。	2020年6月22日
強化されたマルチキーのインデックス機能のサポートを追加しました。	Amazon DocumentDB は、2,048 バイトを超える配列のインデックス作成を含むマルチキーインデックス機能のサポートを強化し、同じ配列に複数のキーを持つ複合マルチキーインデックスを作成する機能を追加しました。詳細については、「MongoDB との機能的な違い」を参照してください。	2020 年 4 月 23 日
Amazon DocumentDB AWS CloudFormation スタックの削除保護のサポートを追加しました。	Amazon DocumentDB は、Amazon DocumentDB AWS CloudFormation スタックの作成時に削除保護を有効にするためのサポートを追加しました。	2020 年 4 月 20 日

ロールベースのアクセスコントロールのサポートを追加しました。	Amazon DocumentDB は、組み込みロールを使用して、ロールベースのアクセスコントロールのためのサポートを追加しました。	2020 年 3 月 26 日
カナダ (中部) (ca-central-1) で追加されたアベイラビリティーゾーンのサポートを追加しました。	Amazon DocumentDB が今、R5 クラスのインスタンスと 3 つのアベイラビリティーゾーンを用いて、カナダ (中部) リージョン (ca-central-1) で利用できるようになりました。	2020 年 3 月 26 日
さらに 2 つの MongoDB API のサポートを追加しました。	Amazon DocumentDB は、`$dateFromString` と `executionStats` の MongoDB API のためのサポートを追加しました。	2020 年 3 月 23 日
さらに 5 つの MongoDB API のサポートを追加しました。	Amazon DocumentDB は、`$objectToArray` 、`$arrayTo0bject` 、`$slice`、`$mod`、および `$range` の MongoDB API のためのサポートを追加しました。	2020 年 2 月 6 日
カナダ (中部) のサポートを追加しました。	Amazon DocumentDB は今、R5 クラスのインスタンスを用いてカナダ (中部)リージョン (ca-central-1) で使用できるようになりました。	2019 年 12 月 11 日

ChangeStreamLogSize のサポートを追加しました。	Amazon DocumentDB は、Cloudwatch メトリクスのための`ChangeStreamLogSize` のサポートを追加しました。	2019 年 11 月 22 日
欧州 (パリ) のサポートを追加しました。	Amazon DocumentDB は、R5 クラスのインスタンスを用いて、欧州 (パリ) リージョン (eu-west-3) で使用できるようになりました。	2019 年 10 月 30 日
アジアパシフィック (ムンバイ) リージョンのサポートが追加されました	Amazon DocumentDB は今、R5 クラスのインスタンスを用いて、アジアパシフィック (ムンバイ) リージョン (ap-south-1) で使用できるようになりました。	2019 年 10 月 17 日
3 つの追加の MongoDB API のサポートが追加されました	Amazon DocumentDB は `$addFields` 、`$concatArrays` 、および `$lookup` の MongoDB API のためのサポートを追加しました。	2019 年 10 月 16 日
アジアパシフィック (シンガポール) リージョンのサポートを追加しました	Amazon DocumentDB は今、R5 クラスのインスタンスを用いて、アジアパシフィック (シンガポール) リージョン (ap-southeast-1) で使用できるようになりました。	2019 年 10 月 14 日
TLS 証明書を更新するための新しいドキュメントが追加されました	新しい CA 証明書を使用して TLS 接続を作成するための CA 証明書の更新手順が追加されました。	2019 年 10 月 2 日

証明書の API サポートが追加されました	インスタンスの新しい証明書データ型、Amazon DocumentDB。詳細については、「DBInstance」を参照してください。	2019 年 10 月 1 日
クエリプロファイリングのサポート	Amazon DocumentDB は、クラスターのインスタンスとデータベース上で、でサポートされたオペレーションをプロファイルする機能を追加しました。	2019 年 8 月 19 日
アジアパシフィック (東京) に 3 番目の AZ を追加しました。	Amazon DocumentDB は、アジアパシフィック (東京) でのコンピューティングインスタンスのために 3 番目のアベイラビリティーゾーン (AZ) を追加しました。	2019 年 8 月 9 日
追加の Mongo API のサポート	`$in`、`$isoWeek`、`$isoWeekYear` 、`$isoDayOfWeek` 、および `$dateToString` の集計演算子と `$addToSet` の集計ステージを含む、追加のアグリゲーションパイプライン機能のサポートが追加されました。Amazon DocumentDB は、コレクションレベルの診断のための `top()` コマンドのサポートと、`expireAfterSeconds` のコマンドを使用してTTL インデックスの `collMod()` のパラメータサポートを変更する機能も追加しました。	2019 年 7 月 31 日

欧州 (ロンドン) のサポートを追加しました	Amazon DocumentDB は今、R5 クラスのインスタンスを用いて、欧州 (ロンドン) (eu-west-2) で使用できるようになりました。	2019 年 7 月 18 日
コードサンプルの追加	プログラムで Amazon DocumentDB に接続するための R および Ruby のコード例が追加されました。	2019 年 7 月 17 日
ベストプラクティスの追加	Amazon DocumentDB コストの管理に役立つベストプラクティスを追加しました。	2019 年 7 月 17 日
クラスターの停止と開始のサポート	Amazon DocumentDB は、開発とテスト環境のコスト管理に役立つクラスターの停止と開始のサポートを追加しました。	2019 年 7 月 1 日
クラスターの削除に対する保護のサポート	クラスターが誤って削除されることを防ぐために、Amazon DocumentDB に削除保護が追加されました。さらなる詳細については、次のトピックを参照してください： Amazon DocumentDB クラスターの作成、 Amazon DocumentDB クラスターの変更、Amazon DocumentDB クラスターの削除、および API トピックの DBCluster の `DeletionProtection` 。	2019 年 7 月 1 日

機能的な違いの更新	機能的な違いに暗示的なトランザクションが追加されました。	2019 年 6 月 26 日
機能的な違いの追加	Amazon DocumentDB でのストレージとインデックスの圧縮に関する注記が追加されました。	2019 年 6 月 13 日
サポートされるリージョンの追加	Amazon DocumentDB は今、R5 クラスのインスタンスを用いて、アジアパシフィック (シドニー) (ap-southeast-2) で使用できるようになりました。	2019 年 6 月 5 日
R5 インスタンスクラスをサポートするリージョンの追加	R5 インスタンスクラスのサポートが 4 つのリージョンに追加されました: 米国東部 (オハイオ)、米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド)。この変更により、R5 インスタンスは Amazon DocumentDB をサポートするすべてのリージョンでサポートされます。	2019 年 5 月 17 日

サポートされているその他のリージョン	R5 インスタンスクラスを用いて、2 つの追加リージョン、アジアパシフィック (東京) (ap-northeast-1) およびアジアパシフィック (ソウル) (ap-northeast-2) のサポートが追加されました。詳細については、「リージョン別にサポートされているインスタンスクラス」および「インスタンスクラスの仕様」を参照してください。	2019 年 5 月 8 日
接続コード例の追加	Amazon DocumentDB に接続するためのコード例が Java と C# 言語で追加されました。	2019 年 4 月 24 日
追加の Mongo API のサポート	7 つの集約文字列演算子 ($indexOfBytes 、$indexOfCP 、$strLenBytes 、$strLenCP 、$toLower、$toUpper、$split)、9 つの日時演算子 ($dayOfYear 、$dayOfMonth 、$dayOfWeek 、$year、$month、$hour、$minute、$second、$millisecond)、および $sample 集約パイプラインステージのサポートが追加されました。	2019 年 4 月 4 日
接続コード例の追加	Amazon DocumentDB に接続するための Python、Node.js、PHP、および Go のコード例を追加しました。	2019 年 3 月 21 日

フランクフルトリージョンと R5 インスタンスのサポート	R5 インスタンスクラスを用いて、欧州 (フランクフルト) リージョン (eu-central-1) のためのサポートが追加されました。詳細については、「リージョン別にサポートされているインスタンスクラス」および「インスタンスクラスの仕様」を参照してください。	2019 年 3 月 13 日
集約パイプライン演算子のサポート	新しい集約文字列演算子 ($concat、$substr、$substrBytes 、$substrCP 、$strcasecmp)、配列集約演算子 ($size)、集約グループアキュムレータ演算子 ($push)、および集約ステージ ($redact および $indexStats) のサポートが追加されました。位置配列演算子 ($[] および $[<identifier>]) と hint() のサポートも追加されました。	2019 年 2 月 28 日
エンジンのアップグレード	保留中のクラスターの変更の確認およびクラスターのエンジンバージョンのアップグレードに関するドキュメントが追加されました。	2019 年 2 月 15 日
監査イベント	Amazon CloudWatch Logs を用いてデータベースイベントを監査するためにサポートを追加しました。	2019 年 2 月 12 日

クイックスタート	を使用して Amazon DocumentDB を簡単に開始できるように、クイックスタートトピックを追加しましたAWS CloudFormation。	2019 年 1 月 11 日
パブリックリリース	これは、Amazon DocumentDB (MongoDB 互換)の最初のパブリックリリースです 。このリリースには、開発者ガイドと統合されたリソース管理 API リファレンスが含まれています。	2019 年 1 月 9 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。